Fast Substructure Search in Combinatorial Library Spaces

Thomas Liphardt* and Thomas Sander*

Idorsia Pharmaceuticals Ltd.

E-mail: thomas.liphardt@idorsia.com; thomas.sander@idorsia.com

Abstract

We present an efficient algorithm for substructure search in combinatorial libraries defined by synthons, i.e. substructures with connection points. Our method improves on existing approaches by introducing powerful heuristics and fast fingerprint screening to quickly eliminate branches of non matching combinations of synthons. With this we achieve typical response times of a few seconds on a standard desktop computer for searches in large combinatorial libraries like the Enamine REAL space. We published the Java source as part of the OpenChemLib under the BSD license, and we implemented tools to enable substructure search in custom combinatorial libraries.

Introduction

Academic institutions and pharmaceutical companies have designed large synthetically accessible virtual combinatorial libraries,[1] and commercial players offer on-demand synthesis of individual library compounds. The size of synthetically accessible compounds has continued to grow rapidly in recent years, with suppliers like Enamine that are capable of reliably synthesizing compounds from their Enamine REAL space,[2] currently containing 30 billion structures. To answer the need, more efficient algorithms for searching large combinatorial spaces have been developed, which process reactant structures and connection rules instead of processing all enumerated structures. In 2001 Rarey and Stahl published a method to retrieve similar molecules based on feature tree similarity.[3] In 2009 Yu and Bakken employ a method called 'Monomer-based Similarity Searching (MoBSS)' that calculates atom pair descriptors from the reactant structures to locate matching combinatorial products.[4] More recent papers describe other methods for performing similarity and substructure searches in combinatorial spaces.[5,6]

Recently, Schmidt, Klein and Rarey presented a fast algorithm locating similar compounds from combinatorial libraries based on the maximum common induced substructure (MCIS).[7] Library molecules are considered matches if they contain a substructure, which exactly matches a large part of the query structure. For the special case where the MCIS size is required to match the entire query structure, the result is equivalent to a substructure search, therefore the algorithm can also be used to search for substructures. The problem of computing the MCIS in general is much harder than the problem to search for a substructure, but an excellent option if the goal is to locate similar compounds sharing some substructure.

The first efficient algorithm to search for substructures in combinatorial libraries was described by Rarey, Volkamer and Ehrlich.[8,9] Rarey et al. propose to first cut the query substructure into all possible sub-substructures, then to match these sub-substructure onto the combinatorial building blocks keeping track of sub-substructure connection points. The hit structures are then constructed from matching building blocks if the sub-substructures reconnection matches the query substructure. This procedure essentially reduces the substructure search problem spanning multiple building blocks to a large number of substructure searches in individual building blocks.

We rely on the same basic idea, but improve it (i) by adding a fingerprint pre-screening step to the substructure searching to quickly remove impossible candidates, and (ii) by providing heuristics that allow us to eliminate a large number of building block combinations. These heuristics perform exceptionally well in real-world scenarios, and searches in the Enamine REAL space (containing around 300 separate combinatorial libraries with over 1 million non-unique synthons) usually take a couple of seconds on a single thread.

While our implementation of the algorithm does not support SMARTS queries and no recursive definitions of substructures, it supports a number of features to define generalized substructures. For this, atoms and bonds can be annotated with specific query features. These can be narrowing, like enforcing an atom or bond to be part of an aromatic ring, require an atom to exactly carry two hydrogen atoms or a positive charge. Or they may be broadening, allowing an atom to be either an oxygen or any halogen, or a bond to be a single or a double bond. Our substructure search method comes with the limitation that the query structure must not consist of multiple disjoint substructures, it is however possible to employ so called bridged bonds (corresponding to arbitrary paths of specific length) to connect disjoint substructures. It is noteworthy, that in comparison to most other methods, there are no limitations on the combinatorial library itself, e.g. any kinds of bond formation, ring closures or macrocyle formation that can be represented in a suitable way is supported.

We believe that the ability to efficiently perform substructure, similarity, and pharmacophore searches in synthetically accessible large combinatorial spaces will contribute to make the drug discovery process more efficient, e.g. by multiplying the possibilities for hit expansions and scaffold hopping within pharmaceutical projects.

In order to foster easy and flexible usage of our method we provide it as open-source implementation in Java as part of the OpenChemLib.[10] Furthermore, we believe that the algorithm can easily be implemented for any other cheminformatics toolkit that provides substructure search capabilities, as it is a meta algorithm that performs substructure searches on the building blocks, which form the combinatorial library.

Methods

Fingerprints for substructure search

We denote the fingerprint of a structure s by $\mathrm{FP}(s)$. We can use a fingerprint to heuristically speed up substructure search, if there is an easily verifiable property for the fingerprints of graphs G and G' that must hold if G' is a subgraph of G. There are a number of binary fingerprints with the property that if G' is a subgraph of G, then $\mathrm{FP}(G') \subseteq \mathrm{FP}(G)$. This property holds for various common fingerprints, e.g. path fingerprints, tree fingerprints, or some configurations of the CSFP from Bellman and Rarey.[11] Any of these can be used to accelerate substructure search.

Query features and the fragment fingerprint of the OpenChemLib

For efficient substructure search we rely on the the fragment fingerprint from the OpenChemLib. The fragment fingerprint consists of 512 bits, containing the results of 512 separate searches for substructure features. These 512 substructure features are selected in a highly optimized way and can eliminate a high percentage of non-matching structures. Another key advantage of the fragment fingerprint over classic fingerprints is, that it can be used to create fingerprints from generalized substructure searches using query features of the OpenChemLib.

Superset testing using binary trees

Fingerprint filtering (i.e. superset testing of bitstrings) is a key step in the algorithm and crucial for performance. We perform fast fingerprint filtering by first sorting the library bitstrings into balanced binary trees. A balanced binary tree conceptually is a kd-tree generated on a set of bitstrings $B = \{b_1, b_2, .., b_n\}$, $b_i \in B^m$. To generate a balanced binary tree, we recursively split the set of bitstrings B_a by selecting a specific bit s that is balanced (i.e. occurs similarly often as 0 and 1 in B_a) into two roughly equal subsets $B_a^1 = \{b \in B_a : b(s) = 1\}$ and $B_a^0 = \{b \in B_a : b(s) = 0\}$. We denote the resulting parent tree nodes by N_i, where $s(N_i)$ is the splitting bit of the node, and N_i^0 / N_i^1 denote the 0 / 1 subtrees. While constructing the tree we keep track of the bits that are guaranteed to be 1. For the subtree rooted at N_i we denote this value by $\Sigma(N_i)$. The corresonding values for the subtrees are $\Sigma(N_i^0) = \Sigma(N_i)$, and $\Sigma(N_i^1) = \Sigma(N_i) \cup \{s(N_i)\}$. We stop the recursive splitting when the number of bitstrings in B_a is below a certain threshold L. We found that balancing the binary trees generally is easily possible with the descriptors and datasets that we tested.

For testing if B contains a superset of the bitstring q we can use the procedure described in Algorithm 1: we start at the root of the tree and omit all branches that describe a non-set bit in a position where the query bitstring contains a set bit. This procedure is particularly fast for processing query bitstrings that are not contained in the tree. Under the assumption that the tree is roughly balanced, this takes $O(\log(n))$ time, where n is the number of bitstrings in the tree.

Combinatorial libraries defined via synthon reactions

In this work we assume that the structures in the combinatorial library are defined via synthon reactions. A synthon is a vertex and edge labeled graph representing a chemical structure, with additional connector vertices c_x that are connected to atom vertices by a single edge. A synthon reaction is defined by multiple sets of synthons R_{F1}, R_{F2},..,R_{FN}.

Algorithm 1: Algorithm for Fast Superset Testing

```
   Input: Tree node $N$, query bitstring $q$
   Output: $\exists u \in N : u \supseteq q$
 1 Function testSuperset($N$,$q$):
 2     if $\Sigma(N) \supseteq q$ then
 3         return true
 4     end
 5     if $N$ is leaf then
 6         return $\exists u \in N : u \supseteq q$
 7     else
 8         if $q(s(N))$ then
 9             return testSuperset($N^1, q$)
10         else
11             return testSuperset($N^0, q$) $\vee$ testSuperset($N^1, q$)
12         end
13     end
14 End Function
```

All synthons in each set contain the same connector vertices (at most one per synthon and connector type). The feasible products of the synthon reaction are described by picking one synthon from each synthon set and then glueing together the connector vertices of the same type, where the resulting bond types are described by the edges to the connector vertices.

We classify a synthon reaction based on the number of synthon sets, and the types of connector vertices in each set. We do this in two ways, (i) by considering the *labeled connector configuration* that corresponds to the set of sets of connectors in each synthon set, and (ii) by considering the *unlabeled connector configuration* that corresponds to the set of number of connector vertices in each synthon set. The simplest synthon reaction, where two synthons are connected at one connector vertex type c_a we classify as $\{\{c_a\}, \{c_a\}\}$, and as $(1, 1)$ in the unlabeled representation. A synthon reaction where two synthons are connected at two connector vertex types c_a and c_b to close a cycle by $\{\{c_a, c_b\}, \{c_a, c_b\}\}$, or by $(2, 2)$ in the unlabeled representation. See Fig. 1 for examples of three different common connector vertex patterns that we find e.g. in synthon reactions of the Enamine REAL space.

Practically, we encode the connector vertices in a way such that it is possible for the descriptor to compute fingerprints for the fragments including the connector pair vertices. In our implementation we use atoms of transuranium elements to represent the connector vertices.

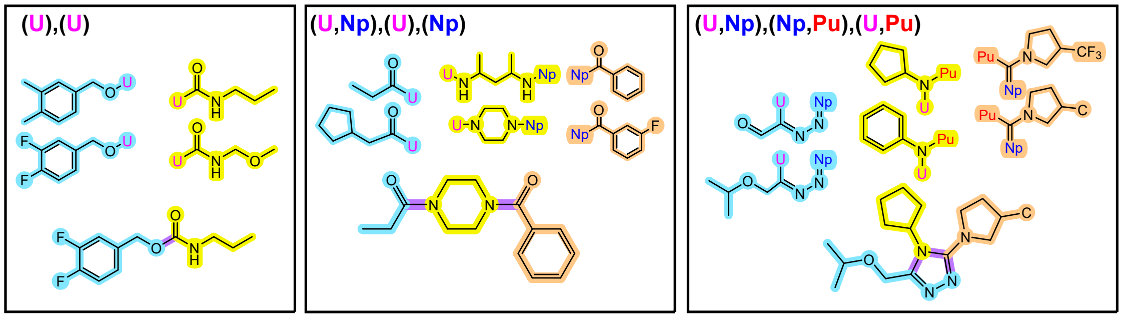


Figure 1: (A) Example for synthon reaction with one connector and connector configuration $\{U\}\{U\}$ describing four different carbamates. (B) Example for synthon reaction with two connectors and connector configuration $\{U, Np\}\{U\}\{Np\}$ describing 8 different bisamides. (C) Example for synthon reaction with three connectors and connector configuration $\{U, Np\}\{Np, Pu\}\{U, Np\}$ describing eight different triazoles.

The main algorithm for substructure search

In the first step we compute all possible combinations of synthon substructures that can yield the query substructure after synthon assembly. For c connectors this can be done simply by considering all sets of c bonds, and for each set removing the c bonds from the structure, trying to break it apart into multiple fragments. When removing a specific set of bonds q from the query structure, we perform the following step for every contained bond $b_1 = (x_{b1}, y_{b1})$

1. we attach a new vertex with label c_{b1} to x_{b1}, and another new vertex also with label c_{b1} to y_{b1}. We call this vertex pair the connector pair for b_1. The edge label (representing the bond type) for both new edges is similar to the edge label of the bond x_{b1}, y_{b1}.

2. we remove the bond x_{b1}, y_{b1}.

We denote the resulting graph by $F(q)$, and we call the cut-inducing set of bonds q the cut set. We say, $F(q)$ is valid if for each connector pair the two vertices fall into different disconnected components. The computation of all valid cut sets can be done by trying out all possible sets of bonds of size c, this results in $\sum_{i=1..c} \binom{n}{i}$ cut sets to validate, where c is the maximum number of connector pairs used in the combinatorial library. We call the resulting connected components in $F(q)$ the unlabeled query synthons. Note that there may exist more efficient methods to compute valid cut sets.

For each valid cut set q we first determine the unlabeled connector configuration of $F(q)$ and call it $C(q)$. We then check for each synthon reaction R, if based on its unlabeled connector configuration $L(R)$ could contain the cut set, i.e. if we find an injective map Ψ from the integers in $C(q)$ to the integers in $L(r)$, such that $\forall i \in C(q) : \Psi(i) >= i$.

We next describe the procedure required for every combination of a valid cutset $F(q)$ and a specific synthon reaction R. The connected components in $F(q)$ differ from the sub-substructures that we have to match in the synthon sets of the synthon reaction only in that they contain unlabeled connector vertices instead of the labeled connector vertices. To find all possible synthon substructures, we have to consider all possible labeling patterns of the connector vertices. For a synthon reaction with x connector vertex types we end up with $\binom{\|q\|}{x}(\|q\|!)$ labeled forests, because we have to consider all permutations of all connector vertex subsets of size $\|q\|$. The resulting forests of synthon substructures with labeled connector vertices then describe possible substructure hits in a synthon reaction: if we find an injective map from the computed synthon substructures to synthon sets of the synthon reaction, where these synthon sets contain synthons that contain the mapped synthon substructure, then we have a substructure hit. The full procedure that is performed for a specific cut set and a specific synthon reaction is summarized in Algorithm 2, where $\Psi(F_i, R)$ denotes the set of all injective maps from the connected components of the graph F_i to the synthon sets of synthon reaction R. Please note that two important pruning steps will be explained in the next sections.

Algorithm 2: Substructure Search in Synthon Reaction

```
   Input: Query structure $Q$, Synthon reaction $R$, valid cut set $u$
   Output: Set $H$ of synthon tuples of $R$ that contain $Q$ after assembly
15 Function processCutSet($Q$,$R$,$u$):
16     connector region heuristic for $F(u)$ and $R$
17     for $F_i^* \leftarrow$ labeled forests of $F(u)$ do
18         largest fragment heuristic for $F(u)$ and $R$
19         for $(M_j = (g_j \to s_j)) \leftarrow \Psi(F_i^*, R)$ do
20             if $\forall\ t_i \in g_j\ \exists s' \in M_j(t_i) : t_i$ is subgraph of $s'$ then
21                 add $s_j$ to hits $H$
22             end
23         end
24     end
25 End Function
```

The connector region heuristic

We define the connector region of a synthon as the fragment or set of fragments that we get by considering all atoms and bonds up to a specific distance from the connector, and by replacing all labeled connectors by unlabeled connectors. We denote the graph resulting from this procedure for a synthon s_i by $\gamma(s_i)$. We sort all synthons of each reaction according to their connector region with distance 3 bonds (see Fig. 2.2). We then apply this heuristic on the unlabeled query synthon substructures resulting from a cut set. A synthon reaction can be eliminated, if we do not find an injective mapping η from unlabeled fragments to synthon sets, such that for all unlabeled fragments f holds: in $\eta(f)$ exists a synthon with a connector region that contains the connector region of f as subgraph. We formulate the connector region heuristic as:

$$\exists (M_j : g_j \to s_j) \in \Psi(F_i) : \forall g'_j \in g_j : \gamma(g'_j) \text{ is subgraph of } \gamma(M_j(g'_j))$$

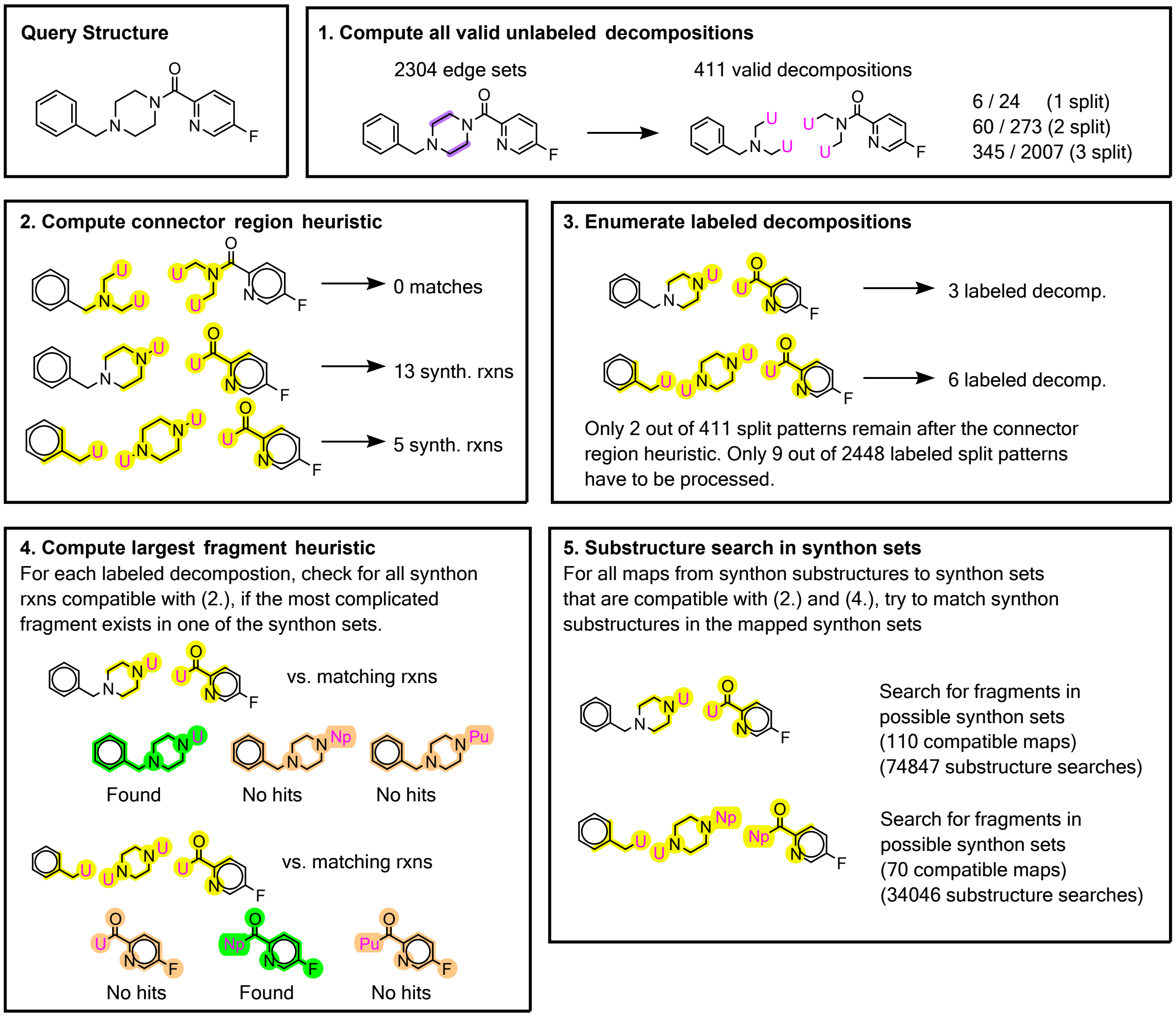
If this condition is not satisfied, we can exclude that the fragments in $F(u)$ could match synthons of the synthon reaction. Because we can run this heuristic already on the unlabeled forest $F(u)$, we can omit a large number of cut sets without ever looking at their enumerated labeled forests. The example shown in Fig. 2 illustrates the efficiency of the connector heuristic, as 409 out of 411 valid decompositions can be eliminated in this way. Please note that we can again use the method for fast superset testing in enumerated libraries to perform the necessary substructure searches.

The largest fragment heuristic

If we find cut sets together with synthon reactions that pass the connector region heuristic based on the unlabeled fragments, we start enumerating all labeled fragments. In a first step, we sort the fragments by their potential to eliminate non-matching synthon reactions. We found that sorting descending by the number of set bits in the fragment fingerprint leads to excellent results. We take the fragment f_L with the most set bits in the fragment fingerprint and check if we find among all fragment fingerprints of labeled synthons any bitsets that are a superset of the fragment fingerprint of f_L. We found that this heuristic works very efficiently to eliminate labeled splits that cannot be matched to any synthon reaction of the synthon space. Please note that we can again use the method for fast superset testing to evaluate this heuristic.

OpenChemLib query features

The OpenChemLib substructure search provides a number of mechanisms to perform advanced substructure search that goes beyond searching for a specific subgraph. These mechanisms are provided via so called query features. There exist broadening (i.e. relaxing) query features on bonds and atoms, as well as narrowing (i.e. constraining) query features on bonds and atoms. Broadening query features include the possibility to match bonds of the query structure to different bond types, or to match atoms of the query structure to multiple atom



Query Structure
1. Compute all valid unlabeled decompositions
2304 edge sets
411 valid decompositions
6 / 24 (1 split)
60 / 273 (2 split)
345 / 2007 (3 split)
2. Compute connector region heuristic
0 matches
13 synth. rxns
5 synth. rxns
3. Enumerate labeled decompositions
3 labeled decomp.
6 labeled decomp.
Only 2 out of 411 split patterns remain after the connector region heuristic. Only 9 out of 2448 labeled split patterns have to be processed.
4. Compute largest fragment heuristic
For each labeled decompostion, check for all synthon rxns compatible with (2.), if the most complicated fragment exists in one of the synthon sets.
vs. matching rxns
Found
No hits
No hits
vs. matching rxns
No hits
Found
No hits
5. Substructure search in synthon sets
For all maps from synthon substructures to synthon sets that are compatible with (2.) and (4.), try to match synthon substructures in the mapped synthon sets
Search for fragments in possible synthon sets (110 compatible maps) (74847 substructure searches)
Search for fragments in possible synthon sets (70 compatible maps) (34046 substructure searches)

Figure 2: An Illustration of the different steps and heuristics used. (1.) This shows an example for a valid decomposition, together with the total number of unlabeled decompositions that arise for this structure. (2.) The subgraphs used for the connector region heuristic are highlighted, for the two decompositions that result in hits, and for one out of the 409 eliminated decompositions. The heuristic eliminates over 99% of the unlabeled decompositions. (3.) For the remaining unlabeled decompositions, all labeled decompositions are enumerated. In this example we assume that we consider synthon reactions with up to 3 different connector vertex types, therefore we get $\binom{1}{3}(1!) = 3$ labeled decomp. for the single split decomposition, and $\binom{2}{3}(2!) = 6$ labeled decomp. for the double split case. (4.) The most complicated fragment (measured by number of set fragment fingerprint bits) is considered first for actual substructure search in the synthon sets (5.) Finally, matching synthons are searched according to the maps that are compatible with all heuristics.

types. Narrowing query features include the possibility to prohibit further substitution on atoms or to require atoms of the query substructure to have a specific number of (unspecified) neighbor atoms.

The OpenChemLib fragment fingerprint is compatible with all of the provided query features and therefore can be used for the prefiltering during substructure search. When using query features for the search in combinatorial libraries, for computing the connector region heuristic we do remove all narrowing query features, as they may incorrectly filter out candidate synthons, e.g. when one of the terminal atoms of the connector region has a query feature that requires it to contain a specific number of neighbors.

Results

Searching the Enamine REAL Space - Some structures from Schmidt et al.[7]

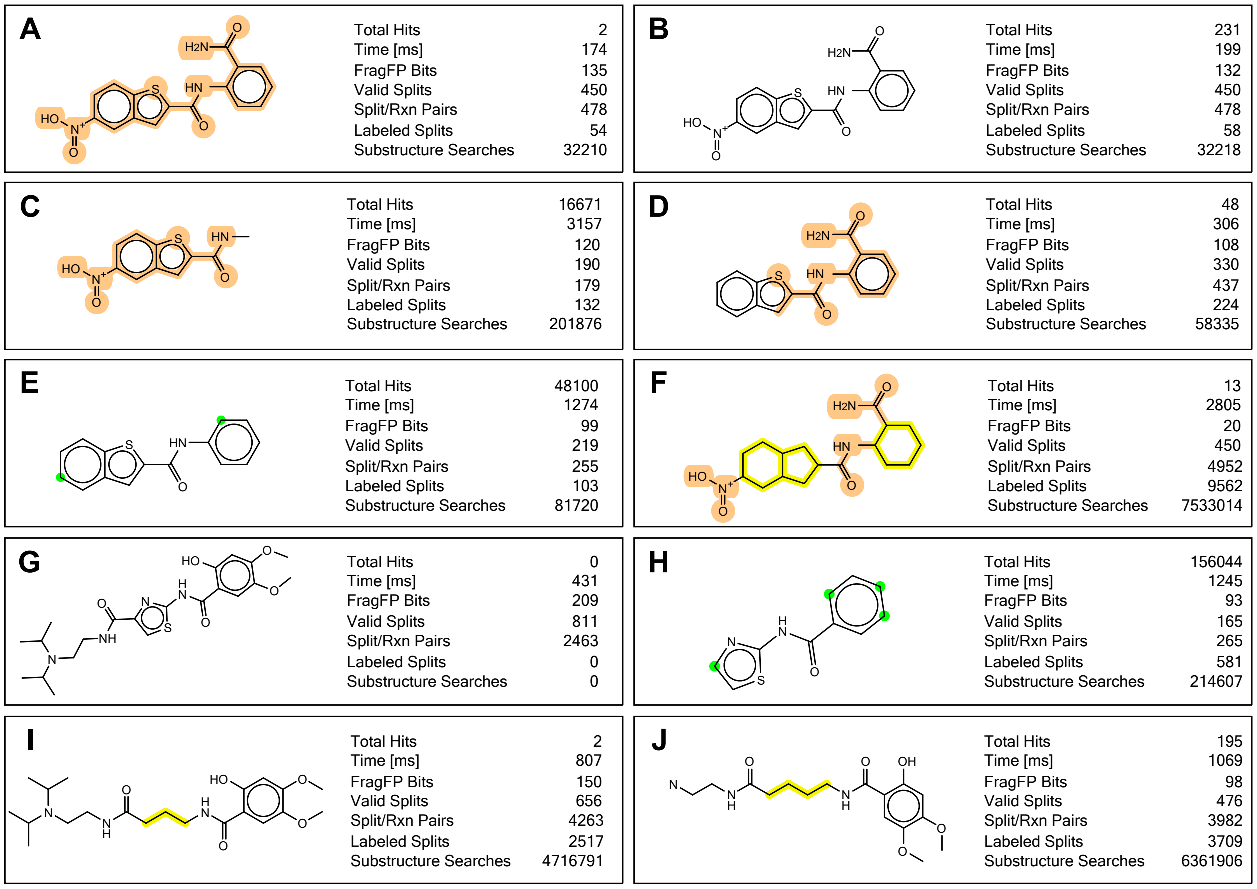
To illustrate the performance of our substructure search algorithm for searches in the Enamine REAL Space we consider structures that were mentioned in a recent publication on computing the IMCS in combinatorial spaces. The Enamine REAL space is the only commercial space, for which we have access to a synthon representation, therefore we cannot state any results for the WuXi AppTec GalaXi[12] or OTAVA CHEMriya[13] spaces. It is difficult to directly compare runtimes for our and their method for two reasons: (i) their method in general solves a much harder problem, and (ii) their method only works under the strong restriction that ring / non-ring edges are exclusively matched to similar edges. Regarding (ii), it is important to stress that our method comes without any restrictions except for that the query substructure must be connected and that it returns the exact solution to the given substructure search problem and not in any way an approximation.

All benchmarks were performed on a 6 core (12 threads) Intel Core i7-7800X at 3.5 GHz with 64 GB of RAM. All time-critical steps of the algorithm are parallelized and we always observed near perfect scaling behavior when running the algorithm on multiple cores. The searches were performed by using parallel computation using 8 threads. Searches in the REAL Space can be performed with less than 4 GB of RAM, so it is feasible to run searches locally on standard desktop computers. We first consider six queries based on the compound G43 (a lead compound for the development of a glycosyltransferase inhibitor) that was considered in their publication (see Fig. 3 A-F). In A, we search if the exact structure exists in the Enamine REAL Space and find two synthon reactions that describe this compound. We observe, that this computation on a single core would take less than a second. In B, we search for all structures containing the structure of G43 as a substructure, without any further constraints. Here, finding the 231 synthon sets that assemble superstructures of this compound again takes less than one second on a single core. In C, we keep the left hand side of the compound fixed and search for all extensions, and in D we keep the right hand side fixed and search for all substitutions of the Benzothiophene. In E, we search for all structures containing the Murcko scaffold and must be further substituted at the two indicated atoms. Computing the 48100 synthon sets that describe matching structures takes around ten seconds on a single core. In F, we illustrate wild card feature matching. We allow any combination of atoms and bonds in all ring structures. we observe an extreme drop in bits of the Fragment FP compared to the other considered structures, due to the fact that a wide variety of different structures can match this search pattern. The low number of Fragment FP bits illustrates that only very few specific substructure features can be deduced for this query. The reason why the search for C takes so long (3 seconds) for this rather simple query is due to the large number of substructure searches that have to be performed for the left hand side of the molecule and that are not filtered by the fingerprint filtering of fragments. This is, because for a large number of decompositions, only the single carbon atom outside of the fixed area has to be matched against all synthons, and therefore a large number of trivial substructure searches have to be performed that cannot be filtered out by fingerprint filtering.

Searching the Enamine REAL space - A small molecule queries benchmark

For testing and benchmarking internally we use a set of 2977 small molecule structures collected from DrugCentral.[14] We consider only compounds containing in between 12 and 32 non-H atoms, because this seems to cover most realistic substructure searches (smaller substructures may lead to excessive amounts of results, for larger compounds it becomes increasingly unlikely to find hits). We further exclude structures containing alkylic chains of more than ten unsubstituted carbon atoms as this for certain molecules leads to rather long runtimes and because we do not consider these searches as very relevant. We ran all tests on a 6 core (12 threads) Intel Core i7-7800X at 3.5 GHz with 64 GB of RAM, and we run the queries parallelized using 8 threads. For each structure we consider two different searches: (a) the search for the specific structure, and (b) the search for structures containing the compound structure as a substructure. The full list of results is included in the supplement and can easily be reproduced using the Hyperspace software package.

For the 2997 small molecule structures, we find 261 exact hits, and 721 substructure hits in the Enamine REAL space. On our 8 threads test setup, 90 % of the queries finished in less than 0.5 seconds (see Fig. 4). There are a few queries that take several seconds. We found two main reasons for longer query times. Structures that contain long alkyl chains are problematic, as they give rise to a large number of decompositions that cannot be rejected by the connector neighborhood heuristic. Simple structures that contain very few functional groups can be problematic, as in certain cases they lead to unspecific synthon substructures, resulting in large number of actual substructure searches due to the fact that fingerprint filtering of synthons is not efficient. However, in these cases it is usually possible to signif-



A

Total Hits	2
Time [ms]	174
FragFP Bits	135
Valid Splits	450
Split/Rxn Pairs	478
Labeled Splits	54
Substructure Searches	32210

B

Total Hits	231
Time [ms]	199
FragFP Bits	132
Valid Splits	450
Split/Rxn Pairs	478
Labeled Splits	58
Substructure Searches	32218

C

Total Hits	16671
Time [ms]	3157
FragFP Bits	120
Valid Splits	190
Split/Rxn Pairs	179
Labeled Splits	132
Substructure Searches	201876

D

Total Hits	48
Time [ms]	306
FragFP Bits	108
Valid Splits	330
Split/Rxn Pairs	437
Labeled Splits	224
Substructure Searches	58335

E

Total Hits	48100
Time [ms]	1274
FragFP Bits	99
Valid Splits	219
Split/Rxn Pairs	255
Labeled Splits	103
Substructure Searches	81720

F

Total Hits	13
Time [ms]	2805
FragFP Bits	20
Valid Splits	450
Split/Rxn Pairs	4952
Labeled Splits	9562
Substructure Searches	7533014

G

Total Hits	0
Time [ms]	431
FragFP Bits	209
Valid Splits	811
Split/Rxn Pairs	2463
Labeled Splits	0
Substructure Searches	0

H

Total Hits	156044
Time [ms]	1245
FragFP Bits	93
Valid Splits	165
Split/Rxn Pairs	265
Labeled Splits	581
Substructure Searches	214607

I

Total Hits	2
Time [ms]	807
FragFP Bits	150
Valid Splits	656
Split/Rxn Pairs	4263
Labeled Splits	2517
Substructure Searches	4716791

J

Total Hits	195
Time [ms]	1069
FragFP Bits	98
Valid Splits	476
Split/Rxn Pairs	3982
Labeled Splits	3709
Substructure Searches	6361906

Figure 3: Ten searches in the Enamine REAL Space. The orange highlighted parts of the structures are fixed, i.e. highlighted atoms are constrained to have no additional neighbors. Yellow highlighted bonds correspond to any bond type (single, double, triple or aromatic), and the yellow highlighted atoms correspond to atom wild cards, i.e. any atom is allowed. Green highlighted atoms indicate that an additional neighbor is required. Total Hits is the number of synthon sets that create matching structures (non-unique). Time [ms] is the time in milliseconds that this search took on our 8 core test setup. FragFP Bits is the number of bits for the query structure in the OpenChemLib fragment fingerprint. Valid splits is the number of splits that have to be considered. Split/Rxn Pairs is the number of pairs of synthon reactions and cut sets that survive the connector region heuristic. Labeled splits is the total number of labeled splits that survived the largest fragment heuristic. Substructure Searches is the total number of substructure searches (labeled split fragments vs. synthons) that were performed (before fingerprint filtering, usually, the very large majority of these substructure searches will be done by fingerprint filtering).

icantly reduce the query time by making the query slightly more specific, e.g. by adding a single additional atom or group, or even by adding certain query features like requiring / excluding substitution at specific atoms.

Conclusion

The algorithm for substructure search in combinatorial libraries improves on existing algorithms and seems to be at least ten to hundred times faster compared to the published results for the only other algorithm that is able to perform substructure search in the Enamine REAL space. The presented algorithms are available as open source in the software package OCL Hyperspace as part of the OpenChemLib. In addition to the algorithms we provide a convenient GUI to perform queries, a simple server / client infrastructure and a plugin for the open-source cheminformatics software DataWarrior[15] to access the functionality.

There are a number of key advantages over alternative algorithms: (i) no constraints on how the synthons can be assembled, (ii) no internal parameters that have to be chosen, the only optional parameter is a stop criterion preventing excessive enumeration of search results (iii) results are returned efficiently as combinatorial hits, meaning that our algorithm can also efficiently return very large numbers (millions) of matching substructures.

Impact on drug discovery and outlook

At Idorsia we see a dramatic impact on drug discovery in multiple ways. Search in commercial combinatorial libraries using this method became a key technology for drug discovery and is used on a daily basis through all phases of drug discovery projects. The configurable

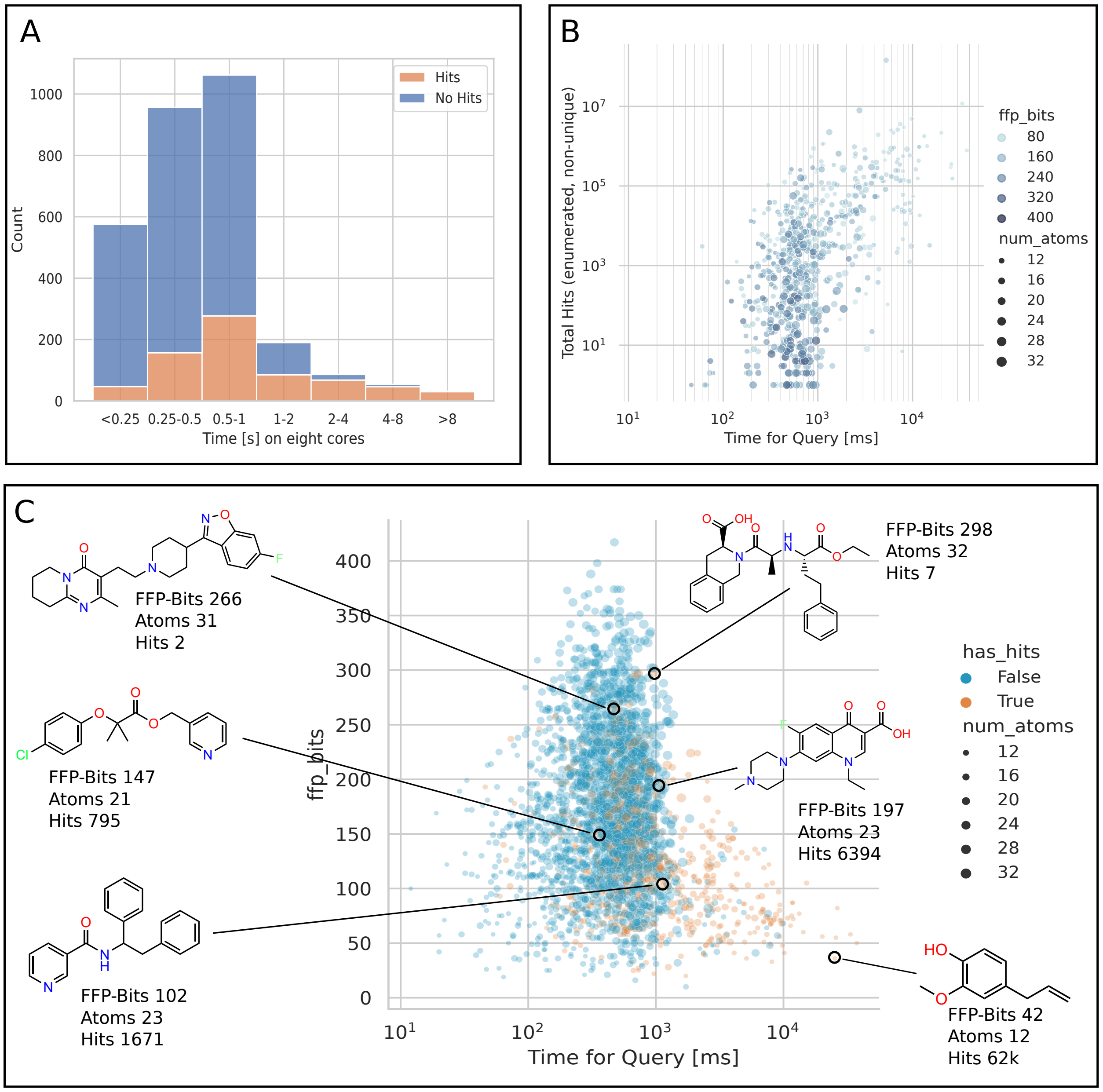


Figure 4: Summary of the results of the benchmark based on the 2997 considered structures from DrugCentral. Please note that results are not for the specific structure, but for the substructure including all superstructures of the query. We considered queries with in between 12 and 32 atoms, and we excluded molecules containing decyl groups (i.e. very long alkylic chains). (A) Overview of query times. (B) Plot of query times versus the number of enumerated hits (without checking for uniqueness of the assembled structures, only based on matching synthons). ffp_bits is the number of set bits in the fragment fingerprint. Please note that this plot only includes queries that returned hits. (C) Plot of query times versus set bits in the fragment fingerprint including some example structures. The color indicates whether the query returned hits.

substructure queries provided by the OpenChemLib allow us, to search for very specific classes of structures, e.g. for replacing specific parts of molecules or for scaffold hopping. This if often used in automated workflows that integrate modeling and prediction of properties. Following up the advent of this technology, we are currently in the process of largely automating the first phases of small molecule drug discovery projects.

The efficiency of this method allows us to search through combinatorial spaces that are much larger than the Enamine REAL space. Therefore, we created combinatorial spaces of structures that are synthetically easily accessible through in-house high-throughput medicinal chemistry via up to three steps.[16] These spaces are several orders of magnitude larger than the Enamine REAL space (depending on the specific setup in between 10^{12} and 10^{15} structures). We are just starting to explore the possibilities of advanced computational methods in cheminformatics and computational chemistry.

Acknowledgement

The authors thank Joel Wahl and Modest von Korff for valuable input and extensive discussions about the topic.

Supporting Information Available

- GitHub repository of the OCL Hyperspace project github.com/Actelion/openchemlib-hyperspace
- CSV file containing results of the small molecules benchmark
- List of OpenChemLib IDCodes that contain the generalized substructures of the fragment fingerprint.

References

(1) Patel, H.; Ihlenfeldt, W.-D.; Judson, P. N.; Moroz, Y. S.; Pevzner, Y.; Peach, M. L.; Delannée, V.; Tarasova, N. I.; Nicklaus, M. C. SAVI, in silico generation of billions of easily synthesizable compounds through expert-system type rules. *Scientific data* **2020**, *7*, 1–14.

(2) Enamine REAL Space and REAL Database. `https://enamine.net/compound-collections/real-compounds/real-space-navigator`, 2022.

(3) Rarey, M.; Stahl, M. Similarity searching in large combinatorial chemistry spaces. *Journal of Computer-Aided Molecular Design* **2001**, *15*, 497–520.

(4) Yu, N.; Bakken, G. A. Efficient exploration of large combinatorial chemistry spaces by monomer-based similarity searching. *Journal of chemical information and modeling* **2009**, *49*, 745–755.

(5) Bellmann, L.; Penner, P.; Rarey, M. Topological Similarity Search in Large Combinatorial Fragment Spaces. *Journal of Chemical Information and Modeling* **2020**, *61*, 238–251.

(6) Hoffmann, T.; Gastreich, M. The next level in chemical space navigation: going far beyond enumerable compound libraries. *Drug discovery today* **2019**, *24*, 1148–1156.

(7) Schmidt, R.; Klein, R.; Rarey, M. Maximum Common Substructure Searching in Combinatorial Make-on-Demand Compound Spaces. *Journal of Chemical Information and Modeling* **2021**,

(8) Ehrlich, H.-C.; Volkamer, A.; Rarey, M. Searching for substructures in fragment spaces. *Journal of chemical information and modeling* **2012**, *52*, 3181–3189.

(9) Ehrlich, H.-C.; Henzler, A. M.; Rarey, M. Searching for recursively defined generic

chemical patterns in nonenumerated fragment spaces. *Journal of chemical information and modeling* **2013**, *53*, 1676–1688.

(10) OpenChemLib. `https://github.com/actelion/openchemlib`, 2022.

(11) Bellmann, L.; Penner, P.; Rarey, M. Connected Subgraph Fingerprints: Representing Molecules Using Exhaustive Subgraph Enumeration. *Journal of chemical information and modeling* **2019**, *59*, 4625–4635.

(12) WuXi GalaXi Space. `https://www.biosolveit.de/software/galaxi`, 2022.

(13) CHEMriya Make-On-Demand Space. `https://www.biosolveit.de/infiniSee#chemriya`, 2022.

(14) Avram, S.; Bologa, C. G.; Holmes, J.; Bocci, G.; Wilson, T. B.; Nguyen, D.-T.; Curpan, R.; Halip, L.; Bora, A.; Yang, J. J., et al. DrugCentral 2021 supports drug discovery and repositioning. *Nucleic acids research* **2021**, *49*, D1160–D1169.

(15) DataWarrior. `https://github.com/thsa/datawarrior`, 2022.

(16) Wahl, J.; Sander, T. Fully Automated Creation of Virtual Chemical Fragment Spaces Using the Open-Source Library OpenChemLib. *Journal of Chemical Information and Modeling* **2022**,