JOURNAL OF CHEMICAL INFORMATION AND MODELING

Article

pubs.acs.org/jcim

Efficient Heuristics for Maximum Common Substructure Search

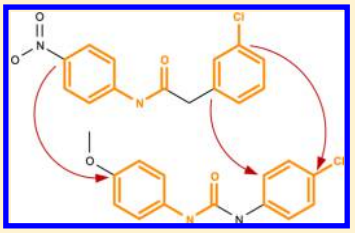
Péter Englert*,† and Péter Kovács*,‡

†Department of Algorithms and Applications, Eötvös Loránd University, Budapest 1117, Hungary

‡ChemAxon Ltd., Budapest 1031, Hungary

Supporting Information

ABSTRACT: Maximum common substructure search is a computationally hard optimization problem with diverse applications in the field of cheminformatics, including similarity search, lead optimization, molecule alignment, and clustering. Most of these applications have strict constraints on running time, so heuristic methods are often preferred. However, the development of an algorithm that is both fast enough and accurate enough for most practical purposes is still a challenge. Moreover, in some applications, the quality of a common substructure depends not only on its size but also on various topological features of the one-to-one atom correspondence it defines. Two state-of-the-art heuristic algorithms for finding maximum common substructures have been implemented at ChemAxon Ltd., and effective heuristics have been developed to improve both their efficiency and the relevance of the atom mappings they provide. The implementations have been thoroughly evaluated and compared with existing solutions (KCOMBU and Indigo). The heuristics have been found to greatly improve the performance and applicability of the algorithms. The purpose of this paper is to introduce the applied methods and present the experimental results.



INTRODUCTION

One of the greatest challenges of cheminformatics is the demand for efficient algorithms for a variety of computationally hard problems. A particular example is the *maximum common substructure* (MCS) problem, which is usually formulated as finding the maximum common subgraph of two molecular graphs.

MCS search plays an important role in a wide range of applications in the field of computational chemistry and, increasingly, computational biology.[1] A typical application is similarity search, which is essential in the early stages of the drug discovery process.[2−6] Intuitive measures of the similarity of two molecular structures can be defined on the basis of the size of their MCS,[7−9] which is also supported by the similar property principle.[10,11] Other important applications of MCS search include clustering,[12−14] identifying matched molecular pairs,[15−17] automated reaction mapping,[18,19] and molecular alignment.[20,21]

The MCS problem is, however, a computationally complex problem, and its applications impose strict and often conflicting requirements. First of all, the solution method should be rather fast, which is why heuristic algorithms are often employed. On the other hand, the results should be equal or at least very close to the actual maximum. Moreover, features that make sense to a chemist but are hard to grasp formally are sometimes desired, such as not breaking rings or functional groups or mapping parts of the input molecules to each other while taking certain topological features into account.

At ChemAxon Ltd., we have developed and implemented efficient MCS algorithms to satisfy these requirements. We have taken various methods and heuristics from the available literature, thoroughly evaluated their performance, and improved upon them with novel ideas. This paper details our results, the methods that have been found to be most effective, and our improvements. Although we focus on molecular graphs, the resulting algorithms are also applicable to other fields outside cheminformatics where the MCS problem occurs, such as computer vision and pattern recognition.[22,23]

Definitions and Terminology. The typical representations of molecules are by means of 2D and 3D molecular graphs.[5,24] In this paper, we focus solely on 2D graphs, as their usage is significantly more common. Thus, we define *molecular graphs* as simple, undirected, labeled graphs in which the nodes represent the atoms, the edges represent the chemical bonds, and the labels of the nodes and edges represent the atom and bond types, respectively. We consider only heavy atoms; hydrogens are not represented in molecular graphs.

A molecular graph G' is a *subgraph* of another molecular graph G, denoted as $G' \subseteq G$, if there exists an injective mapping f of the atoms of G' to atoms of G with the following properties: each atom maps to an atom of the same type, and if two atoms u' and v' in G' have a bond connecting them, $f(u')$ and $f(v')$ in G also have a bond of the same type connecting them. A subgraph G' of G is called an *induced subgraph* if for each pair of atoms u' and v' in G', they are adjacent if and only if $f(u')$ and $f(v')$ in G are adjacent. Deciding whether a graph is a subgraph of another graph is a classic NP-complete problem[25] with countless applications in the field of chemistry.[5,26]

This paper considers the MCS problem, which is also called the *maximum overlapping set* (MOS) problem in the cheminformatics literature. This problem is formulated either as finding a *maximum common induced subgraph* (MCIS) or as

Received: January 21, 2015
Published: April 11, 2015

DOI: 10.1021/acs.jcim.5b00036
J. Chem. Inf. Model. 2015, 55, 941−955

finding a *maximum common edge subgraph* (MCES) of two molecular graphs.

A molecular graph G' is a *common subgraph* of two molecular graphs G_1 and G_2 if $G' \subseteq G_1$ and $G' \subseteq G_2$. A common subgraph G' of G_1 and G_2 is a *common induced subgraph* if it is an induced subgraph of both G_1 and G_2. The MCIS problem is to find a common induced subgraph of two graphs containing as many *nodes* as possible. In contrast, the MCES of two molecular graphs is defined as a common subgraph (not necessarily induced) containing as many *edges* as possible.

Both the MCIS and MCES problems are inherently complex. It is easy to see that they are NP-hard (or, if stated as decision problems, NP-complete),[25] since they are at least as general as deciding whether a graph is an (induced) subgraph of another graph.

It can be argued that the MCES of two molecular graphs is more relevant in the field of cheminformatics, as it is closer to the notion of chemical similarity between molecules than MCIS.[18,27] Figure 1 illustrates the difference. Still, many state-of-the-art MCS algorithms solve the MCIS problem.[28,29] Luckily, there exists a simple transformation that allows us to use these algorithms to solve the MCES problem as well. This transformation will be discussed later on.

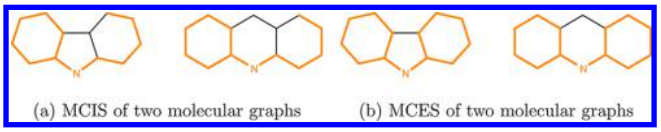


Figure 1. Example of the difference between the MCIS and MCES of two molecular graphs.

Further distinction can be made between *connected* and *disconnected* MCIS or MCES problems. In the connected case, a common subgraph must be composed of a single component, while in the disconnected case, it may consist of an arbitrary number of connected components. Figure 2 shows an example.

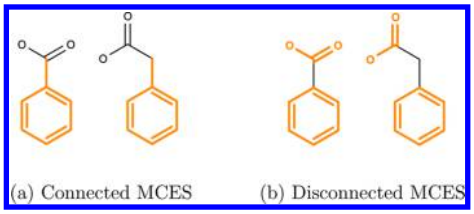


Figure 2. Example of the difference between the connected and disconnected MCESs of two molecular graphs.

In many cases, the ability to handle disconnected common subgraphs is desired, but most of the available algorithms can only find connected ones. Intermediate approaches have also been developed applying constraints on the number of components or on their topological relationships.[29] Another interesting approach is to find a connected MCS tolerating a few mismatches in atom and bond types.[30]

In the rest of this paper, MCS is used as a synonym for the MCES of two molecular graphs, a *common substructure* is taken to mean a common subgraph (not necessarily induced), and the size of the result refers to the number of edges in a common subgraph. Unless stated otherwise, we allow the common subgraphs to be disconnected. The two input molecular graphs are called the *query* and the *target* for the sake of simplicity, although they usually play the same role in MCS search (unless special query features are considered, as discussed later).

A common substructure of a query and a target is also referred to as a *mapping* (one-to-one correspondence) between their bonds. The reason for this is that the algorithms usually keep track of a common substructure as a selection of bonds from the query and an associated target bond for each query bond in the selection. The associated bond represents the same bond in the common substructure. This is essentially a mapping, a partial function from the bonds of the query to the bonds of the target. Adding a new bond to the common substructure can also be described as adding a query−target bond pair to the mapping. (In the case of the MCIS problem, the mapping is composed of atom pairs instead of bond pairs.)

An important observation is that multiple mappings may define equivalent common substructures. We will elaborate upon the importance of the found mapping and how it has essential features apart from the common substructure it defines.

Finding the MCS of more than two graphs, often called the *multi-MCS problem*, is also important in several applications. Although it can be approximated using consecutive MCS computations between pairs of graphs, more efficient methods are often desirable,[31,32] but this is not in the scope of the present paper.

METHODS

Algorithms. There are many published algorithms for MCS search. They can be categorized according to multiple aspects, for example, whether the algorithm is exact or heuristic, whether it solves the MCIS or the MCES variant of the problem, or whether it finds connected or disconnected MCSs. The applied algorithmic approaches also show a wide variety, including backtracking,[28,33−35] reducing the problem to finding a maximum clique of a derived graph,[8,36−38] genetic algorithms,[39,40] and greedy heuristics.[29] Some methods also utilize the concept of reduced graphs.[13,41−43] A remarkable survey of MCS algorithms was presented by Raymond and Willett,[27] although new methods have also been devised since then.[28,29]

Two algorithms have been implemented at ChemAxon, one based on the maximum clique approach and the other on a recently published algorithm called the build-up method.[29] In what follows, we briefly summarize these two approaches.

Clique-Based Algorithms. One of the most popular algorithmic approaches is the reduction of the MCS problem to the maximum clique problem. Several state-of-the-art algorithms, both exact and heuristic, use this approach[8,36−38] because of the extensive literature and efficient algorithms available for clique detection.

The original variant of this method solves the MCIS problem. It involves constructing a *modular product graph* from the two molecules, which represents the compatibility of their atom pairs (hence it is also known as the *compatibility graph*). Let $G_1 \times G_2$ denote the modular product graph of two molecular graphs G_1 and G_2. A node in $G_1 \times G_2$ corresponds to each pair of atoms (u_1, u_2), where u_1 is an atom of G_1, u_2 is an atom of G_2, and they are of the same type. Two distinct nodes (u_1, u_2) and (v_1, v_2) of $G_1 \times G_2$ are connected by an edge if and only if they are compatible, i.e., a mapping of a common induced subgraph can contain both atom pairs at the same time. This means that $u_1 \neq v_1$, $u_2 \neq v_2$, and either u_1 is not adjacent to v_1 and u_2 is not adjacent to v_2 or both are adjacent

DOI: 10.1021/acs.jcim.5b00036
J. Chem. Inf. Model. 2015, 55, 941−955

and the connecting bonds are of the same type. Consequently, a one-to-one correspondence exists between the *cliques* (complete subgraphs) of $G_1 \times G_2$ and the atom mappings that define common induced subgraphs of G_1 and G_2.[44] The size of a clique is obviously equal to the atom count of the corresponding common induced subgraph. Therefore, finding an (approximate) MCIS of G_1 and G_2 can be reduced to finding an (approximate) maximum clique in $G_1 \times G_2$.[8,37,44]

As we are interested in the MCES of molecular structures instead of the MCIS, the above approach is applied to the *line graphs* of the original graphs. The line graph $L(G)$ of a molecular graph G represents the adjacencies between the bonds of G. The nodes of $L(G)$ are associated with the bonds of G, and two nodes are adjacent in $L(G)$ if and only if the corresponding bonds are adjacent in G. For two molecular graphs G_1 and G_2, every common subgraph (not necessarily induced) that does not have isolated atoms corresponds to a common induced subgraph of the line graphs $L(G_1)$ and $L(G_2)$. Conversely, it has been proved that a common induced subgraph of $L(G_1)$ and $L(G_2)$ has a corresponding common subgraph of G_1 and G_2 (without isolated atoms) unless a so-called ΔY *exchange* occurs,[45] which is discussed in the next subsection. Obviously, the bond count of the common subgraph of G_1 and G_2 is equal to the node count of the corresponding common induced subgraph of $L(G_1)$ and $L(G_2)$. Furthermore, we can entirely ignore common subgraphs having isolated atoms, since the bond count is to be maximized in the case of MCES search. Consequently, finding an (approximate) MCES of two molecular graphs G_1 and G_2 can be reduced to finding an (approximate) MCIS of $L(G_1)$ and $L(G_2)$, i.e., finding an (approximate) maximum clique in $L(G_1) \times L(G_2)$, provided that ΔY exchange does not occur.[8,36−38]

Figure 3 shows an example of how the modular product graph is built from two molecular graphs when solving the

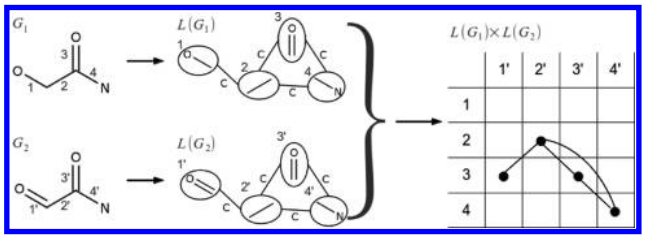


Figure 3. Example of building the modular product graph for clique-based MCES search algorithms.

MCES problem (the example is taken from refs 8 and 38). G_1 and G_2 are first transformed into their line graphs $L(G_1)$ and $L(G_2)$. In the line graphs, the label of a node is the type of the corresponding bond along with the types of the atoms the bond connects. Furthermore, the label of an edge in the line graphs is the type of the common atom of the two adjacent bonds. The product graph, symbolized by the table on the right, contains a node for each pair of line-graph nodes such that one node is taken from $L(G_1)$ (a bond of G_1), the other node is taken from $L(G_2)$ (a bond of G_2), and their labels are the same. The resulting product graph contains two maximal cliques: the first is composed of nodes (3,1′) and (2,2′), while the second is composed of nodes (2,2′), (3,3′), and (4,4′). Thus, the maximum clique is the second one. This corresponds to the common substructure made up of edges 2, 3, and 4 in G_1 and edges 2′, 3′, and 4′ in G_2.

Our implementation of this approach is based on the heuristic maximum clique algorithm by Grosso et al.[46] with several modifications, most of which are detailed in the section on improvements.

ΔY Exchange. A ΔY exchange can occur if one of the input graphs contains a Δ subgraph (K_3) and the other contains a Y subgraph ($K_{1,3}$). These graphs have isomorphic line graphs but are not themselves isomorphic,[47,48] as shown in Figure 4, so an

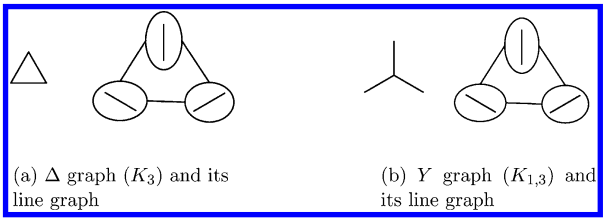


Figure 4. ΔY exchange.

algorithm that operates on line graphs may erroneously match these different subgraphs. This problem becomes apparent only when projecting the mapping of the line graphs onto the original graphs, since it is impossible to create a consistent node−node mapping from an edge−edge mapping if a ΔY exchange has occurred.

ΔY exchange is rare when molecular graphs are concerned, as they rarely contain a Δ subgraph. For reference, in the PubChem Compound Database,[49,50] only about 4% of the structures contain a Δ subgraph.[51]

The usual way of handling this phenomenon is to continuously check whether a ΔY exchange has occurred during search (as in refs 8 and 36). If an exchange is discovered, the algorithm can eliminate it, for example, by selecting a different bond pair instead of the one that caused the ΔY exchange to occur. In this case, the method must keep track of the atom−atom mapping along with the bond−bond mapping during search. This approach is easier to implement in some algorithms and harder in others, but the performance usually suffers.

Our implementations apply another approach. ΔY exchanges are ignored during search, but afterward, when the atom−atom mapping is calculated, a bond is removed from the common subgraph for each ΔY exchange that has occurred to ensure a consistent mapping. This is simple to implement and has negligible effect on performance. However, it may produce suboptimal results even on some small inputs, and hence, it cannot be used in an exact MCS algorithm. An example can be seen in Figure 5. An algorithm using this approach would find no difference in size between the results shown in Figure 5a and Figure 5b, so it may return the one containing a ΔY exchange. In this case, a bond, for example the one colored red, must be removed afterward, and the final result is not optimal.

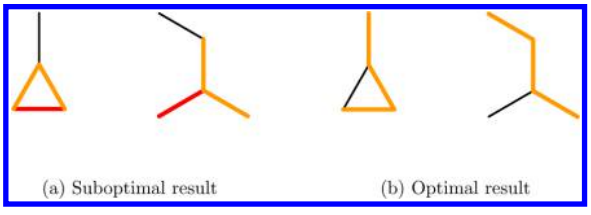


Figure 5. Example demonstrating the disadvantage of detecting ΔY exchange after MCS search. The removed bond is colored red.

DOI: 10.1021/acs.jcim.5b00036
J. Chem. Inf. Model. 2015, 55, 941−955

Build-Up Algorithm. A recent paper by Kawabata[29] describes a greedy heuristic algorithm called the build-up method. The notion of topologically constrained MCS and the topological heuristic scores described along with the algorithm are of special interest, since they have the potential to yield results that are chemically more relevant.

The build-up algorithm was developed to solve the MCIS problem. It maintains an ordered list of common substructures, represented by atom mappings, which are gradually extended. The algorithm begins with trivial mappings having only one atom pair. These mappings are sorted by a scoring function, and only the first K mappings are kept. In the next step, each mapping is extended with each feasible atom pair, and the K best extended mappings are selected again according to the scoring function. This method is iterated until none of the maintained mappings can be extended.

The heuristic scoring function consists of multiple components. The neighbor score measures, for each pair of atoms in the mapping, the dissimilarity of their immediate neighbor atoms in the two molecules. Another score, originally introduced by Morgan,[52] is defined as the difference in the extended connectivity values of the matched atoms. Furthermore, when a topologically constrained MCS is to be found, Kawabata also suggests an additional score comparing the topological distances of the matched atoms in the two chemical graphs. In this study, however, we consider the original MCS problem, so this score is discarded.

The build-up algorithm evaluates each atom mapping with the sum of the scores. A mapping is considered redundant if the current list already contains another mapping that includes the same number of atoms for each atom type and has the same scores.

As noted in the previous section, an MCIS algorithm can be adapted to the MCES problem. We have implemented such a modified version of the build-up algorithm, applying $K = 5$. While it initially underperformed the clique-based implementations, after being augmented with heuristics described later, this implementation turned out to be a viable alternative. It is able to compete with other approaches in terms of both running time and accuracy.

Upper Bound Calculation. An important aspect of MCS search is the existence of various upper bound calculation methods.[27] These methods are generally much faster than MCS algorithms, even heuristic ones. Therefore, they can serve as effective tools for screening molecule pairs and filtering out those that are guaranteed to have a small MCS. This is very useful, for example, to accelerate the search for molecules with large common substructures during similarity search, clustering, or searching for matched molecular pairs.

We implemented a two-step screening procedure similar to the one described by Raymond et al.[8] It consists of two upper bound calculation algorithms, the second of which is significantly slower but more accurate than the first. The first method is run on a pair of molecules, and if the similarity of the two molecules, based on the resulting upper bound on the size of their MCS, is larger than a given threshold, then the second method is also run. If the resulting similarity score is still larger than the threshold, only then is the actual MCS algorithm run; otherwise, the pair is discarded.

Improvements. This section details the methods that helped us improve upon the original algorithms in regard to accuracy, running time, and memory usage. Most of the available literature concentrates on pruning heuristics or reducing the search space, and remarkable methods have been developed.[8,38] However, it is reasonable to assume that heuristic algorithms benefit less from such methods, and this benefit may not be enough to compensate for their computational cost. We concentrate more on making the algorithms faster and better at guessing which part of the search space to explore.

Representation. One of the main improvements concerns the representation of the modular product graph in clique-based algorithms. The size and density of this graph are among the main disadvantages of this approach, as they impact both memory usage and running time. However, we argue that this graph is so dense that the complement graph should be stored instead, as an adjacency list. The same idea was used in ref 31 to improve the efficiency of the Bron−Kerbosch algorithm, which finds all maximal cliques.

We exploit certain properties of molecular graphs, namely, that they are very sparse and each atom has a strictly limited degree. In a random selection of 10^6 molecules from the PubChem Compound Database,[49,50] the maximum degree of an atom was found to be 6, the average degree was 2.15, and the average of the maximum degree for each molecule was 3.31 (hydrogen atoms are ignored).

Let d denote the maximum degree of an atom in the input molecular graphs G_1 and G_2. In the line graphs $L(G_1)$ and $L(G_2)$, the maximum degree of a node corresponding to a bond (u, v) is $2(d - 1)$, since both u and v may have $d - 1$ other bonds connected to them. Let the number of bonds in G_1 be m_1 and the number of bonds in G_2 be m_2. The modular product graph $L(G_1) \times L(G_2)$ has at most m_1m_2 nodes. If two distinct nodes (e_1, e_2) and (f_1, f_2) of the product graph (where e_1 and f_1 are bonds in G_1 and e_2 and f_2 are bonds in G_2) are *not* adjacent (i.e., incompatible), then either e_1 and f_1 or e_2 and f_2 are equal or adjacent in the corresponding graph. Therefore, the number of nodes that are not adjacent to (e_1, e_2) in the modular product graph is at most

$$\begin{aligned}&(\deg(e_1) + 1)m_2 + (\deg(e_2) + 1)m_1\\ &\quad \le (2(d - 1) + 1)m_2 + (2(d - 1) + 1)m_1\\ &\quad = (2d - 1)(m_1 + m_2)\\ &\quad = O(d(m_1 + m_2))\end{aligned}$$

Since d is a small constant (it is usually at most 4), the number of edges in the complement of the modular product graph is $O(m_1m_2(m_1 + m_2))$. In contrast, the number of edges the original product graph may contain is $O((m_1m_2)^2)$, which is why it is usually represented as an adjacency matrix. This is a reduction in memory usage from $O(m^4)$ to $O(m^3)$, where $m = \max(m_1, m_2)$, albeit with a larger constant.

This change can also significantly improve the running time. The algorithms we studied in depth often access the product graph in one way only. When adding or removing a node from the current clique, they traverse the nodes that are not adjacent to that node and update the internal data structures. This operation is made a lot faster with the new representation, as these nodes are explicitly stored in a list and it is not necessary to iterate over an entire row in an adjacency matrix.

Another way in which the new representation improves the running time is that the modular product graph can also be built in $O(m^3)$ time instead of $O(m^4)$ time. This is rather important because building the product graph is a dominant

DOI: 10.1021/acs.jcim.5b00036
J. Chem. Inf. Model. 2015, 55, 941−955

phase of the algorithm and often accounts for the majority of the total running time, especially in the case of large molecules.

Our benchmark tests verified the effectiveness of this improvement in terms of both memory usage and running time. In accordance with the above calculations, the experimental results clearly show an order of magnitude difference in memory requirements.

Early Termination. Early termination is a common heuristic in maximum clique search methods. It involves terminating the algorithm when it is guaranteed that an optimal solution has been found. In our case, such a guarantee can be provided by calculating an upper bound on the size of the MCS. We implemented this heuristic in our clique-based algorithm using the second upper bound calculation method described in ref 8 with satisfactory results. However, we did not add this heuristic to the build-up algorithm, as the amount of computation we may save is negligible.

Connectivity Heuristic. An important heuristic of MCS algorithms is to prefer atoms and bonds that are connected to the current common substructure when it is extended. This idea is applied in, for example, the backtracking algorithm of Cao et al.,[28] and it also turned out to substantially improve the accuracy and speed of both the clique-based and build-up algorithms as well. This heuristic is natural when looking for a connected MCS, but it greatly improves the results even in the disconnected case. It may also decrease the number of fragments (components) the found common substructures comprise, which is obviously beneficial.

The idea, when translated into terms closer to the implementations, is to prefer to add to the current mapping a bond pair (e_1, e_2) for which there is at least one bond pair (f_1, f_2) in the mapping such that e_1 and f_1 are adjacent. This also implies the adjacency of e_2 and f_2 because of the compatibility requirement of the bond pairs, so it does not need to be checked. This heuristic can easily be implemented in backtracking methods and in the build-up algorithm.

In clique-based algorithms, we must distinguish D-edges and C-edges in the modular product graph (as in ref 37) to implement the connectivity heuristic. C-edges connect nodes for which the corresponding bond pairs are adjacent in both the query and the target, while D-edges connect nodes for which the corresponding bond pairs are not adjacent. The heuristic can be implemented effectively by precomputing for each node the list of nodes that are connected to it by a C-edge. With the notations introduced in Representation, a node may have at most $(2(d-1))^2 = O(d^2)$ such neighbors, that is, each adjacent query bond paired with each adjacent target bond. (Recall that d is a small constant, typically at most 4, in the case of molecular graphs.)

An additional question is how to consider the number of C-edges connecting a candidate node to the current clique, which we call the *connectivity score*. Our experiments show that it is generally beneficial to use this heuristic aggressively, i.e., to choose the next node from the feasible nodes having the highest connectivity score. In certain cases, however, this heuristic may lead to suboptimal results, as exemplified by Figure 6. However, we have found it to be more effective to correct such suboptimal results afterward by trying to swap nonincluded nodes with included ones or using an iterated search.

ECFP-Based Heuristic. The main heuristic we developed to improve MCS algorithms is based on *extended connectivity fingerprints* (ECFPs),[53] which already have many applications in similarity search. On the basis of our measurements, the ECFP-based heuristic greatly enhances the accuracy of both the clique-based and build-up algorithms.

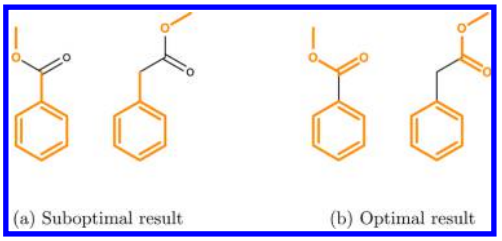


Figure 6. Example demonstrating that the greedy use of the connectivity heuristic may lead to suboptimal results.

The ECFP hash value of an atom in a molecule is an integer hash code of its environment with a given radius. The environment of radius r for an atom is taken to mean the subgraph induced by atoms at a distance of at most r from that atom. The ECFP hash of radius 0 represents the type of the atom. A similar hash value is assigned for each bond as well. Then the ECFP hash of radius $r + 1$ of a given atom is computed in the following way. For each adjacent atom, its ECFP hash of radius r is combined with the hash of the bond leading to it. These combined values are then hashed in an order-independent way to produce the ECFP hash of radius $r + 1$ of the considered atom. As a result, given two atoms, their environments of radius r can be equivalent only if their ECFP hash codes of radius r are the same.

The basic idea is to prefer matching atoms and bonds that have large equivalent environments. Figure 7 illustrates this concept. Similar molecules usually contain relatively large substructures that are equivalent, and this heuristic is very effective in finding and matching such substructures. We define the ECFP score of a pair of atoms u_1 and u_2, denoted as s_{u_1,u_2}, as the largest r such that for each $r' \leq r$, the ECFP hash values of radius r' for the two atoms are the same. MCES algorithms, however, operate on bonds instead of atoms, and thus, we need to combine the scores of atom pairs to obtain suitable scores for bond pairs. For a bond pair (e_1, e_2), where $e_1 = (u_1, v_1)$ is a bond of the query and $e_2 = (u_2, v_2)$ is a bond of the target, we define its ECFP score as $\max(\min(s_{u_1,u_2}, s_{v_1,v_2}), \min(s_{u_1,v_2}, s_{v_1,u_2}))$. Since there might be two ways to pair the atoms of a bond pair to each other, we use the score of the better one. However, to determine the score of a bond pair if the pairing of the atoms is fixed, we use the worse score, since we want to know the location of the closest difference in the bonds' environments.

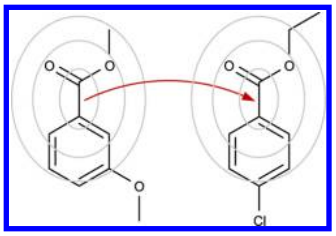


Figure 7. Example demonstrating the concept of the ECFP-based heuristic.

During search, we favor choosing bond pairs with higher ECFP scores. This often leads to finding a near-optimal solution early in the search. Of course, ECFP scores are not perfect, and as such, they may guide the search toward a local

DOI: 10.1021/acs.jcim.5b00036
J. Chem. Inf. Model. 2015, 55, 941−955

maximum. In our clique-based implementation, we try to avoid getting stuck in such a local maximum by lowering the scores of the currently chosen bond pairs between iterations.

Extensions. In this section, we discuss various methods that help improve the chemical relevance of the results and make the MCS algorithms much more usable in many applications. These methods are, to the authors' knowledge, novel ideas that have not been described before.

Mapping Optimization. The original MCS algorithms do not consider anything but the size of the common substructures as the measure of their quality. However, the topological features of the result, such as the orientation of the common substructure in the query and the target, are also important in certain applications, for instance, reaction mapping and molecule alignment. These aspects are rarely explored in depth by the available literature. Notable examples are the work of Kawabata,[29] which introduces the notion of topologically constrained MCS; the article of Stahl et al.,[54] where similarity scores are penalized on the basis of the difference in the arrangements of the largest fragments of the MCS; and the Small Molecule Subgraph Detector (SMSD) toolkit,[55] which ranks common substructures by multiple chemically relevant criteria. In this study, we developed a general method that is capable of handling typical problematic cases.

A straightforward method for finding an appropriate mapping would be to enumerate all possible mappings of the found common substructure. This could be achieved by running a substructure search algorithm to find all possible mappings between the common substructure G_c and the query G_1 and between G_c and the target G_2. Mappings between G_1 and G_2 are then generated by composing each pair of $G_c \rightarrow G_1$ and $G_c \rightarrow G_2$ mappings. However, this approach is not feasible in most cases. The main problem is the large number of mappings that might be produced: even simple query−target pairs can have an MCS with millions of different mappings.

A much better approach is based on the observation that in all of the examples we investigated, if a mapping was found to be inadequate, the problem was with atoms at the edge of the common substructure, i.e., matched atoms that are adjacent to nonmatched parts in the query or the target. The essence of our idea is to consider only mappings where these atoms are matched differently. This can be done in multiple ways, but an (approximate) MCS has to be found first. On the basis of this common substructure, we call an atom in the query *interesting* if it is part of the substructure but has an incident bond that is not part of the substructure. We define interesting atoms in the target analogously. Figure 8 shows an example. It should be noted that, if possible, it is usually preferred to match interesting atoms of the query to interesting atoms of the target.

By focusing on these interesting atoms, we can substantially reduce the number of mappings to be enumerated. Our clique-based algorithm applies this approach as follows. First, an approximate MCS is found as usual. With respect to this result, we call a bond interesting if it is part of the common substructure and one of its atoms is interesting. Then we try to add pairs of interesting bonds (i.e., nodes of the product graph), remove their fragments (so, for example, a whole chain can be reversed), and run a local search to increase the size of the common substructure again. If a larger substructure is found, or if the size is the same as before but the new mapping is to be preferred according to a general scoring function, then it is kept instead of the previous mapping. We try to add each interesting node of the product graph one by one in this way, and only the best mapping is returned. If more mappings are requested, they are enumerated using substructure search in both molecules.

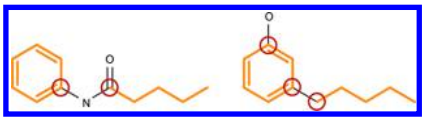


Figure 8. Interesting atoms for mapping optimization. The MCS is highlighted in orange, and the interesting atoms in both molecules are circled in red.

We experimented with different scoring functions and found the following one to be adequate in most use cases. We denote the set of query bonds as E_1 and the set of target bonds as E_2. A mapping of a common substructure is $M \subseteq E_1 \times E_2$ (actually a bijective function between subsets of E_1 and E_2). Let the functions $\deg_1$: $E_1 \rightarrow \mathbb{N}$ and $\deg_2$: $E_2 \rightarrow \mathbb{N}$ assign to each bond in the query and the target, respectively, the number of bonds adjacent to it. Furthermore, let dist_1: $E_1 \times E_1 \rightarrow \mathbb{N}$ and dist_2: $E_2 \times E_2 \rightarrow \mathbb{N}$ assign the distance to each bond pair in the query and the target, respectively. If two bonds are not connected with a path, their distance is defined to be a sufficiently large finite constant, for example, $\max(|E_1|, |E_2|)$. The score of a mapping M, denoted as s_M, is then defined as

$$s_M = \sum_{(e_1,e_2)\in M} |\deg_1(e_1) - \deg_2(e_2)| + \sum_{(e_1,e_2)\in M} \sum_{(f_1,f_2)\in M} (\mathrm{dist}_1(e_1,f_1) - \mathrm{dist}_2(e_2,f_2))^2$$

The lower this value is, the better we consider the mapping to be among mappings of the same size. The first part of the expression sums the differences in degree for each query−target bond pair. This allows us to align chains and rings so that adjacent parts that are not in the common substructure are also aligned. A good example is matching of two rings with different substituents, as shown in Figure 9. The second part of the score

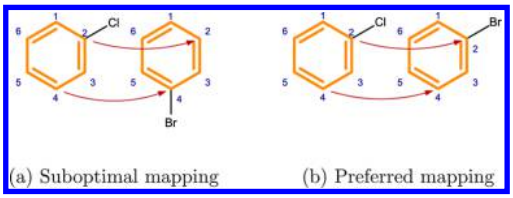


Figure 9. MCS with different mappings. A mapping that matches the carbon atoms that are bonded with halogen atoms is preferred.

is the sum of squares of the differences in distance in the query and in the target for each pair of bonds of the common substructure. This allows us to optimize the alignment of the fragments of the common substructure with respect to each other in the query and the target. Figure 10 illustrates the importance of this.

An important additional benefit of this mapping optimization procedure is that it can also yield a larger common substructure and thereby improve the accuracy of the algorithm. A simple example is depicted in Figure 11. As a result of its inherent inexactness, the first phase of the algorithm may find a solution that is suboptimal in size. However, when we search for better

DOI: 10.1021/acs.jcim.5b00036
J. Chem. Inf. Model. 2015, 55, 941−955

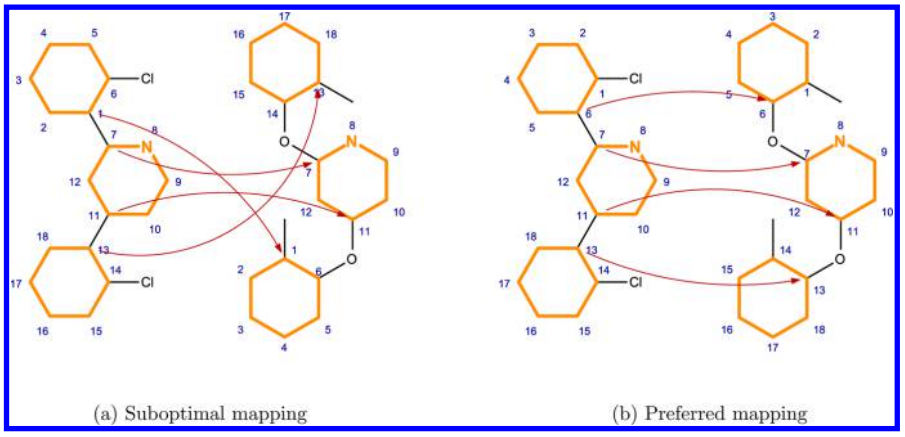


Figure 10. MCS with different mappings. The atoms at the ends of an unmatched chain between rings should be matched to the atoms at the ends of the chain connecting the corresponding rings in the other molecule. In general, we prefer mappings that match atoms that are close to each other in one molecule to atoms that are close to each other in the other molecule.

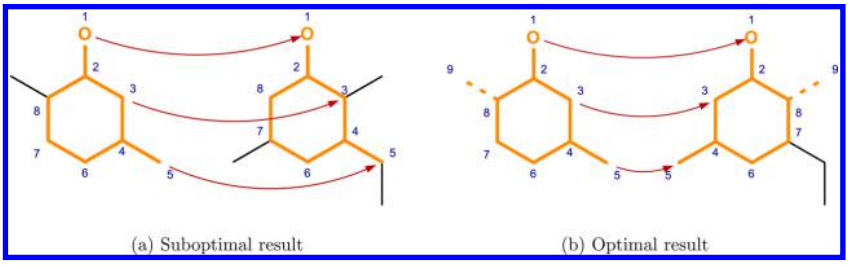


Figure 11. Example demonstrating that finding an appropriate mapping for a suboptimal solution can make it possible to extend the common substructure. The newly added bond is denoted by a dotted line in both molecules.

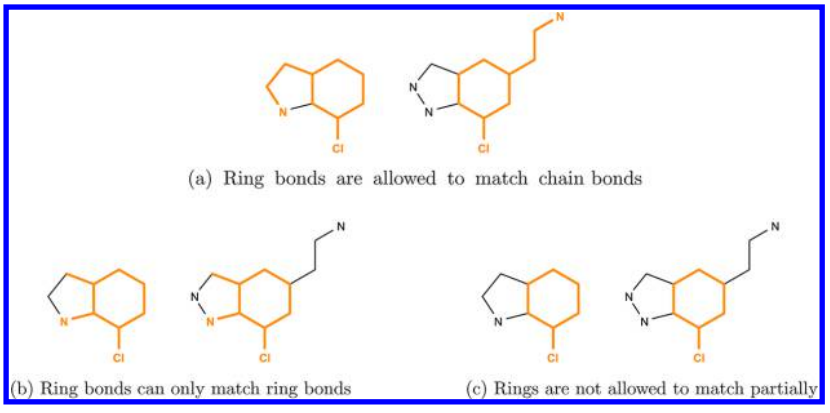


Figure 12. Examples showing the MCS of the same molecule pair with different ways of handling ring bonds.

mappings of this common substructure, we may find a mapping that can be extended to a larger one by local search.

Preserving Rings. In applications such as clustering or alignment, a frequent request is to consider only substructures where rings are not broken,[31] that is, if a bond in the query or the target molecule is part of a ring, it has to be part of a ring in the found common substructure as well. This functionality is straightforward to implement in MCS algorithms using a subgraph enumeration approach[32,37] but more involved when using, for example, the maximum clique approach.

In practice, our algorithms provide three options for handling ring bonds (the same as in ref 31), which are illustrated by Figure 12. The first option is to ignore information about ring membership, i.e., ring bonds are allowed to match chain bonds as well (Figure 12a); the second one is to allow ring bonds to match ring bonds only, but rings need not match fully (Figure 12b); while the third one is preserving rings, i.e., not allowing rings to match partially in the MCS (Figure 12c). The first option is the default behavior of the algorithms, and the second one is also trivial to implement by incorporating ring−chain topology information into the bond type. The third option is, however, more complicated. We implemented it by limiting the search space using the second option and then removing broken rings that do end up in the solution as postprocessing.

The rationale for this approach is that optimizing for the size of the result already favors matching full rings of the same size, and by disallowing matching of rings and chains, provided that the algorithm is accurate enough, we usually end up with a good approximation to the desired result. However, as shown in Figure 13, this approach may fail in some cases, so it is only viable in heuristic algorithms. To further increase the accuracy of the algorithm, the ring or chain topology information on

DOI: 10.1021/acs.jcim.5b00036
J. Chem. Inf. Model. 2015, 55, 941−955

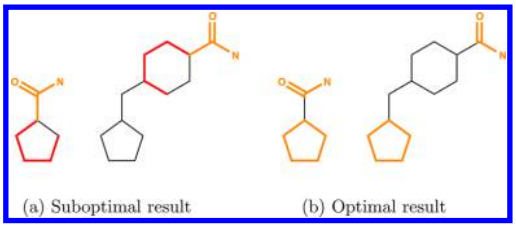


Figure 13. Example where removing bonds of partially matched rings results in a suboptimal solution. The colored bonds represent the common substructure found by the algorithm, and the red bonds are the ones removed during the postprocessing step.

bonds can also be incorporated into the ECFP hash codes described earlier.

We note that in most cases, the request to avoid breaking rings does not extend to large rings (macrocycles). Therefore, our implementations consider only rings within a configurable size limit (by default, this limit is set to 8).

Generic Atoms and Bonds. In real-world applications, generic query structures are often used. These structures may contain generic atoms and bonds that represent chemical atoms and bonds of multiple types. Examples include *halogen* atom, *metal* atom, and *any* atom as well as *single-or-double* bond and *any* bond. We refer to such atoms and bonds as generic features.

Following the needs of applications at ChemAxon, our MCS algorithms support generic features, but only in one of the input molecules, namely, the query. In this case, representing molecules simply as colored graphs no longer suffices. We can represent them, for example, by assigning a nonempty set of labels to each node and edge and by allowing matching of two nodes or two edges if the intersection of their label sets is nonempty. Luckily, the algorithms do not need to be substantially modified to accommodate this more general definition. Still, a number of difficulties arise, upon which we elaborate in the following.

First of all, we note that in case of generic query features, the common substructure might actually differ in the query and the target. This might be strange, but it does not cause substantial difficulties. If we need to represent the common substructure separately from the input molecules, we always extract the matched parts of the target in order to avoid generic atoms and bonds.

A much less obvious problem is that matching bonds correctly becomes more complicated, as demonstrated in Figure 14. Luckily, this phenomenon happens only when two bonds match only in one orientation but matching adjacent bonds forces the other orientation. Therefore, these cases can be avoided easily. When checking whether adjacent bonds e_1 and f_1 of the query can be matched to adjacent bonds e_2 and f_2 of the target, we check not only whether the common atoms match but also whether the other, noncommon atoms of e_1 and e_2 and of f_1 and f_2 match. For example, in clique-based algorithms, this can be done during construction of the modular product graph. Two nodes should not be considered compatible if they do not satisfy this condition.

Another problem with generic features is that they are hard to incorporate into the ECFP heuristic described earlier. If correctness for query features is desired, the hash code assigned to a generic atom should equal the hash code assigned to each atom that it may match. Because of transitivity, this would mean that the hash codes of atoms with different types would have to be equal, and the ECFP heuristic would become much less effective, even for those parts of the molecules that do not contain query features. Another possible solution would be to store a set of hash values for each environment of radius r for each atom by calculating all possible hash values of the environment considering the specific types a generic atom or bond in that environment can represent. The problem with this approach is that the size of these sets can quickly grow too large to manage, and even in moderate cases (few generic atoms or bonds, small r), it would significantly complicate and slow down the calculation of ECFP scores.

An alternative approach is to assign hash values to generic atoms and bonds that are different from the hash values of specific atoms and bonds. This is simple to implement and the calculation remains fast, but the resulting scores can be misleading if the environment contains a generic atom or bond. More specifically, the hash codes might be different even if the environments could be matched. Our implementations use this approach, as generic features are present in only a small fraction of use cases, and even in these cases, if the inaccurate ECFP scores lead the algorithm to a suboptimal solution, subsequent iterations can improve upon this.

In addition, we also have to modify the upper bound calculation methods to account for generic atoms and bonds. In the first algorithm we use, bonds, along with their atoms, are grouped by type in both molecules, and the minimum count is summed for each type group. Since we allow generic features only in the query, the target atoms and bonds are always exact, and matching them to each other remains an equivalence relation. Therefore, the type categories should be defined using the target atoms and bonds, and query bonds with generic type or generic atom should be included in each category that they match. However, this may result in a worse bound, sometimes even higher than the number of bonds in the query (which is a trivial upper bound).

We also implemented the more accurate upper bound calculation method described by Raymond et al.,[8] which finds a maximum-weight matching between query and target atoms of each type. This algorithm can also be adapted to generic query features according to similar considerations.

RESULTS

The different algorithms and heuristics presented in this study were extensively benchmarked. This section contains a selection of measurements and results that demonstrate the

(a) Query structure and its line graph (b) Target molecule and its line graph

Figure 14. A particular problem caused by generic atoms. The query structure contains a generic atom, denoted by "A", that matches all types of chemical atoms. The line graphs of these two molecular graphs match completely, but the original graphs do not, even though no ΔY exchange has occurred.

DOI: 10.1021/acs.jcim.5b00036
J. Chem. Inf. Model. 2015, 55, 941−955

Table 1. Properties of the Main Data Sets[a]

	K	N	n_{min}	n_{max}	n_{avg}	m_{min}	m_{max}	m_{avg}	s_{min}	s_{max}	s_{avg}
NCI1	32	496	24	24	24.0	26	30	27.6	0.43	1.00	0.69
NCI2	16	120	37	46	42.5	41	51	46.8	0.36	1.00	0.59
NCI3	16	120	71	78	74.8	71	84	77.9	0.18	1.00	0.52
NCI4	12	66	106	123	115.6	112	125	117.9	0.44	1.00	0.70
NCI-S	39	741	20	45	30.4	22	50	33.4	0.35	1.00	0.64
CAH2	116	6670	8	35	20.0	8	37	21.0	0.18	1.00	0.48
CDK2	154	11781	9	36	24.7	10	43	27.2	0.16	1.00	0.55
NEU	15	105	20	28	21.7	20	29	21.8	0.42	1.00	0.69
NCI-I	40	20	8	241	102.8	8	264	112.1	0.70	0.87	0.83

[a] K: number of molecules. N: number of MCS problem instances ($K/2$ for NCI-I and $K(K-1)/2$ for the other data sets). $n_{min/max/avg}$ and $m_{min/max/avg}$: minimum/maximum/average numbers of heavy atoms and bonds between heavy atoms, respectively, in the K molecules. $s_{min/max/avg}$: minimum/maximum/average similarity scores among the N molecule pairs.

Table 2. Effects of the Improvements on the Accuracy and Running Time of the Clique-Based Algorithm[a]

	average result size ratio					average running time (ms)				
	none	CONN	ECFP	MAP	all	none	CONN	ECFP	MAP	all
NCI1	0.934	0.938	0.941	0.936	0.943	14.4	13.8	14.2	15.7	11.0
NCI2	0.876	0.892	0.888	0.882	0.895	23.4	20.0	22.0	25.0	19.4
NCI3	0.875	0.893	0.894	0.884	0.907	74.1	57.3	65.0	79.3	59.2
NCI4	0.827	0.863	0.859	0.855	0.900	250.3	198.9	291.2	345.4	237.9
NCI-S	0.959	0.969	0.971	0.964	0.972	12.0	9.8	10.8	12.7	7.1
CAH2	0.942	0.943	0.942	0.942	0.943	5.0	4.9	4.9	5.3	3.6
CDK2	0.884	0.888	0.888	0.886	0.890	11.3	10.6	10.8	11.9	10.2
NEU	0.934	0.939	0.939	0.936	0.939	8.3	8.2	7.8	8.0	6.0

[a] None: the basic algorithm without heuristics was used. CONN: only the connectivity heuristic was applied. ECFP: only the ECFP heuristic was applied. MAP: only the mapping optimization heuristic was applied. All: all of the improvements (including the early termination heuristic) were applied. Each variant used the improved representation of the modular product graph.

effects of the developed improvements as well as the overall performance of the implementations. The latter is also compared with the corresponding method of the Indigo toolkit[56] and the KCOMBU code of Kawabata.[29,57]

Test Setup. The presented algorithms were implemented in the Java programming language as part of the JChem software suite[58] of ChemAxon. For the experimental evaluation, we used JChem 14.10.6, the latest release at the time of writing. The benchmarking was conducted on a PC with an Intel Core i7-4800MQ 2.7 GHz CPU and 16 GB of RAM running a 64-bit Microsoft Windows 7 Professional SP1 operating system. The codes were compiled and run using Java 1.8.0_20.

The experiments were carried out using a test suite of nine data sets, which are summarized in Table 1. Some of these sets were created for this study by selecting molecules from a public database provided by the National Cancer Institute (NCI).[59] The sets NCI1, NCI2, NCI3, and NCI4 are intended to represent rather different ranges of molecule size, as the accuracy and running time of MCS algorithms strongly depend on that. However, within each of these sets, the molecules are similar to each other in both size and characteristics. Another data set, denoted by NCI-S, consists of those 39 compounds from the NCI database that contain both benzenesulfonamide and indole as a substructure. This set is more diverse in size than the previous ones, but it also contains similar molecules, since they are ensured to have a relatively large common substructure. In the case of these data sets, MCS search was performed pairwise, once for each pair of molecules. In total, $K(K-1)/2$ problem instances are derived from a set of K molecules.

In addition, we also used three of the five data sets that were introduced in the experimental study of the KCOMBU code.[29] They are different collections of protein ligands that can be downloaded from the Worldwide Protein Data Bank (wwPDB)[60,61] according to the list files that are available on the Web site of KCOMBU.[57] Our test suite included the following three of these data sets: CAH2 (carbonic anhydrase 2), CDK2 (cyclin-dependent protein kinase 2) and NEU (neuraminidase). They were also processed pairwise. However, in accordance with Kawabata's study,[29] we ignored bond types in the case of these data sets.

Finally, we generated a special data set, denoted as NCI-I, that contains molecule pairs of gradually increasing size. First, two large and very similar compounds were selected from the NCI database (both containing about 240 heavy atoms). Then we successively removed small parts of both molecules. The pairs of similar structures obtained thereby can be used, in the reverse order of their construction, as MCS problem instances of steadily increasing size. Despite its artificial construction, this data set also consists of reasonable molecules, and it allows easier evaluation and clear visualization of the experimental behavior of the algorithms as a function of the input size.

All of the data sets are available in the Supporting Information. Table 1 presents basic statistical data for them. The similarities of the molecule pairs are measured using the following Tanimoto (Jaccard) score:

$$\frac{m_c}{m_1 + m_2 - m_c}$$

where m_1 and m_2 denote the numbers of bonds in the query and target molecules, respectively, and m_c denotes an upper

DOI: 10.1021/acs.jcim.5b00036
J. Chem. Inf. Model. 2015, 55, 941−955

bound on the number of bonds in their MCS, which was computed using the more accurate upper bound calculation method of Raymond et al.[8]

Improvements. In what follows, experimental results are presented to demonstrate the effects of the different improvements on the accuracy and running time of the implemented algorithms. Unless stated otherwise, we consider disconnected MCS search.

To measure the accuracy of the algorithms, we introduced the notion of *result size ratio*. It is defined as the ratio of the number of bonds in a common substructure found by an algorithm to the upper bound on this number calculated using the more accurate method. (We did not consider trivial cases where the upper bound is zero and thus the ratio is undefined.) This measure expresses a relative accuracy and it makes it easier to aggregate and compare results obtained for problem instances of different sizes.

The result size ratio is always between 0 and 1, but reaching 1 is usually not possible because the upper bound is also an estimate and the exact MCS is usually not known (running an exact algorithm would be infeasible in most cases). In fact, the ratio gives a bound on the error of both the upper bound calculation and the MCS algorithm. For example, a ratio of 0.95 can mean that the upper bound is optimal but the result of the MCS algorithm has 0.95 times as many bonds as an optimal solution, that the result of the MCS algorithm is optimal but the upper bound is $0.95^{-1} \approx 1.053$ times the optimal solution, or anything in between.

We evaluated five variants of the clique-based algorithm that differed in regard to whether the different heuristics were applied, but all of them used the improved representation. Table 2 reports the average result size ratio and the average running time (in milliseconds) for these variants measured on eight data sets. The result size ratios are also compared in Figure 15. These experiments show how each heuristic

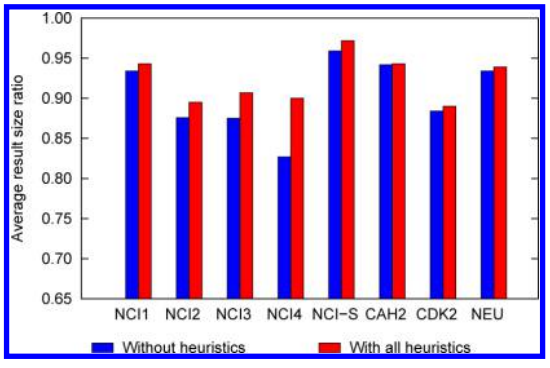


Figure 15. Effects of the improvements on the accuracy of the clique-based algorithm.

improves the accuracy and that their combination works quite well: using all of the heuristics together yields substantially better results than using any one of them alone. As mentioned above, the mapping optimization heuristic, beyond its primary goal, can also help find larger common substructures, as illustrated in Figure 11. The results in Table 2 verify that it does improve the accuracy in several cases.

These experiments also show that the larger the input molecules are, the more significant are the heuristic improvements (see the results for the data sets NCI1, NCI2, NCI3, and NCI4). Figure 16 also illustrates this by depicting the result size

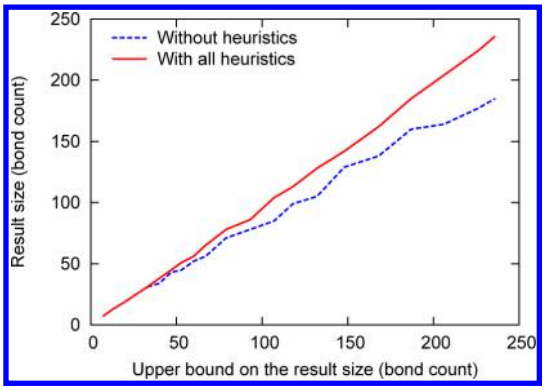


Figure 16. Effect of the improvements on the accuracy of the clique-based algorithm as measured on the NCI-I data set.

as a function of its upper bound for the problem instances defined by the NCI-I data set. In the case of the largest instances, the heuristics have a great impact on the accuracy: the common substructures found by the improved algorithm are 15−30% larger than the ones found by the original method. Furthermore, the improvements also turned out to make the algorithm more robust.

The results in Table 2 also verify that the three main heuristics do not have a significant negative impact on the running time of the algorithm. On the contrary, combined with the early termination technique, they can even make the algorithm faster in several cases. In fact, using all of the heuristics together reduces the average running time on each data set. Figure 17 visualizes the effect of the early termination

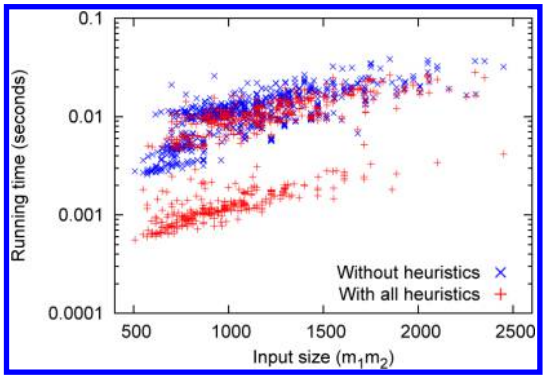


Figure 17. Effects of the improvements on the running time of the clique-based algorithm as measured on the NCI-S data set.

heuristic on the NCI-S data set. The running time is depicted for each problem instance as a function of the input size, which is measured as the product of the bond counts in the query and target molecules. The cases when early termination was achieved are clearly separated from the others. While the solution is also guaranteed to be optimal in these cases, it may be optimal in other cases as well (provided that the upper bound is suboptimal).

Some of the presented improvements were also applied to the build-up algorithm. The connectivity and ECFP heuristics were used similarly as in the clique-based method, but mapping optimization was not implemented for the build-up algorithm. Table 3 and Figure 18 report the benchmark results for different variants of the build-up algorithm. An interesting experimental result is that the ECFP heuristic turned out to have a variable impact on this algorithm. On some data sets it

DOI: 10.1021/acs.jcim.5b00036
J. Chem. Inf. Model. 2015, 55, 941−955

Table 3. Effects of the Improvements on the Accuracy and Running Time of the Build-Up Algorithm[a]

	average result size ratio				average running time (ms)			
	none	CONN	ECFP	all	none	CONN	ECFP	all
NCI1	0.756	0.890	0.792	0.909	4.5	3.3	5.3	3.5
NCI2	0.729	0.842	0.739	0.838	8.8	7.1	10.6	6.8
NCI3	0.761	0.859	0.755	0.880	46.9	29.9	45.9	29.8
NCI4	0.739	0.850	0.737	0.885	335.2	206.1	387.7	213.5
NCI-S	0.864	0.956	0.913	0.959	5.4	3.3	5.4	3.5
CAH2	0.833	0.907	0.842	0.912	1.1	0.7	1.2	0.7
CDK2	0.702	0.822	0.721	0.831	2.5	1.7	2.8	1.7
NEU	0.770	0.914	0.868	0.923	2.4	1.7	3.4	1.6

[a]None: the basic algorithm without heuristics was used. CONN: only the connectivity heuristic was applied. ECFP: only the ECFP heuristic was applied. All: both improvements were applied.

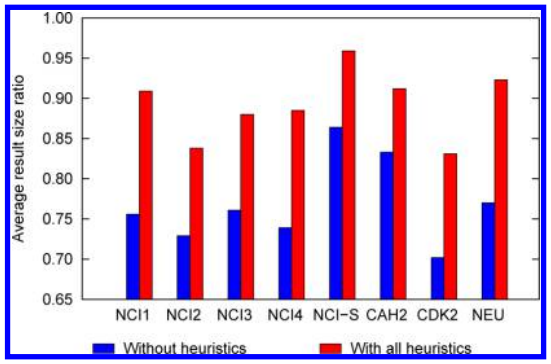


Figure 18. Effects of the improvements on the accuracy of the build-up algorithm.

yields a minor improvement (e.g., NCI2 and CAH2) or even slightly decreases the accuracy (NCI3 and NCI4), while it achieves great improvement on other sets (e.g., NCI1, NCI-S, and NEU). These differences are probably due to the less robust behavior of the build-up approach compared with the clique-based methods. The connectivity heuristic, however, consistently makes the build-up algorithm much more accurate (see Table 3). Furthermore, combined with this improvement, the ECFP heuristic also becomes more robust.

The primary focus of this study was the disconnected MCS search, but we also experimented with the connected case, and the developed improvements turned out to be similarly effective. The connectivity heuristic is especially important: it incorporates a requirement instead of a preference when connected common substructures are to be found.

Comparison of Algorithms. Besides evaluating the effects of the developed improvements, we also intended to compare the final implementations of the two different algorithmic approaches. Table 4 and Figure 19 present benchmark results for three implementations: MC denotes the maximum-clique-based algorithm with all of the improvements; MC-F denotes a fast variant of the MC algorithm in which significantly fewer iterations were performed within the clique detection algorithm (20 instead of 1000) but all of the heuristics were applied in the same way; and BU denotes the build-up algorithm with all of the heuristics. These results clearly show that even though both are heavily optimized, the MC implementation substantially outperforms the BU code in terms of accuracy. However, its average running time is larger on molecules up to a certain size, which is why the MC-F variant was also considered. It turned out to be consistently faster than BU while still producing much better results (see Figure 19).

Table 4. Comparison of the Accuracies and Running Times of Different Algorithms

	average result size ratio			average running time (ms)		
	MC	MC-F	BU	MC	MC-F	BU
NCI1	0.943	0.939	0.909	11.0	3.1	3.5
NCI2	0.895	0.878	0.838	19.4	3.5	6.8
NCI3	0.907	0.899	0.880	59.2	12.8	29.8
NCI4	0.900	0.896	0.885	237.9	69.6	213.5
NCI-S	0.972	0.971	0.959	7.1	1.6	3.5
CAH2	0.943	0.939	0.912	3.6	0.6	0.7
CDK2	0.890	0.879	0.831	10.2	1.5	1.7
NEU	0.939	0.937	0.923	6.0	1.4	1.6

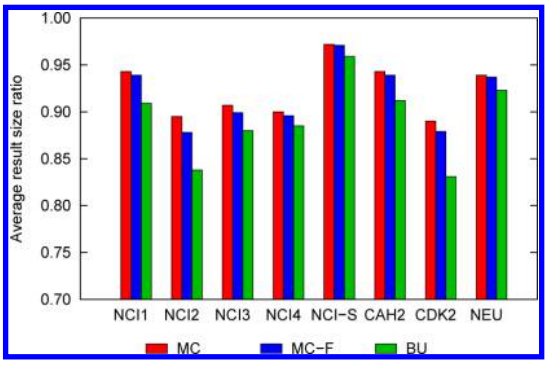


Figure 19. Comparison of the accuracies of different algorithms.

Figures 20 and 21 provide more insight into the running time trends of these three codes. Figure 20 shows that MC-F is the fastest implementation and that MC is usually slower than BU,

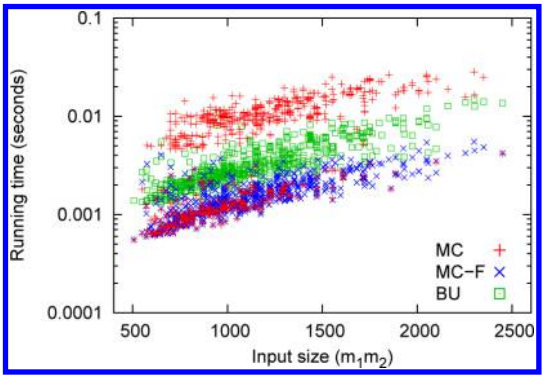


Figure 20. Comparison of the running times of different algorithms as measured on the NCI-S data set.

DOI: 10.1021/acs.jcim.5b00036
J. Chem. Inf. Model. 2015, 55, 941−955

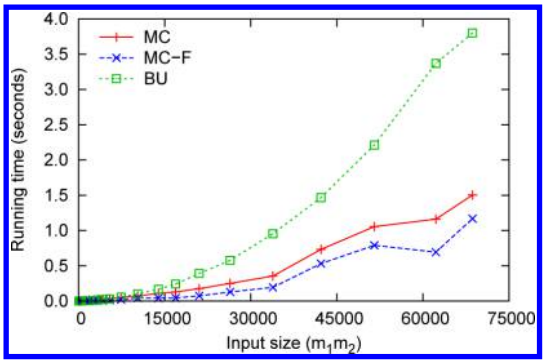


Figure 21. Comparison of the running times of different algorithms as measured on the NCI-I data set.

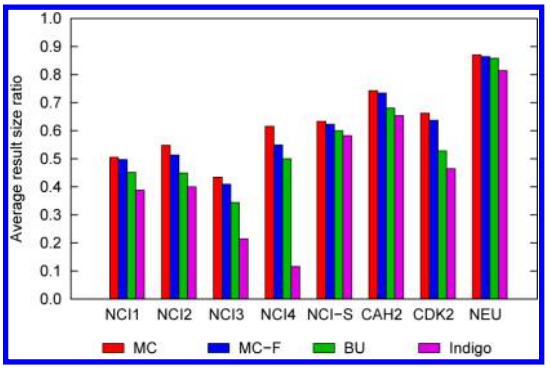


Figure 22. Comparison of the accuracies of the presented algorithms and Indigo (connected MCS).

although it is often as efficient as MC-F because of the early termination technique. On the other hand, Figure 21 demonstrates that these relations are different in case of large input molecules. The running time of BU shows a worse trend as the input size increases, and it is significantly slower than MC for the largest molecules (having 150 or more heavy atoms).

These three implementations were also evaluated in case of connected MCS search, and their relative performance turned out to be rather similar. The results are presented in the next subsection, compared with the algorithm implemented in the Indigo toolkit.[56]

On the basis of these results, we can conclude that the clique-based implementations turned out to be superior to the build-up one, although the developed heuristic improvements substantially decreased the difference. It should be noted, however, that the performance of a clique-based algorithm strongly depends on the applied clique detection method. We used the heuristic algorithm of Grosso et al.[46] for this purpose because it is both simple and very efficient. On the other hand, an important advantage of the build-up algorithm is that it is much easier to implement, so it can be a reasonable alternative, especially if it is enhanced with efficient heuristics.

Comparison with Indigo. Indigo[56] is a widely used general-purpose open-source cheminformatics toolkit. For MCS search, it provides both an exact algorithm and an approximate algorithm. The latter is an implementation of the two-stage optimization method (called 2DOM) of Funabiki and Kitamichi,[62] and solves the connected MCES problem. In what follows, we compare it with our implementations. Both the Indigo library and the simple benchmarking code we used were compiled with GCC 4.8.1 using the -O2 optimization flag. We used Indigo 1.1.12, the latest release at the time of writing.

The experimental results are summarized in Table 5 and Figure 22. It should be noted that the connected MCS is often much smaller than the disconnected MCS, but the upper bound calculation method does not consider connectivity. Therefore, the result size ratios tend to be significantly smaller than in the disconnected case, but they are still useful for comparing the effectiveness of different algorithms.

The relative performances of our implementations are quite similar to the disconnected case, with the only exception that BU turned out to be relatively faster. In regard to accuracy, however, BU performs substantially worse than MC and MC-F, just as in the case of disconnected MCS search. On the other hand, all of them greatly outperform the Indigo algorithm. While the running time of Indigo is similar to or slightly larger than that of MC, it is much less accurate than the other methods, especially on large molecules. Our implementations found 1.5−2 times larger common substructures on average for the problem instances in the NCI3 data set and about 5 times larger results on average for the NCI4 set.

Comparison with KCOMBU. The presented algorithms were also compared with the KCOMBU code of Kawabata.[29,57] This is the authoritative implementation of the build-up algorithm, which has turned out to be effective in different applications[21,63,64] and hence serves as a reference. However, KCOMBU solves the MCIS variant of the MCS problem, while our implementations solve the MCES problem (cf. Figure 1). This difference makes it difficult to compare these algorithms in a reasonable and fair way, so we applied a simple and practical approach: our experiments were restricted to those problem instances for which each considered MCES algorithm produced a common induced subgraph. This means that their results are also feasible for an MCIS algorithm, so MCES algorithms do not have inherent advantage in these cases. Furthermore, we measured the result size in terms of the atom count (instead of the bond count) of the common substructure. This is beneficial for the MCIS approach, especially if its result contains isolated

Table 5. Comparison of the Accuracies and Running Times of the Presented Algorithms and Indigo (Connected MCS)

	average result size ratio				average running time (ms)			
	MC	MC-F	BU	Indigo	MC	MC-F	BU	Indigo
NCI1	0.505	0.497	0.451	0.388	16.6	2.2	1.3	10.8
NCI2	0.548	0.513	0.449	0.400	18.6	2.9	2.7	30.7
NCI3	0.434	0.409	0.344	0.215	38.3	7.8	12.2	99.6
NCI4	0.615	0.549	0.500	0.116	125.5	33.1	107.0	190.6
NCI-S	0.633	0.623	0.600	0.582	9.2	1.6	1.5	12.9
CAH2	0.743	0.734	0.680	0.653	4.4	0.5	0.4	7.3
CDK2	0.662	0.637	0.528	0.464	10.6	1.4	0.8	10.6
NEU	0.870	0.864	0.858	0.814	6.5	1.1	1.1	9.0

DOI: 10.1021/acs.jcim.5b00036
J. Chem. Inf. Model. 2015, 55, 941−955

atoms, because they are infeasible for MCES algorithms, as they operate on bonds.

In this comparison, only three input sets were included, namely, NCI-S, CAH2, and CDK2, which represent a sufficiently large number of problem instances, even when filtered as described above. We considered only disconnected MCS search in this case, for which 215 out of the 741 instances from the NCI-S set, 1316 out of 6670 from the CAH2 set, and 372 out of 11781 from the CDK2 set fulfill the restriction.

The KCOMBU source code was obtained from its Web site,[57] and it was extended with a short code snippet enabling us to measure its running time without the file input−output. The code was compiled with GCC 4.8.1 using the -O2 optimization flag, and it was executed using the following command line options: "-alg B" (the build-up algorithm is selected), "-con D" (disconnected MCIS is to be found), and "-at E" (atoms are classified by their element types). Furthermore, the "-bo C" option was applied for the NCI-S problem instances to consider the bond types, while "-bo X" was used for the CAH2 and CDK2 instances to ignore the bond types (as in earlier experiments).

Table 6 presents the results of these benchmarks. Although the method of comparison is advantageous for KCOMBU, the MC and BU implementations performed much better. They turned out to be an order of magnitude faster, while their results are more accurate. In case of the NCI-S and CDK2 sets, the common substructures they found have 8−10% more atoms on average than the results of KCOMBU. These measurements also verify the effectiveness of the improvements developed in this study.

Table 6. Comparison of the Presented Algorithms and KCOMBU[a]

	average result size			average running time (ms)		
	MC	BU	KCB	MC	BU	KCB
NCI-S	22.66	22.66	20.51	3.5	2.5	59.2
CAH2	10.97	10.93	10.76	1.0	0.5	6.5
CDK2	16.59	16.57	15.37	3.1	1.5	22.7

[a]KCB denotes KCOMBU. The benchmark results were restricted to the problem instances for which each algorithm finds a common induced substructure. The result size was measured in terms of the number of heavy atoms in the common substructure.

CONCLUSION

Several methods have been proposed to improve the accuracy and efficiency of different algorithms for the maximum common substructure (MCS) problem as well as to increase the chemical relevance of their results. Some of these ideas have been mentioned in the literature before (the improved representation for the clique-based method, the early termination technique, and the connectivity heuristic), but they have been applied to different algorithms or in a different way. The other improvements and extensions are, to the authors' knowledge, novel ideas that have not been described before.

Two different MCS algorithms have been implemented and compared: the well-known clique-based method and the recent build-up algorithm. Both algorithms were greatly improved by the developed heuristics, but the clique-based implementation turned out to be superior according to extensive experimental evaluation. The comparison also included publicly available MCS solutions, namely, KCOMBU and the corresponding algorithm in the Indigo toolkit, both of which were significantly outperformed by our implementations.

One of the most important heuristics introduced in this study utilizes extended connectivity fingerprints (ECFPs) in the context of MCS search. It remarkably improves the accuracy of both algorithms, especially when it is combined with the connectivity heuristic. Furthermore, we have addressed the problem of providing a chemically relevant atom mapping. We have proposed a general method that does not result in a combinatorial explosion but handles typical cases well. Besides mapping optimization, this heuristic also turned out to be effective in finding larger common substructures.

The proposed MCS algorithms have been integrated into multiple products of ChemAxon Ltd., which are used by several leading international companies in the pharmaceutical industry. The implementations are part of the JChem software suite, which is available free of charge for academic research and evaluation purposes at http://www.chemaxon.com.

ASSOCIATED CONTENT

Supporting Information

Zip files containing the input structure data files (in SDF and SMILES format) used for the presented experimental study. This material is available free of charge via the Internet at http://pubs.acs.org.

AUTHOR INFORMATION

Corresponding Authors

*E-mail: enpraai@inf.elte.hu.

*E-mail: pkovacs84@chemaxon.com.

Notes

The authors declare no competing financial interest.

ACKNOWLEDGMENTS

This research was conducted at ChemAxon Ltd. in cooperation with Eötvös Loránd University. The authors express their appreciation to Miklós Vargyas and the staff of ChemAxon for their assistance. It has been a privilege to work with them. In addition, special thanks are due to István Fekete and Krisztián Tichler for their support in writing this paper.

REFERENCES

(1) Ehrlich, H.-C.; Rarey, M. Maximum Common Subgraph Isomorphism Algorithms and Their Applications in Molecular Science: A review. *Wiley Interdiscip. Rev.: Comput. Mol. Sci.* **2011**, *1*, 68−79.

(2) Dean, P. M. *Molecular Similarity in Drug Design*; Blackie Academic & Professional: London, 1995.

(3) Sheridan, R. P.; Kearsley, S. K. Why Do We Need So Many Chemical Similarity Search Methods? *Drug Discovery Today* **2002**, *7*, 903−911.

(4) Willett, P.; Barnard, J. M.; Downs, G. M. Chemical Similarity Searching. *J. Chem. Inf. Comput. Sci.* **1998**, *38*, 983−996.

(5) Willett, P. Searching Techniques for Databases of Two- and Three-Dimensional Chemical Structures. *J. Med. Chem.* **2005**, *48*, 4183−4199.

(6) Willett, P. Similarity Methods in Chemoinformatics. *Annu. Rev. Inf. Sci. Technol.* **2009**, *43*, 1−117.

(7) Bunke, H.; Shearer, K. A Graph Distance Metric Based on the Maximal Common Subgraph. *Pattern Recognit. Lett.* **1998**, *19*, 255−259.

(8) Raymond, J. W.; Gardiner, E. J.; Willett, P. RASCAL: Calculation of Graph Similarity Using Maximum Common Edge Subgraphs. *Comput. J.* **2002**, *45*, 631−644.

DOI: 10.1021/acs.jcim.5b00036
J. Chem. Inf. Model. 2015, 55, 941−955

(9) Raymond, J. W.; Willett, P. Effectiveness of Graph-Based and Fingerprint-Based Similarity Measures for Virtual Screening of 2D Chemical Structure Databases. *J. Comput.-Aided Mol. Des.* **2002**, *16*, 59−71.
(10) Johnson, M. A., Maggiora, G. M. *Concepts and Applications of Molecular Similarity*; Wiley-Interscience: New York, 1990.
(11) Boström, J.; Hogner, A.; Schmitt, S. Do Structurally Similar Ligands Bind in a Similar Fashion? *J. Med. Chem.* **2006**, *49*, 6716−6725.
(12) Stahl, M.; Mauser, H. Database Clustering with a Combination of Fingerprint and Maximum Common Substructure Methods. *J. Chem. Inf. Model.* **2005**, *45*, 542−548.
(13) Gardiner, E. J.; Gillet, V. J.; Willett, P.; Cosgrove, D. A. Representing Clusters Using a Maximum Common Edge Substructure Algorithm Applied to Reduced Graphs and Molecular Graphs. *J. Chem. Inf. Model.* **2007**, *47*, 354−366.
(14) Böcker, A. Toward an Improved Clustering of Large Data Sets Using Maximum Common Substructures and Topological Fingerprints. *J. Chem. Inf. Model.* **2008**, *48*, 2097−2107.
(15) Sheridan, R. P. The Most Common Chemical Replacements in Drug-Like Compounds. *J. Chem. Inf. Comput. Sci.* **2002**, *42*, 103−108.
(16) Southall, N. T. Kinase Patent Space Visualization Using Chemical Replacements. *J. Med. Chem.* **2006**, *49*, 2103−2109.
(17) Raymond, J. W.; Watson, I. A.; Mahoui, A. Rationalizing Lead Optimization by Associating Quantitative Relevance with Molecular Structure Modification. *J. Chem. Inf. Model.* **2009**, *49*, 1952−1962.
(18) McGregor, J. J.; Willett, P. Use of a Maximum Common Subgraph Algorithm in the Automatic Identification of Ostensible Bond Changes Occurring in Chemical Reactions. *J. Chem. Inf. Comput. Sci.* **1981**, *21*, 137−140.
(19) Fooshee, D.; Andronico, A.; Baldi, P. ReactionMap: An Efficient Atom-Mapping Algorithm for Chemical Reactions. *J. Chem. Inf. Model.* **2013**, *53*, 2812−2819.
(20) Marialke, J.; Körner, R.; Tietze, S.; Apostolakis, J. Graph-Based Molecular Alignment (GMA). *J. Chem. Inf. Model.* **2007**, *47*, 591−601.
(21) Kawabata, T.; Nakamura, H. 3D Flexible Alignment Using 2D Maximum Common Substructure: Dependence of Prediction Accuracy on Target-Reference Chemical Similarity. *J. Chem. Inf. Model.* **2014**, *54*, 1850−1863.
(22) Shearer, K.; Bunke, H.; Venkatesh, S. Video Indexing and Similarity Retrieval by Largest Common Subgraph Detection Using Decision Trees. *Pattern Recognit.* **2001**, *34*, 1075−1091.
(23) Conte, D.; Foggia, P.; Sansone, C.; Vento, M. Thirty Years of Graph Matching in Pattern Recognition. *Int. J. Pattern Recognit. Artif. Intell.* **2004**, *18*, 265−298.
(24) Trinajstic, N. *Chemical Graph Theory*; CRC Press: London, 1992.
(25) Garey, M. R.; Johnson, D. S. *Computers and Intractability; A Guide to the Theory of NP-Completeness*; W. H. Freeman & Co.: San Francisco, 1990.
(26) Brown, N. Chemoinformatics—An Introduction for Computer Scientists. *ACM Comput. Surv.* **2009**, *41*, 8−38.
(27) Raymond, J. W.; Willett, P. Maximum Common Subgraph Isomorphism Algorithms for the Matching of Chemical Structures. *J. Comput.-Aided Mol. Des.* **2002**, *16*, 521−533.
(28) Cao, Y.; Jiang, T.; Girke, T. A Maximum Common Substructure-Based Algorithm for Searching and Predicting Drug-Like Compounds. *Bioinformatics* **2008**, *24*, 366−374.
(29) Kawabata, T. Build-Up Algorithm for Atomic Correspondence between Chemical Structures. *J. Chem. Inf. Model.* **2011**, *51*, 1775−1787.
(30) Wang, Y.; Backman, T. W. H.; Horan, K.; Girke, T. fmcsR: Mismatch Tolerant Maximum Common Substructure Searching in R. *Bioinformatics* **2013**, *29*, 2792−2794.
(31) Hariharan, R.; Janakiraman, A.; Nilakantan, R.; Singh, B.; Varghese, S.; Landrum, G.; Schuffenhauer, A. MultiMCS: A Fast Algorithm for the Maximum Common Substructure Problem on Multiple Molecules. *J. Chem. Inf. Model.* **2011**, *51*, 788−806.
(32) Dalke, A.; Hastings, J. FMCS: A Novel Algorithm for the Multiple MCS Problem. *J. Cheminf.* **2013**, *5*, O6.
(33) McGregor, J. J. Backtrack Search Algorithms and the Maximal Common Subgraph Problem. *Software: Pract. Exper.* **1982**, *12*, 23−34.
(34) Chen, L.; Robien, W. MCSS: A New Algorithm for Perception of Maximal Common Substructures and Its Application to NMR Spectral Studies. 1. The Algorithm. *J. Chem. Inf. Comput. Sci.* **1992**, *32*, 501−506.
(35) Krissinel, E. B.; Henrick, K. Common Subgraph Isomorphism Detection by Backtracking Search. *Software: Pract. Exper.* **2004**, *34*, 591−607.
(36) Durand, P. J.; Pasari, R.; Baker, J. W.; Tsai, C.-c. An Efficient Algorithm for Similarity Analysis of Molecules. *Internet J. Chem.* **1999**, *2*, 1−16.
(37) Koch, I. Enumerating All Connected Maximal Common Subgraphs in Two Graphs. *Theor. Comput. Sci.* **2001**, *250*, 1−30.
(38) Raymond, J. W.; Gardiner, E. J.; Willett, P. Heuristics for Similarity Searching of Chemical Graphs Using a Maximum Common Edge Subgraph Algorithm. *J. Chem. Inf. Comput. Sci.* **2002**, *42*, 305−316.
(39) Brown, R. D.; Jones, G.; Willett, P.; Glen, R. C. Matching Two-Dimensional Chemical Graphs Using Genetic Algorithms. *J. Chem. Inf. Comput. Sci.* **1994**, *34*, 63−70.
(40) Wang, T.; Zhou, J. EMCSS: A New Method for Maximal Common Substructure Search. *J. Chem. Inf. Comput. Sci.* **1997**, *37*, 828−834.
(41) Takahashi, Y.; Sukekawa, M.; Sasaki, S. Automatic Identification of Molecular Similarity Using Reduced-Graph Representation of Chemical Structure. *J. Chem. Inf. Comput. Sci.* **1992**, *32*, 639−643.
(42) Rarey, M.; Dixon, J. S. Feature Trees: A New Molecular Similarity Measure Based on Tree Matching. *J. Comput.-Aided Mol. Des.* **1998**, *12*, 471−490.
(43) Barker, E. J.; Buttar, D.; Cosgrove, D. A.; Gardiner, E. J.; Kitts, P.; Willett, P.; Gillet, V. J. Scaffold Hopping Using Clique Detection Applied to Reduced Graphs. *J. Chem. Inf. Model.* **2006**, *46*, 503−511.
(44) Levi, G. A. Note on the Derivation of Maximal Common Subgraphs of Two Directed or Undirected Graphs. *Calcolo* **1973**, *9*, 341−352.
(45) Nicholson, V.; Tsai, C.-C.; Johnson, M.; Naim, M. A Subgraph Isomorphism Theorem for Molecular Graphs. In *Graph Theory and Topology in Chemistry*; Elsevier: Amsterdam, 1987; pp 226−230.
(46) Grosso, A.; Locatelli, M.; Pullan, W. Simple Ingredients Leading to Very Efficient Heuristics for the Maximum Clique Problem. *J. Heuristics* **2008**, *14*, 587−612.
(47) Whitney, H. Congruent Graphs and the Connectivity of Graphs. *Am. J. Math.* **1932**, *54*, 150−168.
(48) Harary, F. *Graph Theory*; Addison-Wesley: Reading, MA, 1969.
(49) Bolton, E. E.; Wang, Y.; Thiessen, P. A.; Bryant, S. H. PubChem: Integrated Platform of Small Molecules and Biological Activities. In *Annual Reports in Computational Chemistry*, Vol. *4*; American Chemical Society: Washington, DC, 1990; Chapter 12.
(50) National Center for Biotechnology Information. The PubChem Project. http://pubchem.ncbi.nlm.nih.gov (accessed March 2015).
(51) National Center for Biotechnology Information. PubChem Compound Structure Search.http://pubchem.ncbi.nlm.nih.gov/search/ (accessed March 2015).
(52) 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.
(53) Rogers, D.; Hahn, M. Extended-Connectivity Fingerprints. *J. Chem. Inf. Model.* **2010**, *50*, 742−754.
(54) Stahl, M.; Mauser, H.; Tsui, M.; Taylor, N. R. A Robust Clustering Method for Chemical Structures. *J. Med. Chem.* **2005**, *48*, 4358−4366.
(55) Rahman, S.; Bashton, M.; Holliday, G.; Schrader, R.; Thornton, J. Small Molecule Subgraph Detector (SMSD) Toolkit. *J. Cheminf.* **2009**, *1*, 12.
(56) EPAM. Indigo Toolkit. http://lifescience.opensource.epam.com/indigo/ (accessed March 2015).

DOI: 10.1021/acs.jcim.5b00036
J. Chem. Inf. Model. 2015, 55, 941−955

(57) Kawabata, T. KCOMBU: For Matching Chemical Structure by the Build-Up Algorithm. http://strcomp.protein.osaka-u.ac.jp/kcombu/ (accessed March 2015).
(58) ChemAxon. JChem Base. http://www.chemaxon.com/products/jchem-base/ (accessed March 2015).
(59) National Cancer Institute. Downloadable Structure Files of NCI Open Database Compounds. http://cactus.nci.nih.gov/download/nci/ (accessed March 2015).
(60) Berman, H. M.; Westbrook, J.; Feng, Z.; Gilliland, G.; Bhat, T. N.; Weissig, H.; Shindyalov, I. N.; Bourne, P. E. The Protein Data Bank. *Nucleic Acids Res.* **2000**, *28*, 235−242.
(61) Berman, H. M.; Henrick, K.; Nakamura, H.; Markley, J. L. The Worldwide Protein Data Bank (wwPDB): Ensuring a Single, Uniform Archive of PDB Data. *Nucleic Acids Res.* **2007**, *35*, D301−D303.
(62) Funabiki, N.; Kitamichi, J. A Two-Stage Discrete Optimization Method for Largest Common Subgraph Problems. *IEICE Trans. Inf. Syst.* **1999**, *E82-D*, 1145−1153.
(63) Fukunishi, Y.; Nakamura, H. Integration of Ligand-Based Drug Screening with Structure-Based Drug Screening by Combining Maximum Volume Overlapping Score with Ligand Docking. *Pharmaceuticals* **2012**, *5*, 1332−1345.
(64) Brylinski, M. Nonlinear Scoring Functions for Similarity-Based Ligand Docking and Binding Affinity Prediction. *J. Chem. Inf. Model.* **2013**, *53*, 3097−3112.

DOI: 10.1021/acs.jcim.5b00036
J. Chem. Inf. Model. 2015, 55, 941−955