JOURNAL OF CHEMICAL INFORMATION AND MODELING

Article

pubs.acs.org/jcim

"Social" Network of Isomers Based on Bond Count Distance: Algorithms

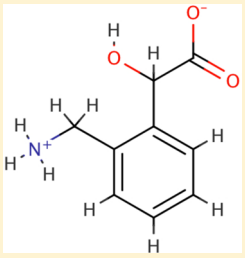
Tina M. Kouri,*[,†] Mahendra Awale,[‡] James K. Slyby,[¶] Jean-Louis Reymond,[‡] and Dinesh P. Mehta[¶]

[†]Department of Computer Science and Engineering, University of South Florida, 4202 E. Fowler Avenue, Tampa, Florida 33620, United States

[‡]Department of Chemistry and Biochemistry, University of Bern, Freiestrasse 3, 3013 Bern, Switzerland

[¶]Department of Electrical Engineering and Computer Science, Colorado School of Mines, 1500 Illinois Street, Golden, Colorado 80401, United States

ABSTRACT: This paper introduces the concept of an isomer network based on the reaction step counts between pairs of isomers as an alternative means to view and analyze isomer space. The computation of isomer networks is computationally expensive with respect to both run time and memory. Accordingly, this paper focuses on the design of algorithms to compute isomer networks and their analysis on structurally diverse subsets of isomers of nicotine, tyrosine, and phenmetrazine generated using molecular quantum number nearest neighbors. An analysis correlating isomer networks to extended connectivity fingerprints is also provided.



INTRODUCTION

There is a large body of research in the cheminformatics literature related to the counting, generation (listing), and sampling of isomers with papers dating back to the 1920s. In all three versions, the *input* is a chemical formula (e.g., C_4H_{10}). In the *counting* version of the problem, only the *number* of structures is calculated. In the *generation* version, all structures corresponding to the formula are returned. Because of combinatorial explosion, it may not be possible to list all structures. Accordingly, in the *sampling* version, a fraction of structures obtained through random techniques are returned. This work finds applications in structure elucidation and molecular design.[1] Structure elucidation uses information provided by an expert user to determine which isomers are consistent with the input information.[2] Molecular design is used to design molecules, such as drugs, which optimize specific characteristics.[2]

In this paper, we seek to fundamentally expand on this line of inquiry by extracting the *relationships* among a set of isomers based on reaction step counts and representing these in the form of a social network graph. *This paper introduces this idea and specifically focuses on the nontrivial algorithmic challenges of computing this graph, given a set of isomers.*

We begin with some basic definitions:

Definition. Two molecules are *isomers* if they have the same formula (i.e., composition of atoms), but different structures.[3] See Figure 1 for an example of two C_2H_6O isomers.

Definition. The *bond count distance* $BCD(I_1, I_2)$ between two isomers I_1 and I_2 is the *minimum* number of bonds that must be broken or formed to transform one isomer into the other. The bond count distance between the two isomers in Figure 1 is four. This is obtained by breaking the C−C and O−H bonds in I_1 and then forming O−C and C−H bonds to get I_2. This idea can be developed further to define the complete isomer network and *k*-isomer networks:

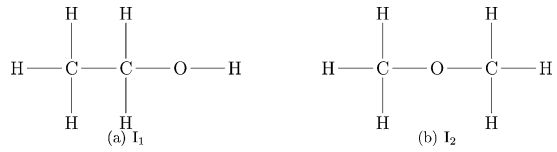


Figure 1. Two C_2H_6O isomers.

Definition. A *complete isomer network* defined on a set of *k* isomers is a complete weighted graph on *k* vertices (each corresponding to an isomer) with edge weights equal to the bond count distances between the corresponding isomers. Figure 2

Figure 2. Simple isomer network for the isomers shown in Figure 1

shows a trivial complete isomer network defined on the isomers of Figure 1. Table 1 shows a more complex complete isomer network of a set of 10 isomers in matrix form.

Definition. A *c-isomer network* (an order-*c* social network of isomers) defined on a set *S* of *k* isomers is the subgraph of the complete isomer network on *S* obtained by deleting edges of weight greater than *c*. For the example of Figure 1, the *c*-isomer network is identical to the complete isomer network shown in Figure 2 for $c \geq 4$. If $c < 4$, the *c*-isomer network simply consists of two isolated vertices. Figure 3 shows the four-isomer network for the complete isomer network of Table 1.

While the complete isomer network captures more information about the relationship among isomers, an order-*c* social network of isomers captures the more *relevant* information

Received: September 5, 2013
Published: December 18, 2013

   dx.doi.org/10.1021/ci4005173 | *J. Chem. Inf. Model.* 2014, 54, 57−68

Table 1. Complete Isomer Network for 10 Nicotine Isomers without Small Rings, Where Entry (i, j) Corresponds to BCD(i, j)[a]

	isomer number									
isomer number	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	0	4	4	6	4	6	6	6	4	8
2		0	6	4	6	4	8	8	8	6
3			0	4	4	6	8	4	8	10
4				0	6	4	10	8	10	8
5					0	4	4	8	8	8
6						0	8	10	10	4
7							0	4	4	4
8								0	4	6
9									0	6
10										0

[a]Note that the matrix is symmetric. The lower triangle of the matrix is omitted to improve clarity.

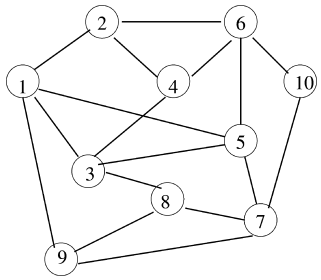


Figure 3. Order-4 isomer social network for the complete isomer network of Table 1. All edge weights (not shown in the figure) are 4.

about pairs of isomers with small bond count distances. For reasons that will become clearer, we will also see that it is easier to compute an order-c social network than the complete isomer network. In the remainder of this paper, when c is known or assumed to be fixed, we will refer to the order c social network of isomers simply as "social network of isomers" or "isomer social network".

Some observations about complete isomer networks and isomer social networks follow:

1. Complete isomer networks and isomer social networks can both be represented by *undirected* graphs because of symmetry; i.e., BCD(I_A, I_B) = BCD(I_B, I_A).
2. The complete isomer network satisfies the triangle inequality; i.e., BCD$(I_A, I_C) \leq$ BCD(I_A, I_B) + BCD(I_B, I_C) for any three isomers I_A, I_B, and I_C.
3. The length of the shortest path between I_A and I_B in an isomer social network is an upper bound on BCD(I_A, I_B). The shortest path from I_1 to I_8 in Figure 3 has length 8. This is an upper bound on BCD(I_1, I_8), which is actually 6. (The upper bound becomes tighter as the order of the social network increases.)
4. An order-i social network is a subgraph of an order-$(i + 1)$ social network. An order-0 social network has no edges (assuming there are no duplicate isomers). An order-c social network is identical to a complete isomer network if and only if c is greater than or equal to the largest bond count distances between any two isomers.
5. If isomers conform to the octet rule (i.e., atoms do not contain a net positive or negative charge and all bond cleavages are homolytic), bond count distances will always be an even number greater than or equal to 4. Under this formalism, we consider double bonds to be two bonds and triple bonds to be three bonds.
6. Social networks can be analyzed by tools such as *igraph* to compute attributes of the network including number of connected components, edge density, path length, betweenness, and vertex degree.[4]

■ ALGORITHM BACKGROUND

Overview. As stated earlier, the goal of this paper is to develop algorithms to compute a complete isomer network on a set of isomers, which amounts to computing the bond count distance between every pair of isomers. The problem of computing the bond count distance between isomers can in turn be reduced to (and is essentially equivalent to) the *automated reaction mapping* (ARM) problem. A reaction may be represented as a collection of reactant and product graphs where a set of reactant graphs is transformed into a set of product graphs. *The reaction-mapping problem is that of finding a mapping from the atoms of the reactant graphs to the atoms of the product graphs that minimizes the number of bonds broken or formed.*[5] To illustrate this concept, consider the reaction OH + CH_4 ⇔ H_2O + CH_3 which arises in combustion mechanisms and is shown in Figure 4. The

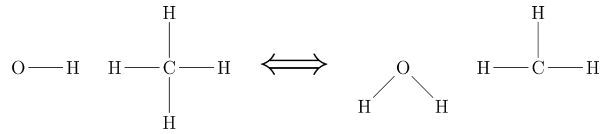


Figure 4. Simple chemical reaction: OH + CH_4 ⇔ H_2O + CH_3.

optimal mapping will break a C−H reactant bond and form an O−H product bond. The general ARM problem is known to be nondeterministic polynomial-time hard (NP-hard). However, algorithms that work fast in practice for most reactions have been previously developed by our group.[5] The time complexity of these algorithms includes a factor of $\Theta(n^{b+2})$, where n is is the size of the reaction and b is the number of bonds that are broken or formed in the reaction. Our experimental results show that, as expected, the performance of the algorithms is particularly impacted by b, which is in the exponent.

In principle, it should be possible to use existing ARM algorithms to compute an isomer network on a set of isomers by running these algorithms on each pair of isomers. We investigate this approach and discuss it briefly in this paper. However, it turns out that due to the size of each isomer (usually over 30 atoms), the number of bonds broken between pairs of isomers (often over 10 bonds per pair) and the number of isomers for a given set of atoms (usually over 10,000 isomers), it is impractical to use our ARM algorithms that were used for validating combustion mechanisms[5−7] (which usually contain a few thousand reactions and breaks fewer than three bonds per reaction).

We present an alternative approach for computing a complete isomer network that *considers all isomers simultaneously* rather than iterating through each pair individually. Although this algorithm performs significantly better than the naive algorithm described in the previous paragraph (as our experimental results will show), it has the potential to be expensive in both time and memory for large isomer sets. However, this approach can be easily adapted to compute isomer social networks instead, which as we have noted earlier can be analyzed by a number of graph metrics.

We begin by reviewing the concepts of graph isomorphism and canonical labeling that are central to the development of our algorithms.

dx.doi.org/10.1021/ci4005173 | J. Chem. Inf. Model. 2014, 54, 57−68

Graph Isomorphism and Canonical Naming. We review some of the relevant literature on graph isomorphism and canonical labeling and end with a description of canonical labeling using *degree neighborhoods*. This is a technique that was originally developed to obtain faster run times for the ARM problem, but is also used to develop algorithms to compute isomer social networks.

Definition. Two graphs, G_1 and G_2, are isomorphic if there is a bijection of the vertices of G_1 and the vertices of G_2, f: $V(G_1) \rightarrow V(G_2)$, such that two vertices, u and v are adjacent in G_1 if and only if $f(u)$ is adjacent to $f(v)$ in G_2.

No efficient algorithm has been found to determine if two general graphs are isomorphic.[8] There are special cases where determining whether two graphs are isomorphic can be solved in polynomial time. These special cases include triconnected planar graphs,[9] planar graphs,[10] interval graphs,[11] and trees.[12] Chemical graphs may not meet these special cases, and therefore, we cannot use these algorithms. The graph isomorphism problem is closely related to that of finding a canonical name for a graph.

Definition. A canonical naming l is a function over all graphs, mapping a graph G to a *canonical name* $l(G)$ such that for any two graphs, G_1 and G_2, G_1 is isomorphic to G_2 if and only if $l(G_1) = l(G_2)$.

If canonical names can be found for two graphs, the graphs can easily be checked for isomorphism by comparing their canonical names.[13] The problem of determining whether two graphs are isomorphic can be performed at least as fast as the problem of finding a canonical name for a graph. Algebraic methods for testing for graph isomorphism involve determining the automorphic groups of vertices. Previous work has bridged the gap between the canonical naming problem and the graph isomorphism problem by using the knowledge of automorphic groups of vertices to compute a canonical label of a graph.[13]

It has been shown that a theoretical polynomial time solution for graphs of bounded valence exists, but the research does not describe an algorithm for determining graph isomorphism.[14] Although these polynomial time solutions exist for graphs of bounded valence, no practical, polynomial time algorithm has been implemented.[15] A theoretical argument has been made that a canonical labeling can be computed for all molecular graphs in polynomial time.[16] This claim is based on the idea that a molecular graph can be transformed, in polynomial time, to a simple bounded valence graph.

Chemical Graph Isomorphism and Canonical Labeling. Several methods for labeling molecules have been developed in the literature for use in cheminformatics systems. These methods may also be used to solve the chemical graph isomorphism problem. Some graph isomorphism algorithms and our automated reaction mapping algorithms are designed for simple chemical graphs, but can be modified to support chemical multigraphs, either directly or by using Faulon's algorithm to convert a chemical multigraph into a simple graph in polynomial time.[16]

Morgan's Algorithm. One of the first canonical labeling algorithms for chemical graphs was proposed by Morgan.[17] The algorithm is based on node connectivity and the creation of unambiguous strings which describe a molecule. Morgan's algorithm may fail on highly regular graphs since it results in oscillatory behavior.[18]

Nauty. One of the most well-known and fastest algorithms for determining chemical graph isomorphism is Nauty[19] which is based on finding the automorphism groups of a graph.[20] The worst-case complexity of Nauty was analyzed and found to be exponential,[21] but in practice Nauty is much faster.

Bliss. The authors of Bliss improve the Nauty algorithm using an advanced data structure and incremental computations. The worst-case complexity of Bliss remains exponential, but the authors have shown Bliss performs better than Nauty on benchmark tests.[22]

Signature. Another well-known canonical naming algorithm for chemical graph isomorphism is Signature[15] which finds a canonical name using extended valence sequences. The authors state the algorithm has exponential worst-case complexity, but in practice it appears to run much faster.

SMILES. SMILES is a chemical notation system commonly used to describe the structure of a molecule.[23−25] However, the SMILES string used to describe a molecule is not necessarily unique.[26]

Fast Canonical Labeling Using Degree Neighborhoods. Next, we describe a canonical labeling algorithm that is fast but is not guaranteed to be accurate (i.e., two different molecules could get the same label). This is used as a filter prior to using one of the more expensive algorithms described above that guarantees that molecules have unique names. The algorithm presented is inspired by the random graph canonical labeling algorithms presented in refs 27 and 28 and the primitive refinement step used in Nauty[19] and was used to speed up ARM computations.[6,7]

The main idea of the degree neighborhood (DN) algorithm is to assign each atom a name based on its symbol and degree and the symbol and degree of each of its neighbors. The names of each atom are then used to assign a name to the molecule. For example, consider the CH_3O molecule in Figure 5a. We label

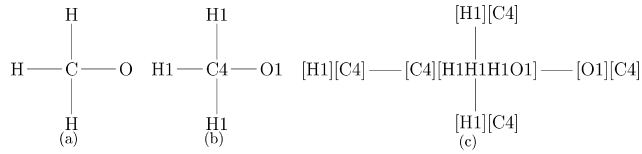


Figure 5. Degree neighborhood canonical labeling.

each atom, using its symbol and degree, Figure 5b. We then add to each atom's name, the symbol and degree of its neighbors lexicographically, Figure 5c. Now that each atom is named, we lexicographically sort the atom names to create the name for the molecule. The resulting canonical name is therefore:

[[C4][H1H1H1O1]][[H1][C4]][[H1][C4]][[H1][C4]][[O1][C4]]

Pseudocode for DN is presented in Algorithm 1.

As previously stated, the DN canonical label is not guaranteed to be accurate. We consider two types of errors:

Definition. A *type 1 error* occurs if an isomorphism algorithm determines two nonisomorphic molecules are isomorphic (i.e., a false positive).

Definition. A *type 2 error* occurs if an isomorphism algorithm determines two isomorphic molecules are not isomorphic (i.e., a false negative).

Theorem 1. *The degree neighborhood canonical naming algorithm will have no type 2 errors. (Proof is omitted.*[7]*)*

DN Naming can have type 1 errors. For example, the two nonisomorphic molecules in Figure 6 have the same canonical name:

[[C2][C2H1]][[C2][C2C3]][[C2][C3H1]]
[[C3][C2C3C3]][[C3][C2C3H1]][[C3][C3H1H1]]
[[H1][C2]][[H1][C2]][[H1][C3]][[H1][C3]]
[[H1][C3]]

dx.doi.org/10.1021/ci4005173 | J. Chem. Inf. Model. 2014, 54, 57−68

ALGORITHM 1: Degree Neighborhood Canonical Labeling

Input: Molecule (M) as an adjacency matrix and list of n atoms
Output: Canonical Name (C)

```
for i = 1 to n do
    /* labels[ ] holds the initial symbol and degree label for
       each vertex */
    labels[i] = LabelVertex(i);
end
for i = 1 to n do
    /* degreeLabels[ ] holds the neighbor list for each vertex
       */
    degreeLabels[i] = CreateArrayOfConnectedLabels(i);
    Sort(degreeLabels[i])
    /* canonicalVertexLabels[ ] holds the resulting canonical
       label for each vertex */
    canonicalVertexLabels[i] =
    "[" + labels[i] + "][" + ArrayToString(degreeLabels[i]) + "]"
end
Sort(canonicalVertexLabels)
C = ArrayToString(canonicalVertexLabels)
Return C
```

The molecules in Figure 6 are generated during intermediate steps of mapping two C_6H_5 isomers. Experimentally, we have found that DN

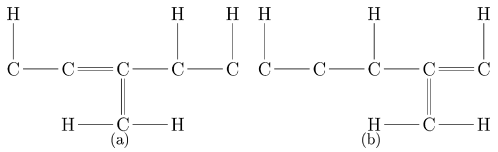


Figure 6. C_6H_5 molecule.

correctly distinguishes between nonisomorphic molecules over 99% of the time.

Theorem 2. *The worst-case time complexity for the degree neighborhood canonical labeling algorithm is $O(n \log n)$ where n is the number of bonds. (Proof is omitted.*[7]*)*

ARM-Based Computation of Complete Isomer Networks. As noted earlier, it is possible to compute complete isomer networks through repeated application of ARM algorithms[5−7] on each pair of isomers. Pseudocode is presented in Algorithm 2.

This approach may be used for small sets of isomers (e.g., sets of ten isomers), but will not scale for large sets of isomers (e.g., sets of 10 000 isomers) as shown. For example, using 2-CCV FDN MS (one of our two fastest ARM algorithms), it took 156 750.73 s (i.e., 43.54 h) to map 20 isomers of the nicotine isomers without small rings data set[29] (See Table 7 in Results and Discussion for additional results). In a set of 20 isomers, there are $\binom{20}{2}$ = 190 pairs of isomers. Therefore, the average time to map a pair of isomers is:

$$\frac{43.54\ \text{h}}{190} = 0.23\ \text{h}$$

We estimate the time required to map a large set of isomers (e.g., 10 000 isomers) using 2-CCV FDN MS by assuming that the average time to map a pair of isomers will remain the same. In a large set of isomers, there are $\binom{10\,000}{2}$ = 49 995 000 pairs of isomers. Therefore, the time to map all pairs of isomers can be estimated by

$$\begin{aligned} &49\,995\,000 \times 0.23\ \text{h} \\ &= 11\,456\,748.95\ \text{h} \\ &= 477\,364.54\ \text{days} \\ &= 1307.85\ \text{y} \end{aligned}$$

Clearly, it is not practical to attempt to map all pairs of isomers using 2-CCV FDN MS!

MAIN ALGORITHMIC CONTRIBUTION

Motivation. The graph algorithms literature provides examples of all-pairs problems that can be solved more efficiently by a direct simultaneous approach rather than through the repeated application of single-pair algorithms. We briefly mention two such examples:

1. *Shortest Path Problem:* In the shortest-paths problem we are given an edge-weighted graph with n vertices and m edges and are asked to find the shortest path between a pair of vertices. Algorithms for this problem such as the Bellman−Ford algorithm (which permits negative edge weights) actually solve the *single-source all-destination* version of the problem. These compute the shortest paths from a single source vertex to all vertices in the graph. The all-pairs version of the problem can be solved by running Bellman−Ford n times, with a different source

ALGORITHM 2: All-Pairs Algorithm using Previous Algorithms

Input: A set of k isomers
Output: A mapping from each isomer i to each isomer j for all i, j

```
for each isomer i in [1, k] do
    for each isomer j, with j > i do
        MapUsingARMAlgorithm(i, j)
    end
end
```


dx.doi.org/10.1021/ci4005173 | J. Chem. Inf. Model. 2014, 54, 57−68

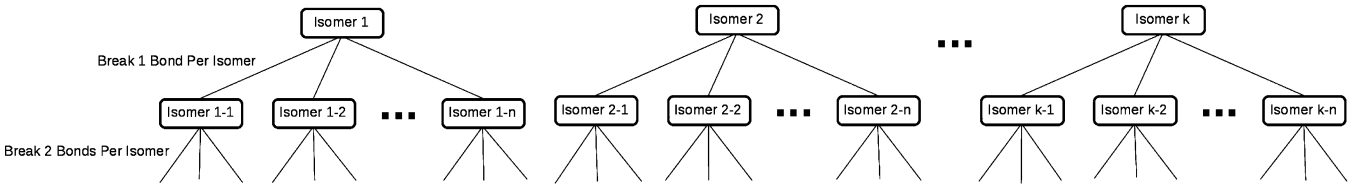


Figure 7. Simultaneous all−all algorithm.

ALGORITHM 3: Simultaneous All-All Algorithm

Input: A set of k isomers
Output: A mapping from each isomer i to each isomer j for all i, j

```
1  numBondsToBreak = 1
   /* partialResults is used to hold each candidate isomer for each
      iteration.  */
2  partialResults = [ ]
3  while all pairs of isomers are not mapped do
4  |  for each isomer m in [1, k] do
   |  |  /* bitPatterns is an array which holds each possible bit
   |  |     pattern which breaks numBondsToBreak for isomer m.
   |  |     */
5  |  |  bitPatterns = GetAllWaysToBreakBonds(numBondsToBreak, m)
6  |  |  for each bp ∈ bitPatterns do
   |  |  |  /* dnName holds the resulting DN when bp bonds are
   |  |  |     broken on isomer m.  */
7  |  |  |  dnName = computeDNname(bp, m)
   |  |  |  /* Add the new candidate to end of the partial
   |  |  |     results array.  */
8  |  |  |  addToResults(partialResults, dnName, bp)
9  |  |  end
10 |  end
11 |  sort(partialResults)
12 |  lookForDuplicates(partialResults)
13 |  numBondsToBreak = numBondsToBreak + 1
14 end
```

vertex in each run. Bellman−Ford takes $O(nm)$ time. Using it repeatedly (n times) would yield a run-time of $O(n^2m)$, which is $O(n^4)$ for dense graphs. In contrast, the Floyd−Warshall algorithm (which is specifically designed for the all-pairs version) only requires $O(n^3)$ time.[30] Clearly there is redundancy in the repeated Bellman−Ford approach which is eliminated by the direct Floyd−Warshall approach.

2. *Min-Cut Maximum-Flow Problem:* In the minimum-cut max-flow problem we are given a graph with edge weights that represent capacities, a source vertex s and a sink vertex t and are asked to find a maximum flow from s to t that does not exceed edge capacities and conserves flow on intermediate vertices. The maximum flow in a graph corresponds to the minimum $s - t$ cut. The overall minimum cut in a graph can be found by solving the $s - t$ problem for all $n(n-1)/2$ possible $\{s, t\}$ pairs. However, it tuns out that this problem can be solved more efficiently through only $n - 1$ applications of the $s - t$ version by using Gomory−Hu trees.[31] Once again, this suggests there is some redundant computation in the naive repeated approach which can be exploited in a direct solution.

This intuition guides us in the development of an algorithm that attempts to directly compute a complete isomer network rather than solving for each pair separately. We caution that, in contrast to the shortest-path and minimum-cut problems where the single-pair versions can be solved in polynomial time, the underlying single-pair ARM problem is NP-hard.[5] Although this suggests the existence of a better approach than that presented in algorithm 2, we should not expect to find a fast solution for computing isomer networks.

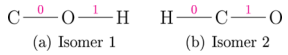
Simultaneous All−All Algorithm. We compute an isomer network by *simultaneously* processing all k isomers in the set. In the first step, we find all of the ways to break *one* bond in *each* isomer. Each such break results in a (possibly disconnected) *candidate* graph. If each isomer has n bonds, there are kn candidates. (Note that breaking different, but symmetric, bonds in an isomer will result in two copies of the same candidate. Duplicates resulting from symmetry are eliminated so that candidates generated from a single isomer are distinct.) If candidates obtained from two different isomers (say I_l and I_k) are identical, then we have BCD(I_l, I_k) = 2. In the second step, we generate additional candidates by breaking *two* bonds in each isomer. Duplicate candidates obtained from different isomers result in additional BCD values being computed. The algorithm continues increasing the number of bonds broken in each isomer until all pairs have been mapped. This process is guaranteed to terminate because breaking all n bonds in each isomer will result in a single candidate consisting of disconnected atoms. Figure 7 illustrates the approach.

We describe the algorithm in more detail. Consider iteration b of the **while** loop in line 3 of the pseudocode of algorithm 3 in which we compute all of the candidates resulting from breaking b bonds in each isomer. To achieve this, we use a bit string of length n where each bit corresponds to a bond. A "1" denotes a broken bond; whereas, a "0" denotes a unbroken bond. All bit strings with b 1s and $n - b$ 0s are then generated (line 5) and processed by the **for** loop of line 6. The DN name of each candidate (along with corresponding bit pattern that uniquely

dx.doi.org/10.1021/ci4005173 | J. Chem. Inf. Model. 2014, 54, 57−68

describes it) is computed (line 7) and stored in a *partialResults* array (line 8). After all of the ways to break b bonds in each isomer have been determined, we sort the *partialResults* array by DN-name (line 11) and look for mappings. A potential mapping is found when adjacent entries in the partial results array have the same DN name. When entries have the same DN name (i.e., a potential mapping), then their Nauty name is generated and used to determine definitively whether a mapping has been found.

For illustrative purposes, consider the *hypothetical* isomers shown in Figure 8. We first compute the DN name for each way



(a) Isomer 1 (b) Isomer 2

Figure 8. *Hypothetical* CHO isomers.

to break one bond in each isomer and add the bit pattern and DN name to the *partialResults* array. Since each of the two isomers has exactly two bonds, there will be four entries in the *partialResults* array. We next sort the array (see Table 2) and

Table 2. Sorted Partial Results Array

isomer number	bit pattern	DN name
1	(0)	[[C0]][[H1][O2]][[O1][H1]]
2	(1)	[[C1][H1]][[H1][C1]][[O0]]
1	(1)	[[C1][O1]][[H0]][[O1][C1]]
2	(0)	[[C1][O1]][[H0]][[O1][C1]]

check for duplicate DN names. Notice in the table that the last two isomer names are the same (breaking bond 1 on isomer 1 and breaking bond 0 on isomer 2). Since the two entries have the same DN name, their Nauty names are generated. The fact that they are identical means that we have found a mapping of cost 2.

Theorem 3. *The worst-case time complexity for the simultaneous all−all algorithm is* $O((k2^n)(n \log n) + (k2^n)\log(k2^n) + (k2^n)\mathrm{CN}(n))$ *where* k *is the number of isomers,* n *is the number of bonds per isomer, and* $\mathrm{CN}(n)$ *is the time to compute the Nauty name for a candidate with up to* n *bonds.*

Proof. In the worst case, the algorithm must break every bond of each isomer in the set of k isomers. That is, for each isomer we will generate 2^n candidates. For each of the 2^n candidates considered, for each of the k isomers, a DN name must be generated which takes $n \log n$ time. Therefore, the total time to compute the DN name for each candidate is $O((k2^n)(n \log n))$. Once we have a DN name for each candidate in the partial results array, the array must be sorted. It is well-known that sorting can be completed in $O(m \log m)$ time, where m is the number of items to be sorted.[30] Since there are $O(k2^n)$ elements in the partial results array, sorting takes $O((k2^n)\log(k2^n))$. The final step of the algorithm is to check for mappings by generating the Nauty name for any candidates which have the same DN name. In the worst case, a Nauty name will be generated for each candidate in the partial results array. Since there are $O(k2^n)$ elements in partial results array, this takes $O((k2^n)\mathrm{CN}(n))$. Therefore, the resulting worst-case time complexity is

$$O((k2^n)(n \log n) + (k2^n)\log(k2^n) + (k2^n)\mathrm{CN}(n))$$

Theorem 4. *The simultaneous all−all algorithm finds the minimum cost mapping for each pair of isomers in a set of isomers.*

Proof. The simultaneous all−all algorithm terminates once it has found a mapping for each pair of isomers. Since the algorithm examines the number of bonds broken per isomer in increasing order, the solution it returns for each pair is optimal.

Note that if the algorithm finds multiple mappings for a pair of isomers, only the first mapping is retained as the optimal mapping. If the user wishes to find all mappings of minimum cost, then only mappings which have the same cost as that first mapping are retained (i.e., more expensive mappings which may be found for a pair of isomers are discarded).

Optimizing Memory Usage. Clearly memory usage increases exponentially as the all-pairs algorithm progresses since we maintain $\sum_{i=1}^{b}\binom{\eta}{i}$ partial results for each isomer during each iteration of the algorithm, where n is the number of bonds in each isomer and b is the number of bonds broken per isomer.

Theorem 5. *The worst-case memory usage for the simultaneous all−all algorithm is* $O((k2^n)(s))$, *where* k *is the number of isomers we are mapping,* n *is the number of bonds per isomer, and* s *is the size of a partial results entry.*

Proof. For each of the k isomers, we generate 2^n candidates, since, in the worst case, we must break all of the bonds for each isomer. For each candidate, we create a partial results entry of size s.

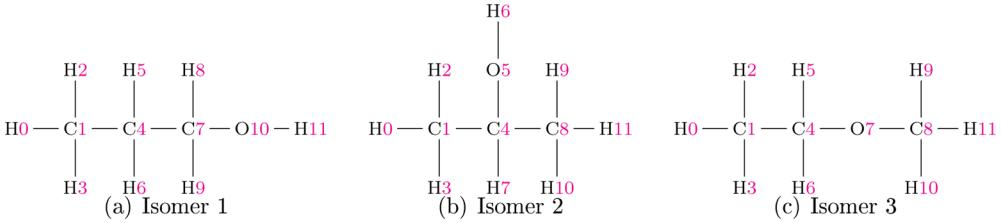
Due to the amount of memory being used, we must utilize methods to reduce the amount of data stored for each partial result.

Since we are looking at isomers, intuitively, we expect there to be many repeated DN vertex names. Rather than storing each complete vertex name, we can give each unique vertex name a unique index. For example, consider the three isomers shown in Figure 9 with the DN vertex names shown in Table 3. Although there are a total of 36 vertices in the three graphs, there are only 9 unique vertex names that are generated. Table 4 gives unique indices for each of the nine vertex names which are generated. Using the indices from Table 4, we can reduce the size of each of the DN names generated. For example, the DN name for isomer 1 is

[[C4][C4C4H1H1]][[C4][C4H1H1H1]]
[[C4][C4H1H1O2]][[H1][C4]][[H1][C4]]
[[H1][C4]][[H1][C4]][[H1][C4]][[H1][C4]]
[[H1][C4]][[H1][C4]][[O2][C4H1]]

Using the vertex indices shown in Table 4 the name can be reduced to

2,1,3,0,0,0,0,0,0,0,5,4



(a) Isomer 1 (b) Isomer 2 (c) Isomer 3

Figure 9. C_3H_8O isomers.

dx.doi.org/10.1021/ci4005173 | J. Chem. Inf. Model. 2014, 54, 57−68

Table 3. DN Vertex Names for Figure 9

vertex no.	isomer 1	isomer 2	isomer 3
0	[H1][C4]	[H1][C4]	[H1][C4]
1	[C4][C4H1H1H1]	[C4][C4H1H1H1]	[C4][C4H1H1H1]
2	[H1][C4]	[H1][C4]	[H1][C4]
3	[H1][C4]	[H1][C4]	[H1][C4]
4	[C4][C4C4H1H1]	[C4][C4C4H1O2]	[C4][C4H1H1O2]
5	[H1][C4]	[O2][C4H1]	[H1][C4]
6	[H1][C4]	[H1][O2]	[H1][C4]
7	[C4][C4H1H1O2]	[H1][C4]	[O2][C4C4]
8	[H1][C4]	[C4][C4H1H1H1]	[C4][H1H1H1O2]
9	[H1][C4]	[H1][C4]	[H1][C4]
10	[O2][C4H1]	[H1][C4]	[H1][C4]
11	[H1][O2]	[H1][C4]	[H1][C4]

Table 4. Indices for DN Vertices for Figure 9

0	1	2	3	4
[H1][C4]	[C4][C4H1H1H1]	[C4][C4C4H1H1]	[C4][C4H1H1O2]	[O2][C4H1]
5	6	7	8	
[H1][O2]	[C4][C4C4H1O2]	[O2][C4C4]	[C4][H1H1H1O2]	

Similarly, the name for isomer 2 becomes 6,1,1,0,0,0,0,0,0,0,5,4, and the name for isomer 3 becomes 1,3,8,0,0,0,0,0,0,0,0,7.

Another observation we make is that within a DN name, there are many repeated DN vertex names. For example, in the DN name for isomer 1, the vertex name [H1][C4] is repeated eight times. We can further reduce the size of a DN name by utilizing this observation. Rather than repeating a vertex index multiple times, we can indicate the quantity for a repeated vertex name. For example, the reduced name of isomer 1 is 2,1,3,7-0,5,4. Similarly the reduced name of isomer 2 is 6,2-1,7-0,5,4 and the reduced name of isomer 3 is 1,3,8,8-0,7.

This method significantly reduced the size of each *partialResults* entry, but an exponential number of entries are being generated, and main memory capacity was quickly consumed. We need to utilize disk space to store the partial results when mapping a large set of isomers. Once the *partialResults* array reaches the capacity of main memory, it is sorted and written to a temporary file on disk, and a new array is started. After all of the partial results have been generated for an iteration (e.g., all of the partial results which break b bonds per isomer), all of the temporary files are merged and checked for potential mappings. An external sorting algorithm based on mergesort[32,33] is used.

Current Limitations. Although we have made significant advances in computing a complete isomer network, our approach is not able to completely solve the problem since many pairs of isomers in large data sets have bond count distances of 10 or more. Recall that our algorithm has an exponential worst-case complexity, and therefore, the time for mapping isomers which break a large number of bonds is not practical using this approach. In addition, although we have significantly reduced the amount of storage required for each partial results entry, we will exceed the amount of storage available on disk when breaking a large number of bonds per isomer.

RESULTS AND DISCUSSION

Computational Experiment Setup. We carried out the experiments on a computer running Microsoft Windows Vista Home Premium with a 2.66 GHz Intel Core 2 Quad Processor and 4GB of RAM. Note that for all of the databases, we used the Nauty[19] chemical graph canonical naming algorithm to test for isomorphism.

The molecules nicotine, tyrosine, and phenmetrazine were used as examples. Isomers of these molecules were retrieved from the Chemical Universe Database GDB-13, which contains all possible molecules up to 13 atoms of C, N, O, S, and Cl following simple rules of chemical stability and synthetic feasibility.[34] We used the MQN-browser available at www.gdb.unibe.ch[35] and retrieved either: (1) all isomers or (2) the first 10 000 MQN-nearest neighbors of each molecule where the 3- and 4-membered rings are either (a) excluded or (b) not excluded. Molecular quantum numbers (MQN) is a set of 42 integer value descriptors of molecules suitable for the classification and visualization of large databases.[36−38] MQN-neighbor isomers of any molecule represent a subset of possible isomers with overall similarities in terms of shape and pharmacology.[39,40] Figure 10

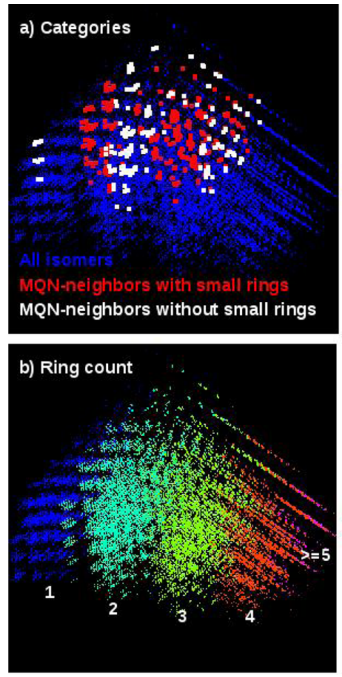


Figure 10. Map of nicotine isomer categories and their properties in MQN-space.

shows a projection of the principal component analysis map (PC1, PC2) from the MQN-data for the 10 000 selected isomers for nicotine versus all isomers of nicotine found in GDB-13, which shows that these isomers spread across a diversity of structural types.

Table 5 provides additional information about each of the data sets tested.

Correlation between Isomer Networks and ECFPs. We consider the statistical correlation between two metrics for computing the distances between two isomers: the first is the reaction step count between isomers defined in this paper. The second is the Tanimoto distance between the extended connectivity fingerprints (ECFPs) of isomers.[41] The latter is an established technique for computing chemical similarities. Given k isomers, we can construct two $k \times k$ matrices with elements RSC_{ij} and $ECFP_{ij}$ corresponding respectively to the two distance metrics defined above for isomers i and j, for $1 \leq i, j \leq k$. Both

dx.doi.org/10.1021/ci4005173 | J. Chem. Inf. Model. 2014, 54, 57−68

Table 5. Isomer Data Sets

isomer set	total no. of isomers	sample size	no. atoms per isomer	no. bonds per isomer
nicotine without small rings ($C_{10}H_{15}N_2$)	219490	2934	27	31
nicotine with small rings ($C_{10}H_{15}N_2$)	507964	10000	27	31
phenmetrazine ($C_{11}H_{16}NO$)	10213951	10000	29	33
tyrosine ($C_9H_{11}NO_3$)	7682352	10000	24	28

matrices are symmetric and have zero diagonal elements. Both are also distance matrices that satisfy the triangle inequality. We are interested in computing the correlation between the two matrices. However, traditional techniques for computing the correlation between two variables (e.g., the work of Navidi[42]) cannot be applied here because of the implicit dependencies within each of the two distance matrices. We instead employ the Mantel test described by Schneider,[43] which is specifically designed for determining the correlation between distance matrices. The version of the Mantel test we used computes the standardized Mantel statistic r_M defined as

$$r_M = \frac{1}{d-1}\sum_{i=1}^{k-1}\sum_{j=i+1}^{k}\left(\frac{RSC_{ij}-\overline{RSC}}{s_{RSC}}\right)\left(\frac{ECFP_{ij}-\overline{ECFP}}{s_{ECFP}}\right)$$

where d is the number of distances in the upper triangular part of each matrix (for k isomers, $d = k(k-1)/2$), and s_{RSC} and s_{ECFP} are the standard deviations for RSC and ECFP, respectively. We note that this is effectively the standard measure of correlation r[42] which has the following properties:

1. r_M is bounded between −1 and +1.
2. An r_M above +0.5 indicates a strong positive correlation, while an r_M less than −0.5 indicates a strong negative correlation.
3. If r_M is close to 0, it suggests that no correlation is present in the data.

As noted, because of the dependencies within the two distance matrices, we do not exclusively use r_M to make statistically valid assertions. In addition, the Mantel test employs a permutation test to also compute the p-value, the probability of obtaining a test statistic that is at least as extreme as the r_M value initially observed for the two matrices. The permutation test consists of permuting the isomers in one of the two distance matrices (say ECFP) and computing r_M between the unchanged RSC and the permuted ECFP matrices. p is the fraction of r_M values computed that are greater than or equal to the r_M computed between the original RSC and ECFP matrices. Typically $p < 0.05$ (or in some cases $p < 0.001$) would cause us to reject the null hypothesis (the default assumption that there is no relationship between the two matrices). Ideally the permutation test evaluates all $k!$ permutations of the ECFP matrix. This is not feasible, however, for large k. In this case, we use a randomly generated subset of permutations to compute p. The p-value becomes more reliable as more random permutations are performed.

We used the Quantitative Insights Into Microbial Ecology (QIIME),[44] an open-source software package, to perform the Mantel test. Upon completion of the Mantel test, the output of QIIME provides a file with the final r_M statistic and the p-value. We computed a eight-isomer network for a data set of 10 000 tyrosine isomers using the techniques described in this paper. This was used to populate the RSC matrix as follows: if there is an edge in the network between isomers i and j, $RSC_{ij} = RSC_{ji}$ = the weight of that edge; if there is no edge between i and j, we set $RSC_{ij} = RSC_{ji}$ to be the length of the shortest path from i to j. This quantity is an upper bound on the actual reaction step count. We also created a second matrix of the same size using the Tanimoto distances between the ECFPs of each tyrosine isomer using Jchem.[45] Both matrices are of size 10 000 × 10 000. As this is too large to perform a meaningful permutation test, we instead executed the Mantel test on 1000 sets of 20 randomly selected isomers. In each case, the test was executed with 10 000 permutations. This generated an average r_M statistic of 0.721 974 17 with an average p-value of 0.0001. The r_M statistic is greater than 0.5, indicative of a high correlation, and the p-value is significantly less than the recommended values of 0.05 and 0.001. These results are indicative of a strong correlation between the two matrices.

Visual Correlation between Isomer Networks and ECFPs. In addition to the statistical correlation described above, we also present a visual means of understanding the relationship between isomer networks and ECFPs. For this analysis, we selected six isomers of tyrosine labeled M1, M2, M4, M42, M168, and M8758 as shown in Figures 11−16.

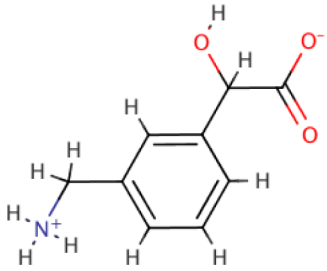


Figure 11. Isomer M1.

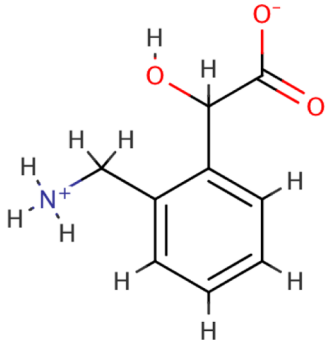


Figure 12. Isomer M2.

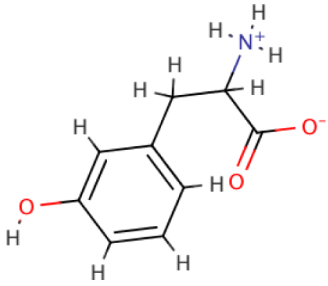


Figure 13. Isomer M4.

Table 6 shows the step count and Tanimoto distance for each isomer pair.

We separately sorted the step count distances and the Tanimoto distances and have shown the correspondence using

dx.doi.org/10.1021/ci4005173 | J. Chem. Inf. Model. 2014, 54, 57−68

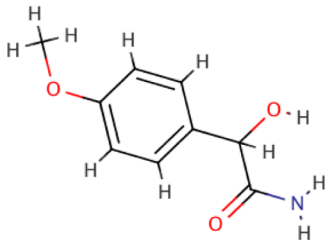


Figure 14. Isomer M42.

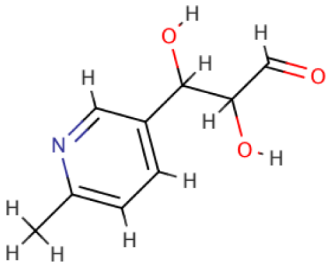


Figure 15. Isomer M168.

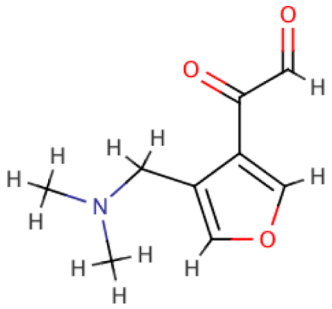
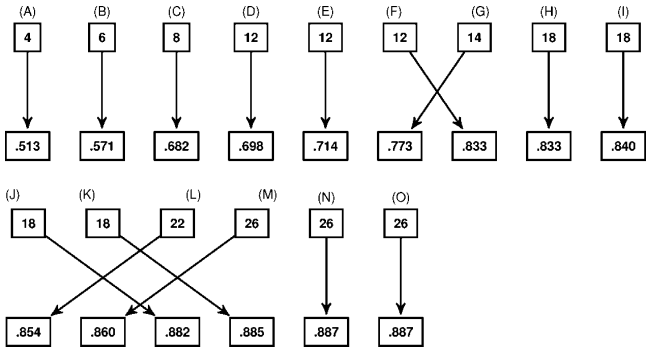


Figure 16. Isomer M8758.

edges in the following. It is easy to see that the step count distances and Tanimoto distances are very well-correlated.



Comparison of Algorithms. Table 7 shows the run time results for completely mapping 20 nicotine isomers without small rings using our new simultaneous all−all algorithm and the previous 2-CCV NR FDN and 2-CCV FDN MS algorithms.[5−7] Notice that simultaneously mapping all isomers provides a significant speed improvement over mapping each pair individually. In this case, it is approximately 15 times faster than 2-CCV FDN MS.

Nicotine without Small Rings. Consider the two nicotine isomers without small rings shown in Figure 17. Note that the bonds are only labeled to clarify our discussion of the results from our algorithm. We can use the algorithms developed in this paper

Table 6. Key

letter	isomers	step count distance	Tanimoto distance
A	(M1, M2)	4	0.513
B	(M1, M4)	6	0.571
C	(M1, M42)	8	0.682
D	(M1, M168)	12	0.698
E	(M1, M8758)	12	0.714
F	(M2, M4)	12	0.833
G	(M2, M42)	14	0.773
H	(M2, M168)	18	0.833
I	(M2, M8758)	18	0.840
J	(M4, M42)	18	0.854
K	(M4, M168)	18	0.860
L	(M4, M8758)	22	0.882
M	(M42, M168)	26	0.885
N	(M42, M8758)	26	0.887
O	(M168, M8758)	26	0.887

Table 7. Simultaneous All−All Algorithm Runtime vs 2-CCV NR FDN and 2-CCV FDN MS Run Times for 20 Isomers from the Nicotine Isomers without Small Rings Data Set

	time	
algorithm	s	h
2-CCV NR FDN[6]	314240.68	87.29
2-CCV FDN MS[6]	156750.73	43.54
all−all (this paper)	10324.16	2.87

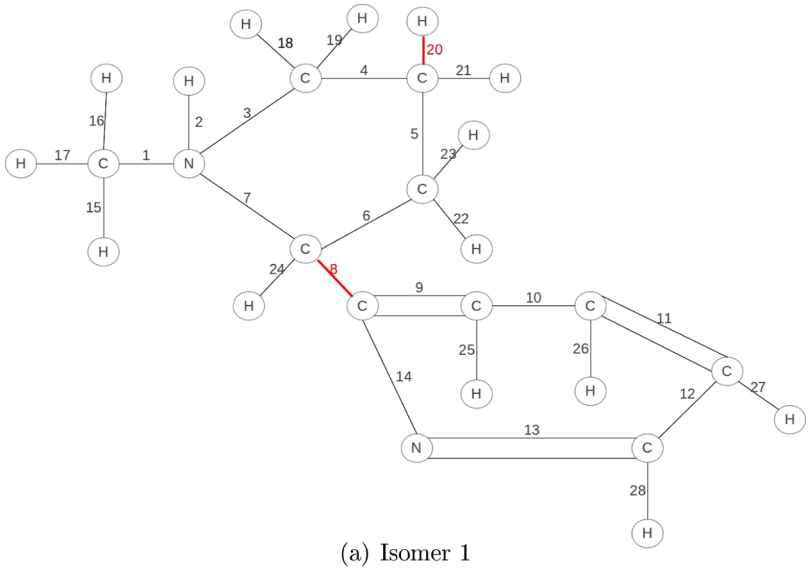
to determine how bonds must break and form to transform isomer 1 into isomer 2. The results of running our algorithms on the set of nicotine isomers without small rings shows that a mapping exists by breaking the bonds labeled 8 and 20 in isomer 1 and by breaking the bonds labeled 8 and 18 in isomer 2 (i.e., removing these bonds will result in two isomorphic graphs). This means that we can transform isomer 1 into isomer 2 by breaking both a C−C bond and C−H bond and then by forming a C−C bond and a C−H bond. This implies that BCD(isomer 1, isomer 2) = 4.

Table 8 shows the results of breaking up to five bonds per isomer for the nicotine isomers without small rings resulting in an order-10 isomer social network. The total size of the partial results stored on disk is 18.8 GB after breaking five bonds per isomer. The social network contains 603 037 of the $\binom{2934}{2}$ = 4 302 711 edges (about 14%) in the complete isomer network. Recall that we expect many vertex names to be repeated when generating DN vertex names for each isomer. The last column of Table 8 shows that although we are naming 27 vertices for each of the 2934 isomers for each iteration of the algorithm, there are only 474 unique vertex names which are generated.

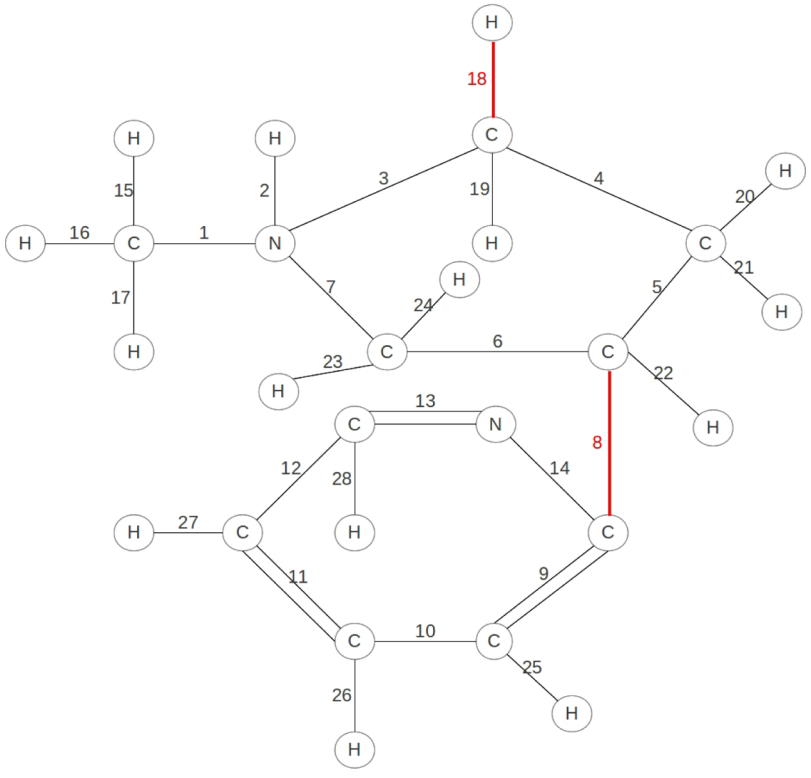
Nicotine with Small Rings. Table 9 shows the results of breaking up to four bonds per isomer for the nicotine isomers with small rings resulting in an order-8 isomer social network. The total size of the partial results stored on disk is 11.6 GB after breaking four bonds per isomer. Here, the number of edges in the isomer social network is about 4.5% of the $\binom{10\,000}{2}$ = 49 995 000 edges in the complete network. The last column of Table 9 shows that although we are naming 27 vertices for each of the 10 000 isomers for each iteration of the algorithm, there are only 430 unique vertex names which are generated.

Phenmetrazine. Table 10 shows the results of breaking up to four bonds per isomer for the phenmetrazine isomers (i.e., we get an order-8 social network of isomers). The total size of the partial

dx.doi.org/10.1021/ci4005173 | J. Chem. Inf. Model. 2014, 54, 57−68



(a) Isomer 1



(b) Isomer 2

Figure 17. Nicotine isomers without small rings.

Table 8. Nicotine Isomers without Small Rings Results Summary

no. bonds broken/isomer	no. of mappings remaining	no. of mappings found	% of mappings found	time (s)	no. of different vertex names
0	4302711	0	0%	1.218	26
1	4302707	4	0.000093%	27.32	97
2	4275260	27451	0.64%	440.207	201
3	4215919	86792	2.02%	4835.36	322
4	4015545	287166	6.67%	36635.07	412
5	3699674	603037	14.02%	211158.03	474

Table 9. Nicotine Isomers with Small Rings Results Summary

no. of bonds broken/isomer	no. of mappings remaining	no. of mappings found	% of mappings found	time (s)	no. of different vertex names
0	49995000	0	0%	4.20	26
1	49994992	8	0.000016%	102.70	103
2	49870216	134741	0.24%	3971.04	218
3	49440804	554196	1.11%	45860.32	343
4	47645983	2349017	4.70%	364265.09	430

dx.doi.org/10.1021/ci4005173 | J. Chem. Inf. Model. 2014, 54, 57−68

Table 10. Phenmetrazine Isomers Results Summary

no. of bonds broken/ isomer	no. of mappings remaining	no. of mappings found	% of mappings found	time (s)	no. of different vertex names
0	49995000	0	0%	1.98	20
1	49995000	0	0%	61.11	83
2	49835184	159816	0.32%	2719.57	176
3	49188028	806972	1.61%	30287.20	298
4	46205903	3789097	7.58%	237357.09	394

results stored on disk is 11.3 GB after breaking four bonds per isomer. We are able to find 7.58% percent of the $\binom{10\,000}{2}$ = 49 995 000 possible mappings. The last column of Table 10 shows that although we are naming 29 vertices for each of the 10 000 isomers for each iteration of the algorithm, there are only 394 unique vertex names which are generated.

Tyrosine. Finally, Table 11 shows the results of breaking up to four bonds per isomer for the tyrosine isomers (again, an order-8 social network). The total size of the partial results stored on disk is 5.0 GB. The social network represents about 3% percent of the $\binom{10\,000}{2}$ = 49 995 000 possible mappings. The last column of Table 11 shows that although we are naming 24 vertices for each of the 10 000 isomers for each iteration of the algorithm, there are only 682 unique vertex names which are generated.

Table 11. Tyrosine Isomers Results Summary

no. of bonds broken/isomer	no. of mappings remaining	no. of mappings found	% of mappings found	time (s)	no. of different vertex names
0	49995000	0	0%	2.93	57
1	49994993	7	0.000014%	43.40	206
2	49903433	91567	0.18%	965.80	398
3	49615670	379330	0.76%	10334.02	577
4	48506906	1488094	2.98%	77423.44	682

CONCLUSION

In conclusion, this paper defined complete isomer networks and order-*k* isomer networks We described a naive algorithm based on ARM and then presented a significantly faster algorithm that simultaneously computes bond count distances between a set of isomers. Experimental data documents the algorithm's performance on several benchmarks. Keeping in mind that our benchmarks are 1 or 2 orders of magnitude smaller than the complete isomer sets, faster algorithms for computing isomer networks are needed.

We have also shown both statistically and visually that there is a high correlation between the isomer step count similarity and isomer ECFP similarity (using Tanimoto distances). Since ECFP similarity is a relatively well-accepted measure of biological activity, isomer step counts also relate to the drug discovery question of estimating if two compounds have similar biological activity. Although the correlations are high in general, there will be a few pairs of isomers that have a large ECFP distance but a small isomer step count distance and vice versa. These will both be interesting cases to study. The former case would mean that a small change in bonds resulted in a large change in ECFP properties, while the latter case will mean that two isomers with a large step count distance had similar ECFP properties. Finally, the social networks generated by our algorithm also be analyzed using existing analytical tools for networks such as igraph.[4] We hope to report on the results of this analysis in the near future.

AUTHOR INFORMATION

Corresponding Author

*E-mail: tkouri@usf.edu.

Notes

The authors declare no competing financial interest.

ACKNOWLEDGMENTS

Research of the authors was funded in part by the National Science Foundation under Grant No. CNS-0931748. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the National Science Foundation.

REFERENCES

(1) Meringer, M. In *Handbook of Chemoinformatics Algorithms*; Faulon, J.-L., Bender, A., Eds.; CRC Press, 2010; Chapter 8, pp 233−268.

(2) Faulon, J.-L.; Visco, J. D. P.; Roe, D. Enumerating Molecules. *Reviews in Computational Chemistry*; Wiley, 2004.

(3) Zumdahl, S. S.; Zumdahl, S. A. *Chemistry*, 6th ed.; Houghton Miflin Company: Boston, MA, 2003.

(4) Csárdi, G.; Nepusz, T. The igraph software package for complex network research. *InterJournal Complex Syst.* **2006**, No. 1695.

(5) Crabtree, J.; Mehta, D. Automated reaction mapping. *J. Exp. Algorithmics* **2009**, *13*, 15:1.15−15:1.29.

(6) Kouri, T.; Mehta, D. Improved Automated Reaction Mapping. *Exp. Algorithms* **2011**, 157−168.

(7) Kouri, T.; Mehta, D. Faster Reaction Mapping through Improved Naming Techniques. *J. Exp. Algorithmics*, submitted for publication.

(8) Pemmaraju, S.; Skiena, S. *Computational Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*; Cambridge University Press: New York, 2003.

(9) Hopcroft, J. E.; Tarjan, R. E. A V log V Algorithm for Isomorphism of Triconnected Planar Graphs. *J. Comput. Syst. Sci.* **1973**, *7*, 323−331.

(10) Hopcroft, J. E.; Wang, J. K. Linear time algorithm for isomorphism of planar graphs (Preliminary Report). *Proceedings of the sixth annual ACM symposium on Theory of computing*, New York, Apr 30−May 2, 1974; pp 172−184.

(11) Garey, M. R.; Johnson, D. S. *Computers and Intractability; A Guide to the Theory of NP-Completeness*; W.H. Freeman & Co.: New York, 1990.

(12) Aho, A. V.; Hopcroft, J. E.; Ullman, J. D. *The Design and Analysis of Computer Algorithms*; Addison-Wesley: Reading, MA, 1974.

(13) Babai, L.; Luks, E. Canonical labeling of graphs. *Proceedings of the −fteenth annual ACM symposium on Theory of computing*; New York, Apr 25−27, 1983; pp 171−183.

(14) Luks, E. M. Isomorphism of Graphs of Bounded Valence Can Be Tested in Polynomial time. *J. Comput. Syst. Sci.* **1982**, *25*, 42−65.

(15) Faulon, J.-L.; Collins, M.; Carr, R. The Signature Molecular Descriptor. 4. Canonizing Molecules Using Extended Valence Sequences. *J. Chem. Inf. Model.* **2004**, *44*, 427−436.

(16) Faulon, J.-L. Isomorphism, Automorphism Partitioning, and Canonical Labeling Can Be Solved in Polynomial-Time for Molecular Graphs. *J. Chem. Inf. Comput. Sci.* **1998**, *38*, 432−444.

(17) Morgan, H. L. The Generation of a Unique Machine Description for Chemical Structures-A Technique Developed at Chemical Abstracts Service. *J. Chem. Doc.* **1965**, *5*, 107−113.

(18) Gasteiger, J.; Engel, T., Eds. *Chemoinformatics: A Textbook*; Wiley, 2003.

(19) McKay, B. No automorphisms, yes? http://cs.anu.edu.au/~bdm/nauty/ (accessed November 2013).

(20) McKay, B. Practical Graph Isomorphism. *Congr. Numer.* **1981**, *30*, 45−87.

(21) Miyazaki, T. The Complexity of McKay's Canonical Labeling Algorithm. *Groups and Computation II, DIMACS Ser. Discret. M.* **1997**, *28*, 239−256.
(22) Junttila, T.; Kaski, P. Engineering an efficient canonical labeling tool for large and sparse graphs. *Proceedings of the Ninth Workshop on Algorithm Engineering and Experiments and the Fourth Workshop on Analytic Algorithms and Combinatorics*, New Orleans, LA, January 2007; pp 135−149.
(23) Weininger, D. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *J. Chem. Inf. Comp. Sci.* **1988**, *28*, 31−36.
(24) Weininger, D.; Weininger, A.; Weininger, J. L. SMILES. 2. Algorithm for generation of unique SMILES notation. *J. Chem. Inf. Comp. Sci.* **1989**, *29*, 97−101.
(25) Weininger, D. SMILES. 3. DEPICT. Graphical depiction of chemical structures. *J. Chem. Inf. Comp. Sci.* **1990**, *30*, 237−243.
(26) Neglur, G.; Grossman, R.; Liu, B. Assigning Unique Keys to Chemical Compounds for Data Integration: Some Interesting Counter Examples. *Data Integration Life Sciences.* **2005**, 145−157.
(27) Babai, L.; Erdös, P.; Selkow, S. Random Graph Isomorphism. *Siam J. Comput.* **1980**, *9*, 628−635.
(28) Czajka, T.; Pandurangan, G. Improved Random Graph Isomorphism. *J. Discrete Algorithms* **2008**, *6*, 85−92.
(29) Reymond, J.-L.; van Deursen, R.; Blum, L. C.; Ruddigkeit, L. Chemical space as a source for new drugs. *Med. Chem. Commun.* **2010**, *1*, 30−38.
(30) Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; Stein, C. *Introduction to Algorithms*, 2nd ed.; The MIT Press/McGraw-Hill Book Company: 2004.
(31) Gomory, R. E.; Hu, T. C. Multi-Terminal Network Flows. *J. Soc. Ind. Appl. Math.* **1961**, *9*, 551−570.
(32) Knurh, D. E. *The Art of Computer Programming, Vol. 3: Sorting and Searching*, 2nd ed.; Addison-Wesley, 2003.
(33) Horowitz, E.; Sahni, S.; Mehta, D. P. *Fundamentals of Data Structures in C++*, 2nd ed.; Silicon Press: Summit, NJ, 2007.
(34) Blum, L. C.; Reymond, J.-L. *J. Am. Chem. Soc.* **2009**, *131*, 8732−8733.
(35) Reymond, J.-L.; Blum, L. C.; van Deursen, R. *Chimia* **2011**, *65*, 863−867.
(36) Nguyen, K. T.; Blum, L. C.; van Deursen, R.; Reymond, J.-L. *Med. Chem.* **2009**, *4*, 1803−1805.
(37) van Deursen, R.; Blum, L. C.; Reymond, J.-L. *J. Chem. Inf. Model.* **2010**, *50*, 1924−1934.
(38) Awale, M.; van Deursen, R.; Reymond, J.-L. *J. Chem. Inf. Model.* **2013**, *53*, 509−518.
(39) Blum, L. C.; van Deursen, R.; Reymond, J.-L. *J. Comput. Aid. Mol. Des.* **2011**, *25*, 637−647.
(40) Reymond, J.-L.; Awale, M. *ACS Chem. Neurosci.* **2012**, *3*, 649−657.
(41) Rogers, D.; Hahn, M. *J. Chem. Inf. Model.* **2010**, *50*, 742−754.
(42) Navidi, W., Ed. *Statistics for Engineers and Scientists*; McGraw Hill, 2006.
(43) Schneider, J. W.; Borlund, P. *J. Am. Soc. Inf. Sci. Tec.* **2007**, *58*, 1−10.
(44) Caporaso, J. G.; et al. *Nat. Methods* **2010**, 335−336.
(45) ChemAxon, Jchem. http://www.chemaxon.com/jchem/intro/index.html (accessed November 2013).