J Comput Aided Mol Des (2016) 30:425–446
DOI 10.1007/s10822-016-9916-1

Ring system-based chemical graph generation for de novo molecular design

Tomoyuki Miyao[1] · Hiromasa Kaneko[1] · Kimito Funatsu[1]

Received: 22 March 2016 / Accepted: 31 May 2016 / Published online: 14 June 2016

Abstract Generating chemical graphs in silico by combining building blocks is important and fundamental in virtual combinatorial chemistry. A premise in this area is that generated structures should be irredundant as well as exhaustive. In this study, we develop structure generation algorithms regarding combining ring systems as well as atom fragments. The proposed algorithms consist of three parts. First, chemical structures are generated through a canonical construction path. During structure generation, ring systems can be treated as reduced graphs having fewer vertices than those in the original ones. Second, diversified structures are generated by a simple rule-based generation algorithm. Third, the number of structures to be generated can be estimated with adequate accuracy without actual exhaustive generation. The proposed algorithms were implemented in structure generator Molgilla. As a practical application, Molgilla generated chemical structures mimicking rosiglitazone in terms of a two dimensional pharmacophore pattern. The strength of the algorithms lies in simplicity and flexibility. Therefore, they may be applied to various computer programs regarding structure generation by combining building blocks.

Keywords Ring systems · Structure generator · Inverse QSPR/QSAR · De novo design

Electronic supplementary material The online version of this article (doi:10.1007/s10822-016-9916-1) contains supplementary material, which is available to authorized users.

✉ Kimito Funatsu
funatsu@chemsys.t.u-tokyo.ac.jp

[1] Department of Chemical System Engineering, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan

Introduction

Generating chemical graphs in silico by combining building blocks is important and fundamental in combinatorial chemistry. A premise in this area is that generated structures should be irredundant as well as exhaustive [1]. Furthermore, a structure generator should be efficient to generate a number of chemical structures when it is employed in de novo molecular design projects, where a large number of solution candidates should be searched. When the expected number of structures to be generated is too large to handle, stochastic or sampling generation is necessary. This situation frequently occurs since generation by combining building blocks easily results in combinatorial explosion. Although the number of structures that can be handled varies from time to time, a methodology to estimate it with adequate accuracy is required. Polya's theorem is a way to estimate the number of structures whose central fragment has some symmetry and multiple substituents [2]. This theorem is applied to estimate the number of isomers of acyclic alkanes C_nH_{2n+2} [3]. MOLGEN-COMB is a well-established software that generates exhaustive combinatorial structures by adding building blocks to a central fragment. The algorithm in it is based on a bijective feature between orbits and double cosets [4, 5]. Polya's theorem, however, cannot estimate the number of combinatorial structures when employing multiple building blocks having symmetry (multiple centers) to the best of authors' knowledge. Therefore, the number of structures to be generated must be estimated based on statistical methods.

Structure generators were originally developed for structure elucidation via mass spectroscopy (MS) and nuclear magnetic resonance (NMR). Since the 1960s, starting from the DENDRAL project, which is the first

expert system implemented in chemistry [6], a number of structure generators have been developed. CHEMICS [7, 8] was developed in the 1980s. It adopted connectivity stacks to enumerate exhaustive structures. Moreover, the software introduced several heuristic rules in constructing connectivity stacks, trying to reduce the number of isomorphism checking operations. MOLGEN [9, 10], which is one of the fastest structure generators today, combines orderly generation method and fast canonical operation by dividing the adjacency matrix to be filled into blocks with an equal feature of vertices. They improved their algorithm with the help of mathematical concept of homomorphism of groups [11]. Faulon extended the graph-equivalent-class algorithm and proposed a structure generation algorithm, without making duplicates, by constructing structures via ideal graphs defined in his paper [12]. He also developed novel canonicalization algorithms with it. In general, structure generators for structure elucidation combine a pre-determined number of fragments that are usually derived from spectroscopies' information. In other words, the goal of such generators is to fill the fragment adjacency matrix, determining connections (bond information).

Another way of employing structure generators is ligand-based de novo molecular design [13]. A structure generator for this purpose should generate molecules satisfying various constraints, such as drug-likeness [14], avoiding generating reactive and unstable structures, and so on. At the same time, molecules having scaffold-hopped [15] features, compared to the known ligands for a target macromolecule, should also be proposed to inspire medicinal chemists. A well-known strategy for structure generation is to combine fragments to mimic combinatorial chemistry. The building blocks are rings and/or heavy atoms or substructures provided by decomposing actual molecules, considering some rules such as RECAP [16], which are related to structure generation strategies. DOGS [17], developed by Schneider's group, uses 25,144 building blocks commercially available and combines them based on known 58 established reaction paths. The objective of DOGS is to generate novel structures similar to a query molecule in terms of potential pharmacophoric points (PPPs) relations. DOGS can use reduced graphs for representing molecules aiming to generate scaffold-hopped structures. FTrees-FS aims to generate exhaustive structures similar to a query molecule in fragment space [18, 19]. Fragment space is defined as fragments to be combined and combination rules that are usually based on reactions. It adopts dynamic programming for fast calculation. This type of generator usually extends structures by adding building blocks to a smaller one and checks the new structures whether they satisfy the pre-determined constraints or not. This is a clear contrast to structure generators for structure elucidation, which usually fill the adjacency matrix of a given set of fragments. When it comes to structure generation in an increased way, the work of Akutsu's group had a breakthrough because they developed a methodology for completely avoiding canonical operation during generation process [20, 21]. Their algorithm for constructing tree-like chemical graphs is based on the rules derived by Nakano et al. [22] and Jordan's theorem, claiming that any tree structure has either so-called *unicentroid* or *bicentroid*. His group recently succeeded in extending their algorithm to monocyclic structures [23]. They also formulated pre-image problems with graph kernels [24] and showed a way to solve inverse quantitative structure–activity relationship or quantitative structure–property relationship (QSPR/QSAR) problems.

Inverse QSPR/QSAR analysis is a promising way for molecular design de novo, since molecular design in silico depends on QSPR/QSAR models, such as estimation of aqueous solubility in virtual screening. We firstly tried to solve the problem with the structure generator that uses an atom as a vertex and employed McKay's canonical construction path method [25]. Although we succeeded in applying the proposed algorithm to a simple case study (i.e. having a small number of heavy atoms), we could not overcome combinatorial explosion for the involved case studies with much flexibility. To reduce the number of structures to be generated, using as many number of descriptors (constraints) as possible and focusing only on structures inside applicability domain (AD) [26] are important. Therefore, we have proposed inverse QSPR/QSAR using ring systems as fragments as well as considering an AD criterion. We, however, did not mention the algorithms for combining fragments in detail in our previous paper [27].

Our contribution in this paper is to describe practical structure generation algorithms by combining ring systems as well as atom fragments for molecular design de novo. In addition to describing the precise algorithm for structure generation used in our previous paper [27], we propose a novel diversity-oriented generation algorithm for structure generation instead of selecting a diversified set of structures (i.e. diversity-oriented sampling). Furthermore, we describe a statistical method of estimating the number of structures to be generated without actual generation. We emphasize on simplicity and practicability of the algorithms rather than on mathematical sophistication. Our approach is simple; growing chemical graphs by repeatedly adding building blocks to smaller ones. The building blocks are both ring systems [28] and atom fragments. During generation process, exhaustiveness and uniqueness of the structures are guaranteed because we substantially employ McKay's canonical construction path method [25]. In the cases where exhaustive generation is not feasible, diversity-oriented generation based on the concept of

pseudo framework [28] is proposed. Then, we introduce how to estimate the number of structures to be generated based on the method presented in Mckay [25]. As an example of practical applications, we generated structures having a pharmacophore profile similar to that of rosiglitazone in terms of 2D-based descriptors. All of the algorithms are simple. Consequently, they can be applied to structure generation by combining fragments.

Methods

Combination of ring systems and atom fragments

Preliminaries

Our strategy for structure generation is to combine ring systems and atom fragments in every possible way. In this paper, every ring system has access points to be extended explicitly. For instance, three arene substitutions with two access points (ortho, meta, and para) are different fragments from one another in this definition. An atom fragment is a single heavy atom with explicit hydrogen atoms. Figure 1 shows examples of ring systems and atom fragments. We call an atomic graph as the graph that is not necessarily completed (i.e. having some unsaturated access points) according to the definition in the paper of Faulon [12]. When every access point is saturated (connected to other fragments), the atomic graph can be regarded as a molecular graph. When combining building blocks, single valence for each element is assumed. Therefore, some functional groups cannot be generated (e.g. nitro and some functional groups with phosphorus). Atom fragments are employed in structure generation. They can be connected

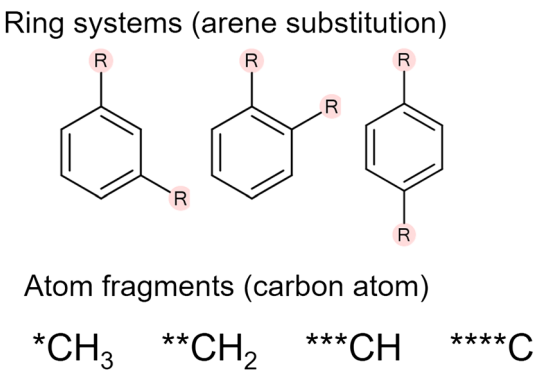

Fig. 1 Examples of ring systems with access points and atom fragments with explicit hydrogen atoms. In the *top row*, three ring systems with different substitution patterns are regarded as different fragments. Rs represent access points (substitution points). In the *bottom row*, carbon atom fragments with explicit hydrogen atoms are depicted. *The remaining degree of the fragment to its valence

to other building blocks in every possible way unless the connection does not violate the valence rule. For example, in Fig. 1, atom fragment ***CH means that the atom fragment has degree three based on its valence. The possible forms of ***CH in a chemical structure are –CH(–)–, –CH=, #CH, where – is a single, = a double, and # is a triple bond.

Structure generation

An atomic graph continues to gain building blocks until it contains a pre-determined number of building blocks, becomes saturated, or violates one of constraints. Treating a building block as a single vertex would be ideal because it saves calculation load. The more vertices a structure has, the more time it takes to conduct graph operations and more storage room is required. Treating a ring system as a single vertex, however, ruins the topology of the ring system. We had to come up with another way to represent a ring system with fewer vertices than those in the original ring system.

Since an automorphism should be considered among vertices having the same color, colored graphs (graphs with colors) are preferable to express atomic graphs. Moreover, the more colors we use for representing a ring system, the fewer automorphisms should be checked (e.g. comparing cyclohexane with piperazine). We assign different colors to different atom fragments. As we described in the subsection of *Preliminaries*, access points characterize ring systems to be combined. Therefore, when considering symmetry during generation process, graphs having fewer vertices than those of the original graphs are introduced based on symmetry including access points. We name these graphs reduced colored graphs. In the field of chemoinformatics, using reduced graphs instead of the actual ones are not a new idea. A graph that is reduced from a molecule can be regarded as an abstract representation of the molecule. For example, Wester et al. [29] introduced the concept of *scaffolds* and *scaffold topology* for molecules along with systemically applied rules for the construction of them. These abstract expressions of a molecule can be effectively applied for database comparison. As a criterion of similarity, different type of reduced graphs is proposed for structure comparison [30]. Gillet et al. analyzed the level of reduction of chemical structures in order to capture the features of bioactive molecules within an activity class [31]. In contrast to the previous definitions of reduced graphs, our definition focuses on the topological symmetry of ring systems. Attaching a building block to an access point of a reduced graph has the same effect as attaching it to the corresponding ring system. Therefore, reduced graphs can be treated in the same manner during structure generation as the corresponding ring systems can. In the

following subsection, we explain some mathematical terms in group and graph theory for our proposed methodologies. Then, we provide a concrete algorithm for structure generation by combining McKay's canonical construction path method [25] with our ideas.

Definition of group homomorphism and isomorphism [11]

Let G_1 be group acting on a set X_1, G_2 be group acting on a set X_2. A pair of maps $\varphi = (\varphi_x, \varphi_g)$ where φ_x: $X_1 \rightarrow X_2$ and φ_g: $G_1 \rightarrow G_2$ is a group homomorphism if φ is compatible with both actions,

i.e. for all $g \in G_1$ and all $x \in X_1$

$$\varphi_x(x^g) = \varphi_x(x)^{\varphi_g(g)}.$$

If both components φ are bijective, φ is an isomorphism.

In short, when two groups are isomorphic, meaning that they are fundamentally identical, we can treat them equally.

Definition of graph isomorphism and automorphism [12]

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two chemical graphs consisting of sets of vertices V_1, V_2 and sets of edges E_1, E_2, a graph isomorphism is a bijection φ: $V_1 \rightarrow V_2$,

s.t. $\forall u_1, u_2 \in V_1$ and $u_1u_2 \in E_1$ then $\varphi(u_1)\varphi(u_2) \in E_2$.

If $G_1 = G_2$, the bijection satisfying the criteria above is an automorphism.

An automorphism is an isomorphism between oneself. Identity mapping is an obvious automorphism preserving a graph structure (edge relation). By applying actions of the automorphism group on a vertex v, the orbit of the vertex is found. The definition of the orbit of $v \in V$ is as follows:

$$o(v) = \{\varphi(v) \in V : \varphi \in Aut(G)\},$$

where $Aut(G)$ is a set of automorphisms of a graph G.

The orbit of a vertex v contains all of the equivalent vertices to v in terms of automorphisms. Roughly speaking, we can treat vertices in an orbit in the same manner.

Finally, we define a reduced colored graph, which has the isomorphic automorphism group as that of the original ring system in terms of access points. The definition of a reduced colored graph is as follows.

Definition of the reduced colored graph for a ring system

Let X be a set of access points in a ring system and G a set of all automorphisms concerning the access points. The reduced colored graph R, having a set of colored vertices Y and a set of automorphisms H, is a graph defining a pair of bijective maps $\varphi = (\varphi_x, \varphi_g)$ among vertices and automorphisms respectively:

$g \in G$ and all $x \in X$,

$$\varphi_x(x^g) = \varphi_x(x)^{\varphi_g(g)},$$

where φ_x: $X \rightarrow Y$ and φ_g: $G \rightarrow H$.

Since every automorphism in the reduced colored graph corresponds to one in the original ring system bijectively, we can treat two graphs equally in terms of access points (in Fig. 2, on the top row).

Every ring system having access points can be replaced with a reduced colored graph. One practical way to find it properly is to use templates. The templates having at least the same number of vertices as that of access points in ring systems are prepared in advance. Then, all of the vertices in a template are mapped to access points in the ring system

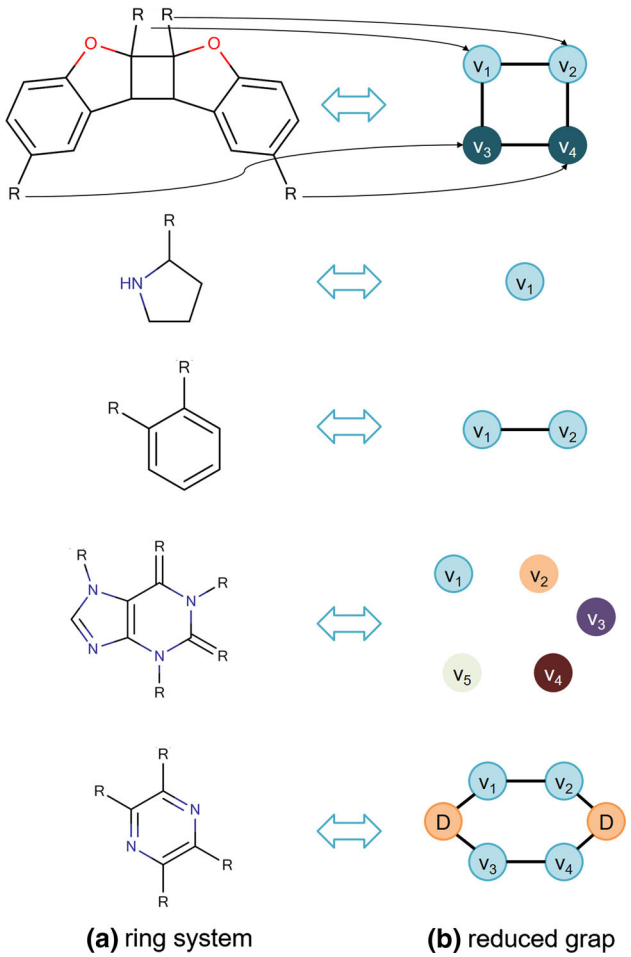

Fig. 2 Reduced *colored graphs* (**b**) and their correspondent ring systems (**a**). Rs represent access points. Ds represent dummy vertices for preserving the symmetry between a ring system and the corresponding reduced graph

so that the map between the two sets of automorphisms becomes bijective. Some templates contain extra vertices other than the corresponding access points in order to hold the same topology as that of the original ring system. We call these vertices dummy vertices. Examples of acquired reduced graphs by applying templates and corresponding ring systems are described in Fig. 2. Pyrazine (the bottom row) with four access points needs two dummy vertices to keep the topology between the ring system and the reduced graph. In this case, translating into reduced graph does not seem to help reduce calculation cost, since the number of heavy atoms in the original structure is the same as that of vertices in the reduced graph. This claim is not entirely true, because several reduced graphs having fewer number of vertices than that of the original access points appear until the growing structure reaches the pyrazine with four access points that is connected to other building blocks. This situation is depicted in Fig. 3. It is noted that coloring should be consistent, among reduced graphs, when the corresponding original ring systems with different access points are isomorphic. It should also be noted that finding out one of the reduced colored graphs does not affect calculation time during generation process, because reduced colored graphs can be prepared before structure generation.

Since ring systems can be replaced with the corresponding reduced graphs, we call a structure consisting of reduced graphs and atom fragments, which are connected to one another in the same way as in the corresponding atomic graph, a contracted graph. Figure 4 shows an example of an atomic graph on the left side and the corresponding contracted graph. Every vertex in the contracted graph (from v_1 to v_6) has its pair access point in the original structure. The important point is that every chemical structure (or atomic graph) can be decomposed into ring systems and other components (i.e. atom fragments) outside ring systems. In other words, every structure

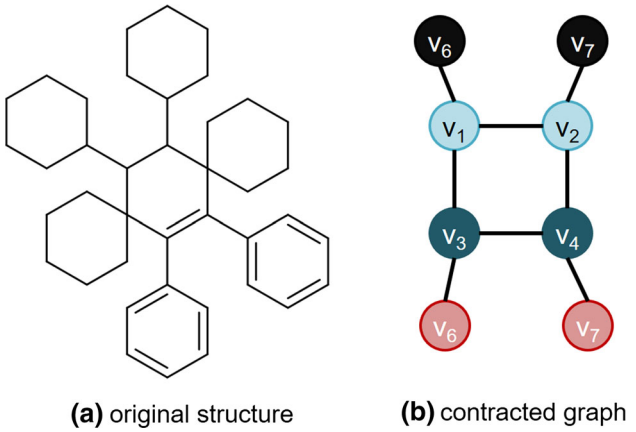

Fig. 4 Original chemical structure consisting of five ring systems (**a**) and the corresponding contracted graph (**b**)

can be constructed by combining ring systems and atom fragments in a tree-like way. Therefore, automorphisms should be considered among the same type of reduced graphs. An edge between ring systems in a chemical graph is bijectively mapped to that in a contracted graph, meaning the structure of the chemical graph is preserved after conversion. Using contracted graphs helps to reduce the number of used vertices during generation process. In general, the number of vertices in reduced graphs equals to the number of access points currently used.

Structure generation algorithm

Structures are constructed by combining building blocks in a tree-like way. Structure generation algorithm is based on McKay's canonical construction path method [25]. The algorithm assures the exhaustiveness and uniqueness of the constructed structures. Moreover, we do not need to check the isomorphism of final generated structures, which allows

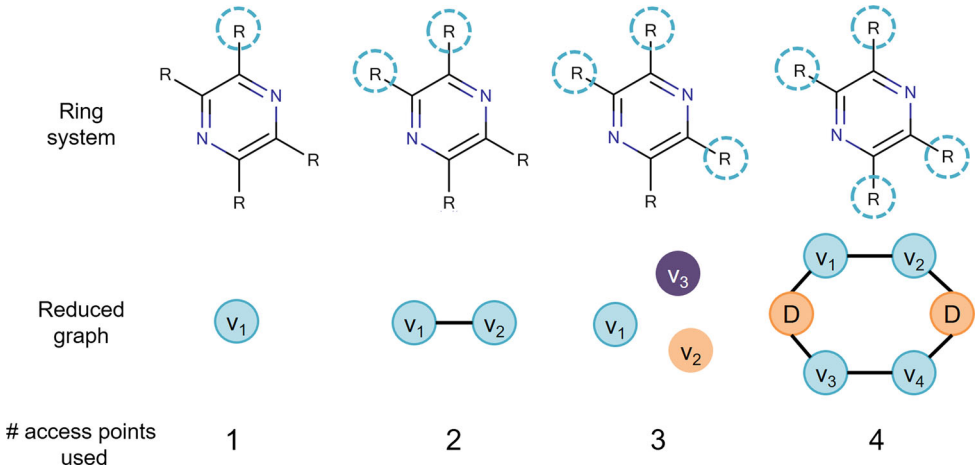

Fig. 3 Reduced *colored graphs* for the ring system of pyrazine with four access points to be saturated. Reduced *colored graphs* (*middle row*) and their ring system connection states (the *top row*) are described. Rs represent access points. *Dashed circles* are the vertices connected to other access points (other building blocks), whose number is described in the *bottom row*

us to generate many structures. The algorithm was successfully applied for generation of certain classes of graphs (e.g. cubic graphs [32]). It is so efficient that a series of GDBs were compiled based on it [33–35]. We modified the algorithm for using contracted graphs. In order not to generate duplicates among the child structures generated from a parent and to check the canonical path to the children, automorphism of both children and a parent should be determined. In Table 1, a pseudo code for growing structures, which is the same meaning of producing children from a parent, is described in the procedure *scan*. First *for* loop and the following selection procedure mean that one of the access points (*ap*) for each orbit is selected as a connection-extended point. The orbits are determined based on the symmetry of input contracted graph X (i.e. automorphism of X ($Aut(X)$)). Unless the selected access point is saturated, the attachment procedure is conducted. The candidate structure X_{new} made by adding a fragment (i.e. reduced graph: *FR*) to X at *ap* is checked by mother function *m*, which determines whether X_{new} should be actually produced from X or not. According to Ref. [25] :(condition M2), *m* is required to determine an orbit of the action $Aut(X_{new})$ on X_{new}. If *FR*, which is a building block attached to X, is inside the orbit, *m* returns *true*, otherwise returns *false*.

In the procedure *scan* in Table 1, we have to determine the orbits of the vertices in contracted graphs. These vertices are unused access points that are not explicitly shown in the contracted graphs. Therefore, a lookup table between used access points orbits and unused access points ones should be made before generation in order to identify the orbit of access points that is to be used.

Structure generation by the proposed algorithm is explained with Fig. 5. Assuming there were four building blocks: methyl, oxygen atom, arene with one access point, and pyrazine with two access points at 2 and 5 carbons. Before applying the algorithm, the order of the fragments has to be assigned for determining which one of the building blocks should be detached after the structure obtain a fragment (i.e. determination of the canonical construction path). CH_3 has the highest priority to be detached and pyrazine with two access points has the lowest in Fig. 5. Therefore, it is not possible to attach CH_3 to arene based on that priority. When combining building blocks, they are considered to be connected in every possible way unless they violate the potential degree of an access point. In this case, oxygen atom has degree two, meaning it can be connected to other building blocks with two single bonds or a double bond. There are four building blocks as roots. When attaching a building block to a growing structure, the symmetry is considered. Only pyrazine on the root row has symmetry, i.e. two access points are equivalent. Therefore, children of the ring system can be produced by attaching building blocks one by one only to one of the two access points (h, i, j and k). This symmetry is also considered when making children of structure k in depth 2. As shown in the canonical construction path checking procedure by the mother function, child structures must identify their parent. In this simple exercise, child structures may form the parent by eliminating the building block having the highest priority at the edge of the structure. When a child structure has two building blocks at the edge having the same highest priority and belonging to the

Table 1 Algorithm of growing contracted graphs by adding reduced graphs and atom fragments. The algorithm was modified from the code in Ref. [25]

line	Pseudo code
1	**procedure** *scan*(*X*: contracted graph, n: integer)
2	**if** *X* is saturated **then**
3	**Output** *X*
4	**Endif**
5	**for** each orbit *A* of the action of *Aut(X)* on *X* **do**
6	select any representative access point $ap \in A$
7	**if** remaining degree of the vertex at *ap* is not zero **then**
8	**for** each reduced graph *FR* to be attached to *X* at *ap* **do**
9	make X_{new} from (*X*,*FR*,*ap*)
10	**if** X $\in$ m(X_{new}) and o(X_{new}) $\leq n$ **then** scan(X_{new}, n) **endif**
11	**endfor**
12	**endif**
13	**endfor**
14	**Endprocedure**

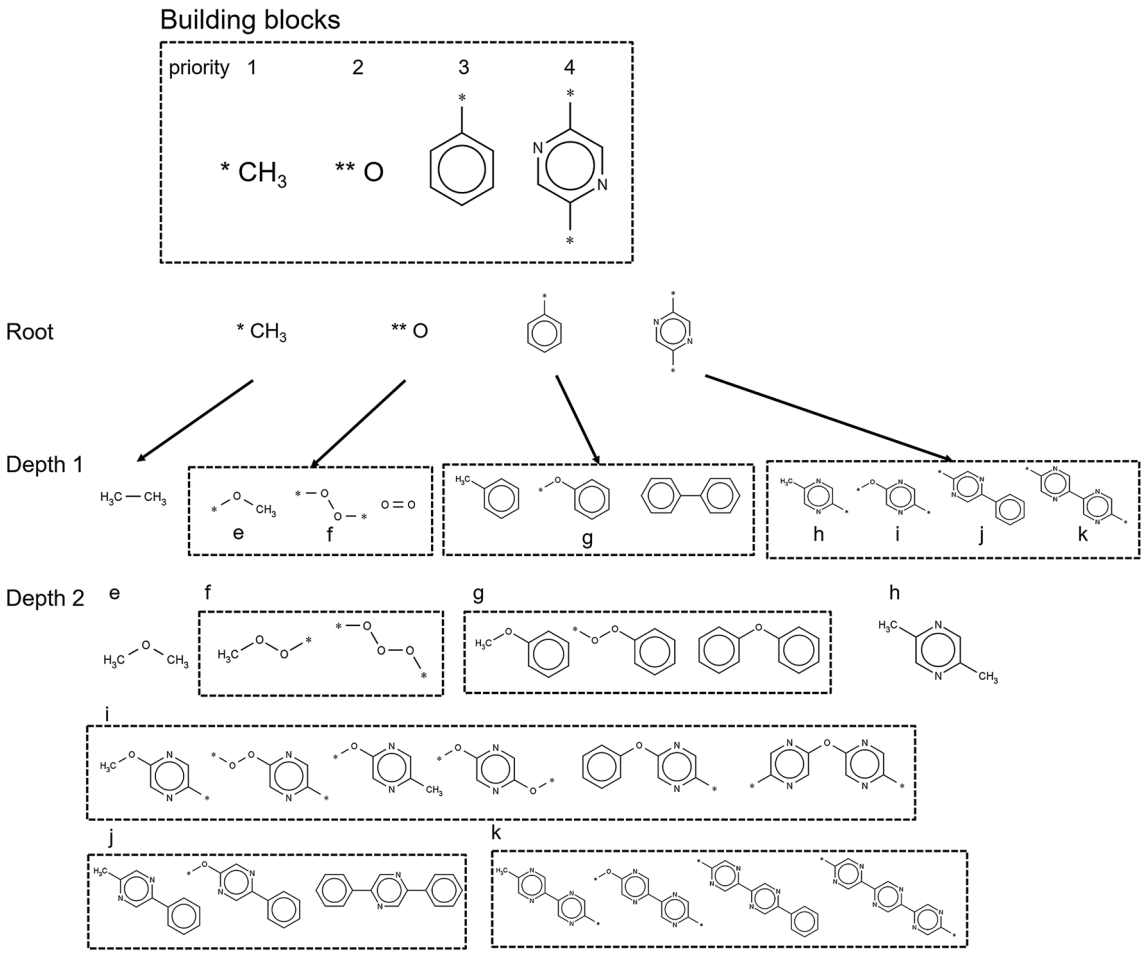

Fig. 5 Generation tree with four building blocks by the proposed algorithm (two ring systems and two atom fragments). *Alphabets* in depth 2 identify their parent structure in depth 1

same orbit (diphenyl ether in Fig. 5), the child structure can be accepted because the *m* function determines an orbit of the action of automorphism group on the child structure. As long as the attached building block is in the orbit, the child structure passes the mother function check. On the other hands, if a child structure has two building blocks having the same highest priority, but they do not belong to the same orbit, either one of them is selected by the *m* function. Although there are no such structures in Fig. 5, when considering appending a methyl group to the leftmost structure in i children (i.e. 2-methoxy-5-methylpyrazine), one of the methyl groups must be selected by the *m* function. In this case, it is possible to determine one of them by, first, canonical labelling of the child structure, then, selecting the orbit containing the vertex having the lowest number.

In actual structure generation, reduced colored graphs and contracted graphs are employed instead of ring systems and atomic graphs. Here, it is sufficient to show examples of them in Fig. 6. As explained in this section, only currently used access points is considered when making reduced graphs. In Fig. 6, two ring systems are translated into the corresponding three reduced graphs when considering currently occupied access points (A in Fig. 6). That pyrazine can connect two building blocks, leading to two different reduced graphs based on the connection during structure generation. By applying the reduced graphs in Fig. 6 to k's children in Fig. 5, the four contracted graphs are formed, which are shown on the bottom row in Fig. 6. Since automorphisms and orbit detection of graphs are considered among vertices having the same color, the translated contracted graphs hold the same topology as the original chemical structures.

Lookup tables for structure generation

Two lookup tables are prepared for speeding up the algorithm in Table 1. One is for selecting a proper reduced graph corresponding to the ring system (line 10 in Table 1).

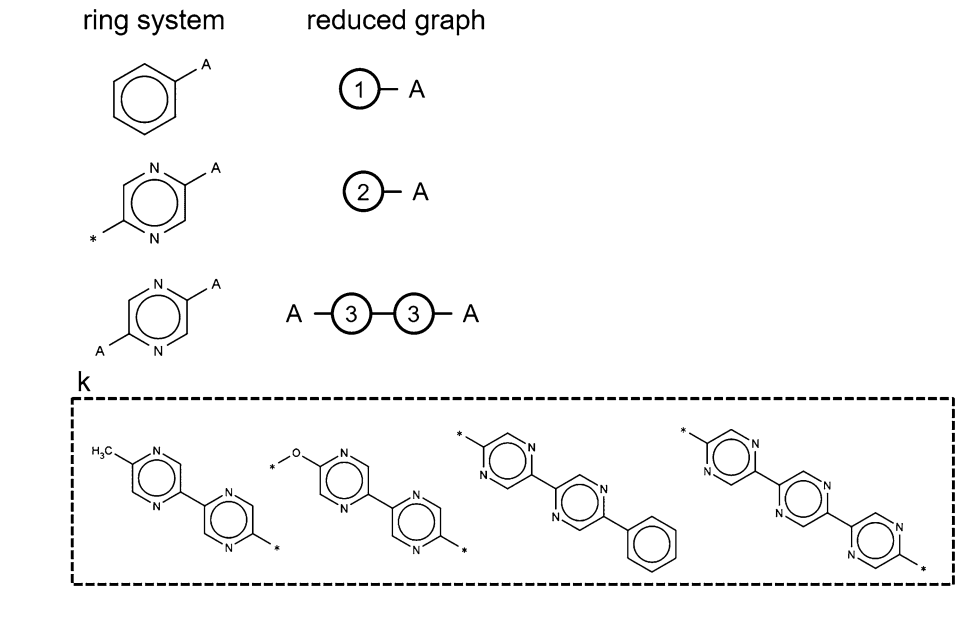

Fig. 6 Structure generation using reduced graphs. Examples correspond to structures in Fig. 5. Ring systems with access points correspond to reduced *colored graphs*. *Colors* are represented as numbers. A is an arbitrary building block that is connected to the ring system. Structures in the *dotted black box* is the same ones as in Fig. 5. The corresponding contracted graphs are depicted in the *bottom row*

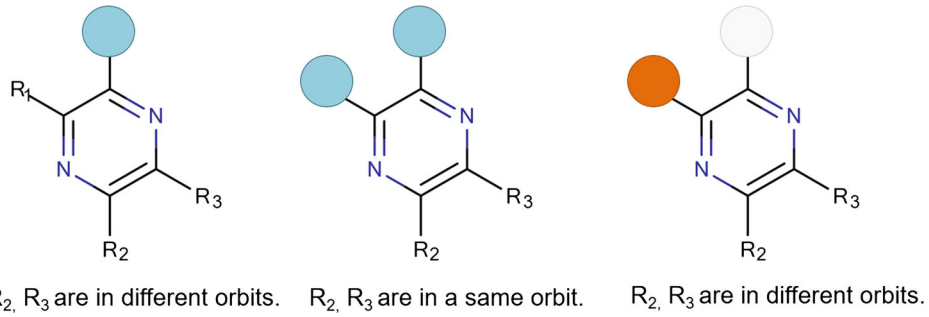

Fig. 7 Orbit changes of access points in pyrazine by attaching building blocks to them. *Circles* are access points already used (connected to other access points) and their *colors* represent orbits of these access points. Rs represent access points. Depending on the *colors* of used access points, the orbits of the remaining ones differ

This lookup table contains a set of access points used and the corresponding reduced graph that has the isomorphic automorphism group to the corresponding ring system in terms of access points (Fig. 3 upper case). Assuming that a ring system has n access points, the number of possible patterns regarding this set is $2^n - 1$. The other lookup table is for selecting the representative access points to which an additional fragment is to be attached (line 5 and 6 in Table 1). For making the latter lookup table, pre-calculated orbits of unused access points corresponding to possibly used patterns are stored. The orbits calculation takes into account the patterns of current access points presuming that they can attach to other access points in other fragments in arbitrary ways. Examples of this pattern preservation are illustrated in Fig. 7. Pyrazine with four access points at 2, 3, 5 and 6. The leftmost structure has one access point used and three unused ones. The unused ones belong to different orbits among one another. The middle and the rightmost ones have two access points used (i.e. R_1 and R_4), but these sets are connected to other sets having different color patterns. This leads to the difference of orbits for the remaining access points (i.e. R_2 and R_3). Assuming that a ring system has 6 access points in the same orbit (e.g. hexane with 6 access points), the number of different coloring patterns with arbitrary number of colors without minding the colors' differences is 203 (for 7 equivalent access points, 877 patterns emerge). Although constructing the second type of lookup table seems to

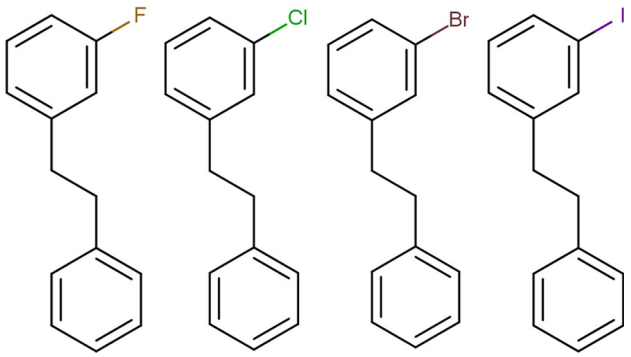

Fig. 8 Examples of combinatorial structures lacking in diversity

hinder our strategy, because of the combinatorial explosion, in most ring systems there is no automorphism except identity map (asymmetry). Thus, only a limited number of ring systems' symmetry information should be stored.

Diversity-oriented generation

Exhaustive structure generation is preferable since there is no oversight of proposed structures [36]. This is, however, not always possible when the structures to be generated are too many to handle. It frequently occurs that most of generated structures have the same scaffold, particularly when using atom fragments as parts for generation (Fig. 8), such as when only one heavy atom in a side chain is different among them. Even worse than that, combinatorial explosion usually occurs if we do not consider any factors except combining fragments. In order to overcome this drawback, stochastic generation [37] and diversity-oriented generation [38] are potential solutions. Herewith, we propose a diversity-oriented generation method. It could also be useful for those who want to make a combinatorial library containing diversified structures by combining ring systems and linkers [28] by orderly generation.

We assume that the generation is conducted recursively by adding either a ring system or an atom fragment. Then, atomic graphs are stored in a stack and the one on the top is used for extension. For applying this algorithm, we have to make initially an order for fragments as follows: the order of every atom fragment is smaller than those of ring systems. In orderly generation, only one structure having the largest number should be generated. In canonical construction path generation, that ordering rule is applicable by designing an adequate mother function; child structures are born by adding, in an orderly manner, a fragment smaller than or equal to any of those in their parent structure. Suppose that the order of an arene substitution with one access point is 2 and that of a methyl group being allowed single connection is 1. Then, (2, 1) is acceptable but (1, 2) is not. This means that to make a toluene by combing these two fragments, a methyl group is added to an arene fragment and not vice versa. Second, all of the atom fragments that can reduce the number of access points (e.g. CH_3, F, Cl etc.) have higher priority than those that cannot. We call this type of atom fragments terminal atom fragments. Once a growing structure (an atomic graph) obtains a terminal atom fragment, the structure preserves current scaffold.

We named the proposed algorithm *pseudo* framework-based generation, since the algorithm cannot distinguish graph based frameworks [28], but can distinguish atom fragment-based frameworks. We illustrate how this algorithm works with a simple depiction in Fig. 9. All of the generated atomic graphs are stored in a stack. Circles are atomic graphs and their colors represent types of *pseudo* frameworks. A *pseudo* framework of a growing structure is updated only when the additional fragment appended is a ring system. Since structure 4 obtains an extra ring system, it becomes the structure with a different *pseudo* framework from that of 1. As we described above, structures that are produced before 4 have the same pseudo framework as that of 1. Once one of the descendants of 1 becomes a molecular graph (6: saturation of the graph), a flag that prohibits using terminate atom fragments for the pseudo framework is switched on. The important part of this algorithm is that we do not have to memorize which frameworks have already been produced. Therefore, introducing this algorithm does not sacrifice generation speed.

Stochastic generation

Estimation of the number of structures to be generated before conducting actual generation is important. Otherwise we have to wait for structure generation to be completed for an uncertain period of time, or we might create too many structures to handle. According McKay [25], we can estimate the number of structures to be exhaustively generated by making use of stochastic pruning with specific probabilities. These probabilities are assigned to each generation, p_0, p_1, etc. (0 for the roots, 1 for the children of the roots, etc.). Unlike ref. 25, we may generate structures having various numbers of fragments combined. So, the estimation of the number of structures to be generated should be

$$E[N] = \sum_{i=1}^{depth} N(i) \prod_{j=1}^{i} \left(1 - p_j\right)^{-1}, \quad (1)$$

where E[N] is the expected number of structures, *depth* is the depth of the generation tree (i.e. the upper number of fragments combined), p_j is the pruning probability at the

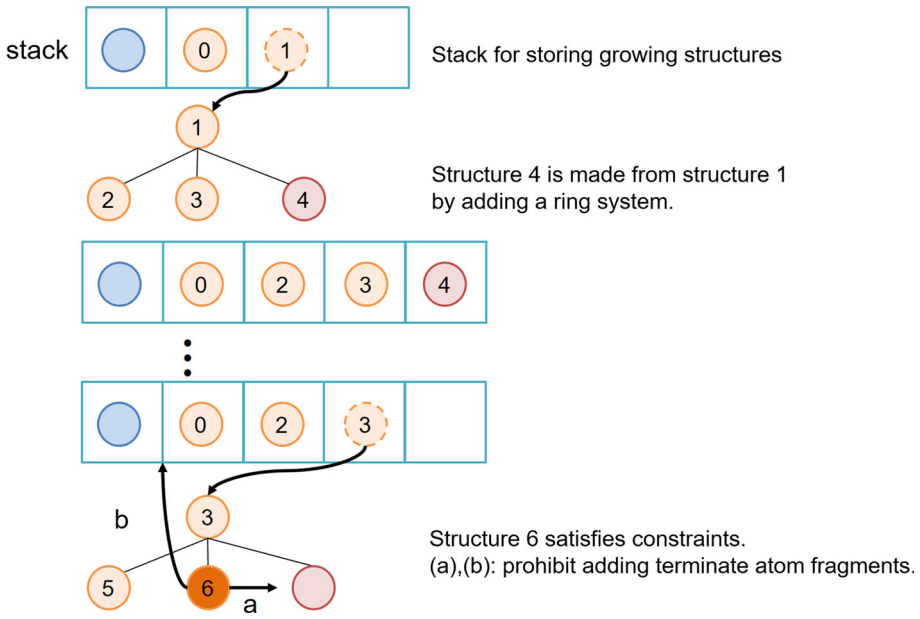

Fig. 9 Schematics on how to generate structures with *pseudo* framework-based generation. Stack is for storing atomic graphs. The structure on top of it is selected as a parent structure. Once a chemical graph is completed, the program is not allowed to select terminal atom fragments for extension from the structures having the same *pseudo* framework (*a*, *b*). Detailed explanation is on the main body

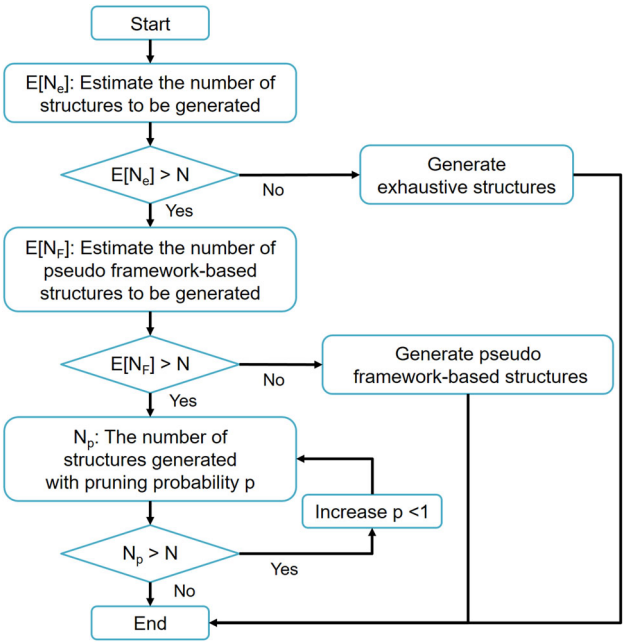

Fig. 10 Proposed structure generation workflow for generating structures less than N. Constraints for the generation as well as the upper number of fragments to be combined are also input

depth j and N(*i*) is the number of sampled structures at depth *i*. We have to generate structures anyway to complete this estimation procedure.

We propose our structure generation strategy for generating a feasible number of structures in Fig. 10. As mentioned on the caption of Fig. 10, this workflow enables generation, where the total number of generated structures is less than *N* by applying various techniques mentioned in this whole article.

Implementations and case studies

We have built a structure generator having the functions mentioned in the *methods* section in C++: generation with contracted graphs, diversity-oriented generation with *pseudo* framework-based generation, and stochastic generation. The structure generator for de novo design is named **Molgilla**, inspired by Godzilla, a Japanese famous giant monster in the film world. In the current version of **Molgilla**, 51 descriptors were implemented. Parallel computing is acceptable by simply dividing root structures (building blocks) into several threads. The performance of the structure generation was tested on Windows 10 PC with 3.33 GHz Intel Xeon CPU and 16 GB RAM.

Reduced colored graphs

We examined ring systems obtained from molecules in the ChEMBL 20 database [39]. To acquire ring systems from molecules, we used an in house program written in C++ with RDKit libraries [40]. There were 1,456,020 molecular records and every record was passed through the node *RDKit from Molecule* in KNIME [41] before decomposition. 1,455,708 structures remained and were input to the program. The following constraints were used during decomposition for selecting appropriate ring systems: maximum number of heavy atoms in a ring system is 30,

Table 2 Pseudo codes for constructing a reduced graph matching a ring system by applying templates (recursive procedure for each ring in a spiro ring system)

line	pseudo codes
1	**procedure** *Assign Reduced Graph*(*R:ring system, RG: reduced graph, SI: Current Spiro index*)
2	**if** *SI* is the number of spiro of *R* **then**
3	**if** *R* and *RG* have the same set of automorphism **then**
4	**return** *true*
5	**else**
6	**return** *false*
7	**endif**
8	**else**
9	*RS* = the part of *R* corresponding *SI*
10	*nAps* = the number of access points of *RS*
11	**Case** based on nAps
12	**Case** == 1
13	append a dot template to *RG* and color *RG* based on orbits of the access points of *R*
14	**if** *Assign Reduced Graph*(*R*, *RG*, *SI*+1) **then**
15	**return** *true*
16	**endif**
17	**Case** == 2
18	append a line template to *RG* and color *RG*
19	**if** *Assign Reduced Graph*(*R*, *RG*, *SI*+1) **then**
20	**return** *true*
21	**endif**
22	Case == 3
23	append a triangle template to *RG* and color *RG*
24	**if** *Assign Reduced Graph*(*R*, *RG*, *SI*+1) **then**
25	return *true*
26	**else**
27	withdrew the triangle template from *RG*
28	append a benzene like template *to RG* and color *RG*
29	**if** *Assign Reduced Graph*(*R*, *RG*, *SI*+1) **then**
30	**return** *true*
31	**endif**
32	**endif**
33	**Case** == 4
34	append a square template to *RG* and color *RG*
35	**if** *Assign Reduced Graph*(*R*, *RG*, *SI*+1) **then**
36	**return** *true*

Table 2 continued

line	pseudo codes
37	**else**
38	withdrew the square template from *RG*
39	append the template of a hexagonal template with two dummy vertices *to RG* and color *RG*
40	**if** *Assign Reduced Graph*(*R*, *RG*, *SI*+1) **then**
41	**return** *true*
42	**endif**
43	Case == 5
44	append a pentagonal template to *RG* and color *RG*
45	**if** *Assign Reduced Graph*(*R*, *RG*, *SI*+1) **then**
46	**return** *true*
47	**endif**
48	**Case** == 6
49	append a hexagonal template to *RG* and color *RG*
50	**if** *Assign Reduced Graph*(*R*, *RG*, *SI*+1) **then**
51	**return** *true*
52	**endif**
53	**Case** == 7
54	append a heptagonal template to *RG* and color *RG*
55	**if** *Assign Reduced Graph*(*R*, *RG*, *SI*+1) **then**
56	**return** *true*
57	**endif**
58	**Default**
59	**return** *Not found proper RG*
60	**EndCase**
61	**Endif**
62	**Endprocedure**

which is an adequate number for searching parts for de novo design based on the literature [42], maximum number of access points in a ring system is 7, which means that macrocyclic ring systems are likely to be omitted, and ring systems consist only of hydrogen, carbon, nitrogen, oxygen and sulfur atoms without formal charge, where the valence of sulfur atoms was two for further analysis. Unique ring systems were searched as a form of ring systems preserving original substitution pattern (access point pattern). As a result, 55,654 unique ring systems with access points were obtained.

We first made templates that are parts for constructing reduced graphs, then constructed reduced graphs corresponding to ring systems by combining these templates in a simple rule. Pseudo codes for this operation and templates used for this case study are described in Table 2 and Fig. 11, respectively. We counted the number of access points used and orbits of access points for each ring in a spiro and treated the ring independently. Spiro is a ring system and, therefore, should be treated correctly. Since rings connecting at a spiroatom can freely move without affecting each other in terms of topology, each template is assigned to each ring in a spiro and once one reduced graph is constructed, automorphism comparison is conducted. When the two sets of all automorphisms are different from each other, meaning the reduced graph is not a proper one,

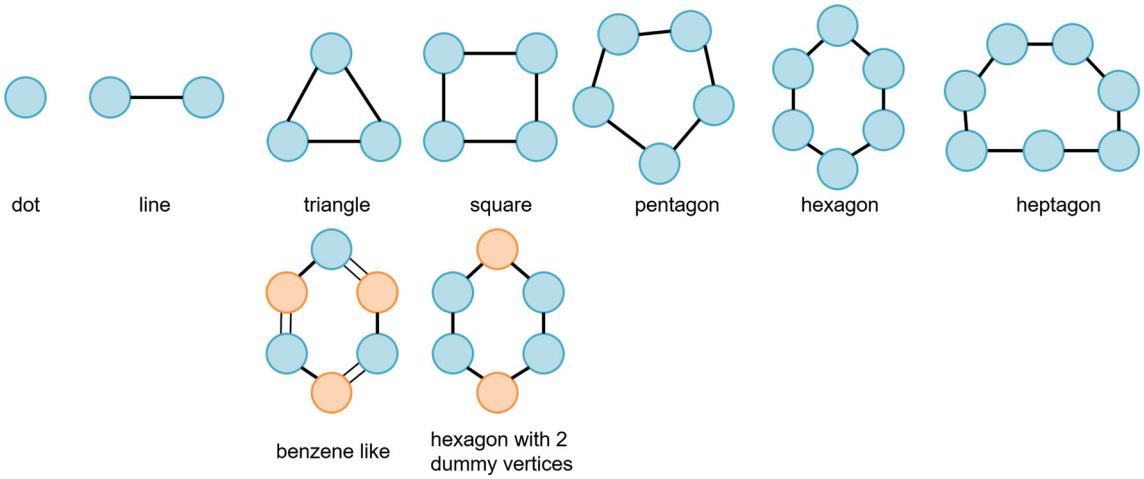

Fig. 11 Set of templates for representing reduced graphs. These templates correspond with those in Table 2. Orange vertices in benzene like and hexagon with 2 dummy vertices are dummy vertices in order to show some topologies

the template that has just been attached is withdrawn and a next template is assigned in a recursive manner. There were 9 templates used for constructing reduced graphs: a dot structure for a single access point, a line structure for two access points, a triangle structure or a kekule benzene-like structure for three access points, a square structure and a hexagonal structure with two dummy vertices assigned across from each other for four access points, a pentagonal structure for five access points, a hexagonal structure for six access points, and a heptagonal structure for seven access points. How the proposed algorithm can find a reduced graph corresponding with a ring system is explained with structures in Fig. 3 (i.e. pyrazines). As for the leftmost pyrazine with one access point in Fig. 3, *RS* in Table 2 on line 9 is the pyrazine itself because it does not form a spiro structure. Therefore, *nAps* on line 10 is one, leading to make a dot structure as a corresponding reduced graph. After calling *Assign Reduced Graph* recursively on line 14, two groups of automorphisms of the reduced graph and the ring system are compared. It results in the dot is a corresponding reduced graph to the pyrazine since only an identical mapping exists. On the other hand, for the pyrazine with four access points at 2, 3, 5 and 6 (the rightmost one in Fig. 3), *RS* on line 9 in Table 2 is the pyrazine itself since it does not have spiro atoms, either. For that pyrazine case, *nAps* on line 10 is four, leading to make a square structure as a corresponding reduced graph. All vertices of the square structure have an identical color since all of the access points of that pyrazine belong to the same orbit. After calling *Assign Reduced Graph* recursively on line 35, the groups of automorphisms of the square structure with a single color and that of the pyrazine are compared in terms of access points. Since two groups of automorphisms are not identical, the function returns *false*, leading to the elimination of the added template. Next, a hexagonal template with two dummy vertices becomes a candidate as a reduced graph (line 39) for that pyrazine. After *Assign Reduced Graph* is recursively called on line 40, the candidate reduced graph is checked whether it has the same automorphism group as that of the pyrazine (line 3) in terms of access points. Finally, the algorithm identifies the reduced graph corresponding to that pyrazine.

As results of applying the algorithm in Table 2, 55,620 ring systems could find the corresponding reduced graphs. Only 34 ring systems did not find any corresponding reduced graphs with the proposed template-based algorithm (Fig. 12). A couple of ring systems failing to find reduced graphs are listed in Fig. 12 along the adequate ones. The cause of the failure is most likely due to the fact that the current set of templates does consider a specific order of labeling to corresponding reduced graphs. Numbering information is stored as all of the vertices are on a single ring. Therefore, ring systems that do not determine a cycle on which all of the access points are located could not find proper reduced graphs (e.g. 2, 3, 5, 8 and 11 in Fig. 12, 11 in Fig. 13). Another reason is the limited variety of templates such as 1 in Figs. 12 and 13. A reduced graph for 1 is tetrahedral where 4 vertices correspond to four access points in 1. This limitation could easily be overcome by adding a tetrahedral template to the template set. Another interesting case is 26 in Figs. 12 and 13. Assuming that the spiro structure used four access points, two of them are in the center cyclobutane and each furan has one equivalent access point. The corresponding reduced graph for it, thus, is a square with two different colors. Although we failed to assign proper reduced graphs to some ring systems, with the current simple rule for assigning reduced graphs, almost all of the ring systems in

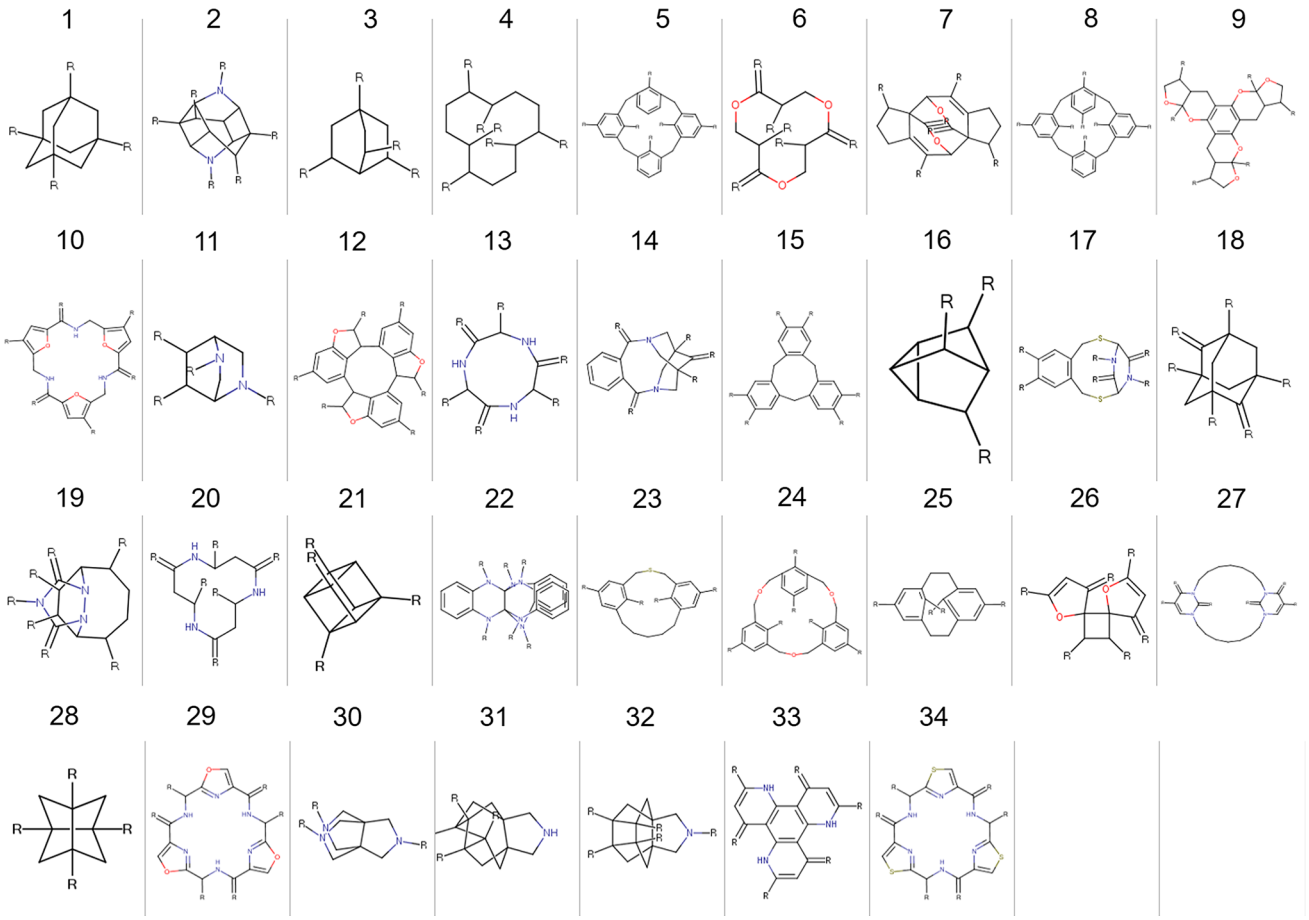

Fig. 12 34 ring systems which could not be assigned to reduced graphs with the current rule mentioned in the body of the text and in Table 2

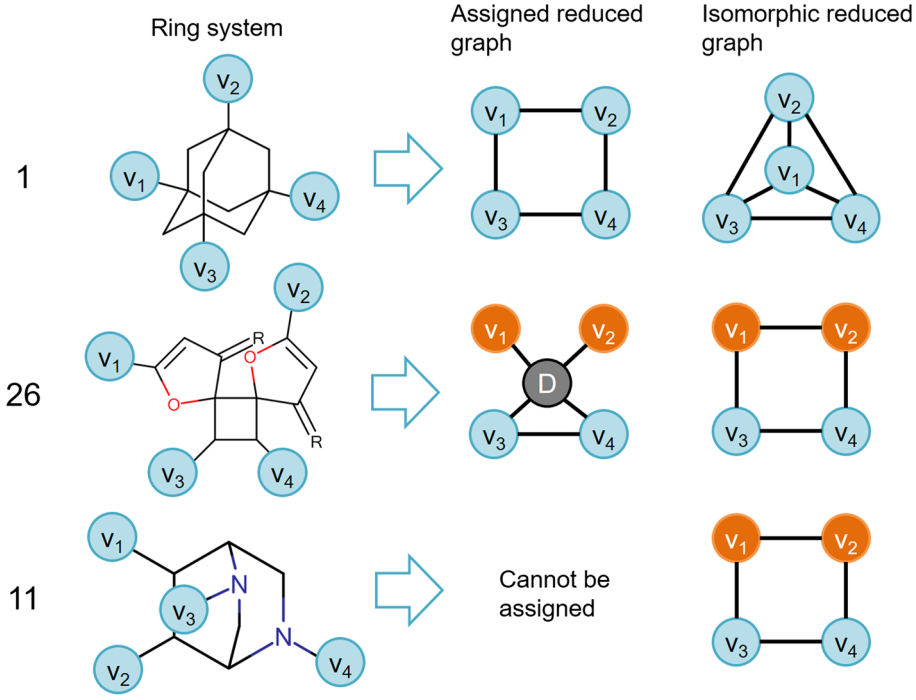

Fig. 13 Examples of ring systems and corresponding reduced graphs that failed to find isomorphic reduced graphs. The *leftmost numbers* represent the structures in Fig. 12. Assigned reduced graphs are the ones that were automatically selected with the algorithm mentioned in the body of the text. Isomorphic reduced graphs are the actual reduced graphs, corresponding to ring systems in terms of access points on the *2nd left column*

ChEMBL with specific constraints could find their partners. Without using the ring systems that are categorized as not-found reduced graph, generation can be conducted based on our proposed algorithm.

Generation efficiency

In order to measure how efficient our proposed generation algorithm becomes, we compared generation speed of Molgilla with that of a simple fragment-combined-based structure generator [43]. The structure generator is implemented in the chemoinformatics software of Chemish ver. 5.09 [44]. The simple fragment-combined-based generator recursively combines building blocks to make a structure until the growing structure is saturated. The generator exhaustively combines building blocks. After the generation procedure is over, canonicalization and elimination of duplicate structures are conducted in order to select unique structures. Since the generator treats a ring system (i.e. a building block) as a chemical graph, we can infer how much the proposed algorithm contributes to the increment of generation speed.

10 ring systems and 13 atom fragments were selected as building blocks. Atom fragments are CH_3, CH_2, CH, C, NH_2, NH, N, OH, O, F, Cl, Br, and I. The employed ring systems are shown in Fig. 14. The number of threads in Molgilla was set one because the simple fragment-combined-based generator is not a multi-threaded program.

The results of the structure generation are on the Fig. 15. Apparently the structure generation by our proposed algorithm surpassed that by the simple fragment-combined-based generator. The numbers of generated structures (number of building blocks combined) are 100 (2), 812 (3), 8037 (4), 116,559 (5), 1,995,641 (6), 33,674,221 (7), and 566,840,430 (8). The simple fragment-combined-based generator could not generate structures by combining more than 6 fragments exhaustively. Calculation time against the number of generated structures looks linear in both cases. It takes 1.39×10^{-3} per structure for the simple fragment-combined-based generator, whereas for our proposed algorithm (i.e. Molgilla), it takes only 3.83×10^{-6} for generating a structure. We cannot conclude that treating ring systems as reduced graphs is solely the cause of the enhancement of Molgilla, because it also

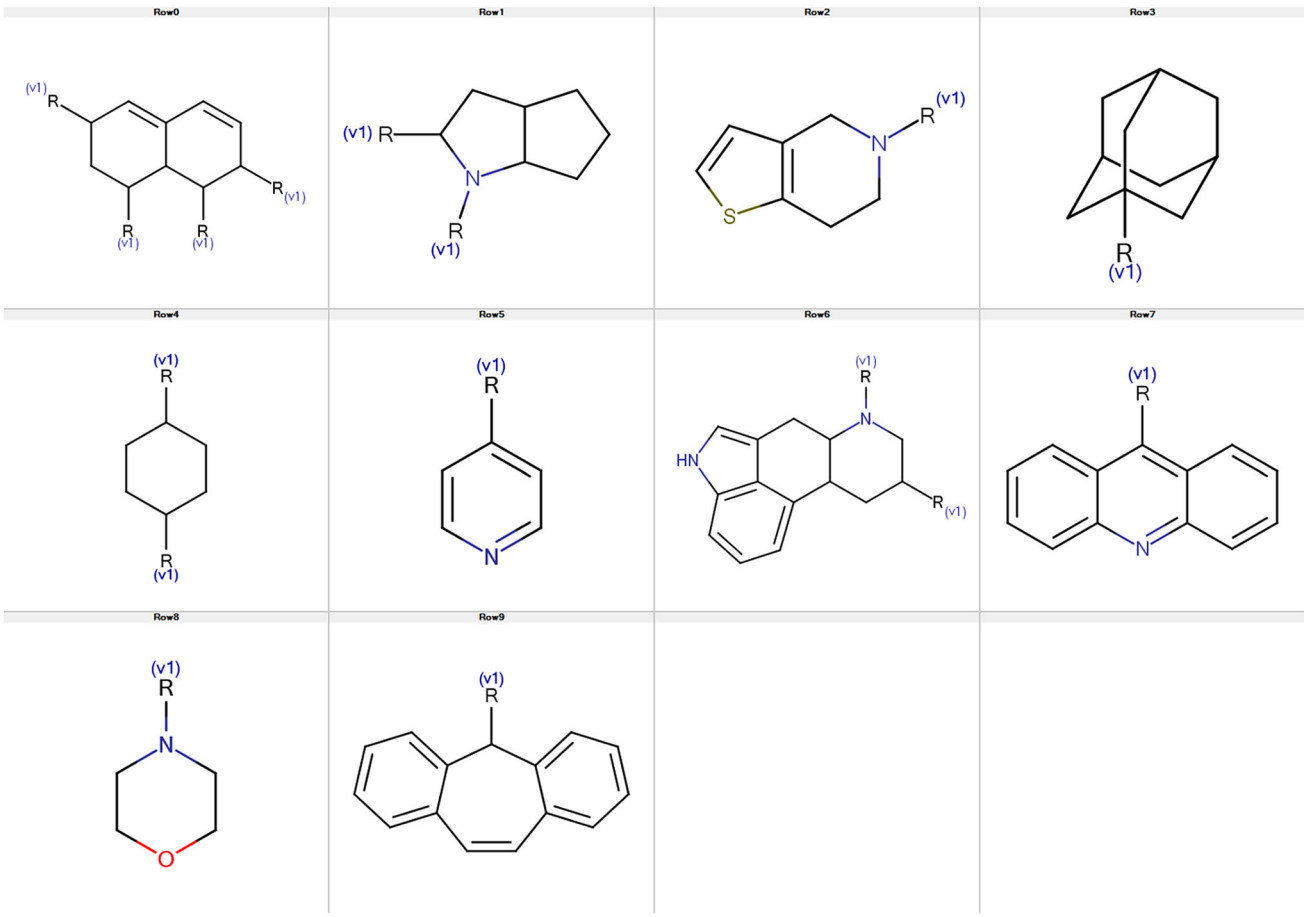

Fig. 14 10 ring systems used for calculation speed test

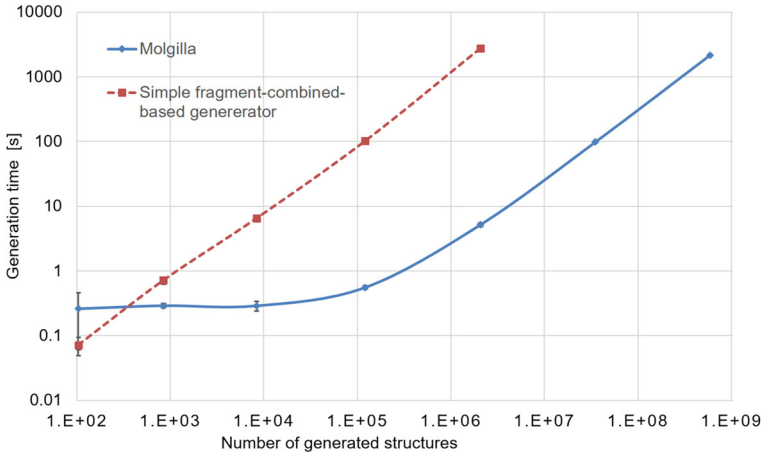

Fig. 15 Calculation speed comparison between Molgilla and the simple fragment-combined-based generator. The *error bar* is based on the three times standard deviation of 5 trials. *Every dot* corresponds to the number of building blocks combined, from 2 to 6 for Chemish, and from 2 to 8 for Molgilla

uses the canonical construction path method. Our proposed algorithm excelled the conventional simple algorithm for fragment-combined-based structure generation.

Diversity-oriented generation

We compared two structure generation algorithms considering a single ring system, one was exhaustive and the other was based on a pseudo framework. The elements used for this work are arene with one access point, as a ring system, and C, N, O, F, Cl, Br, and I atom fragments with explicit hydrogen atoms. We analyzed how the structures generated using the pseudo framework-based generation represented the exhaustive structures. In both cases, we assumed the generation of ring contained structures only (i.e. not generating acyclic ones), combining from two to seven fragments, and assigning some restriction rules, such as avoiding generating aldehydes [45, 46]. Some of the substructures in the Ref. [46] were implemented as a taboo list. The implemented list is as follows: nonC–nonC, =C=, C=N, C(=O)X, CH_2X, CH(=O), aliphatic-C(=O)–aliphatic, where nonC is an atom without carbon, X is a halogen, aliphatic is a non-aromatic atom, – is a single and = is a double bond. Furthermore, the generator was not allowed to combine ring systems directly.

The number of generated structures from exhaustive generation were 21,249 whereas that from pseudo framework-based generation were 624. Both sets of structures are in SDfiles (SI1.sdf and SI2.sdf) as supplementary material.

First, we examined scaffolds in both structure pools. Scaffolds were identified by means of the *Murcko Scaffold* node with default settings by Indigo [47] on the KNIME [41] platform. The scaffold is based on molecular framework defined by Bemis and Murcko [28]. Hence, investigating the molecular frameworks of the generated structures was expected to validate the proposed *pseudo* framework-based structure generation algorithm. In exhaustive generation dataset, 434 unique frameworks were identified, whereas 416 frameworks in pseudo framework-based one. 312 frameworks correspond to a single structure for exhaustive generation, whereas 320 frameworks for *pseudo* framework generation. The most frequently occurred frameworks (top 5) were shown in Table 3. As expected, a large number of structures in the exhaustive structure pool have an arene as their framework

Table 3 Frequently appeared molecular scaffolds in both structure pools

Exhaustive pool[a]		Pseudo framework-based pool[b]	
Framework (SMILES)	Count	Framework (SMILES)	Count
c1ccccc1	15,798	C(C=Cc1ccccc1)c1ccccc1	7
c1ccc(Cc2ccccc2)cc1	959	C(Cc1ccccc1)Nc1ccccc1	7
c1ccc(CCc2ccccc2)cc1	409	N(C=Cc1ccccc1)c1ccccc1	7
c1ccc(Nc2ccccc2)cc1	337	C(C=CNc1ccccc1)c1ccccc1	5
c1ccc(CNc2ccccc2)cc1	268	C(CNc1ccccc1)Cc1ccccc1	5

[a] Dataset consisting of 21,249 structures generated by combining arenes and atom fragments in every possible way

[b] Dataset consisting of 624 structures generated by pseudo framework-based generation. Conditions for structure generation are described in the main body

(15,798/21,249). The proposed algorithm avoids generating every structure except one for this framework, leading to the number of generated structures difference between the two structure pools. It should be noted that the frameworks that could not be generated by pseudo framework-based algorithm are all have linkers [28] double bonded to carbon or oxygen atoms (18 frameworks). Frameworks generated by the Indigo node on the KNIME platform distinguishes linkers containing double-bonded to them. Since the proposed algorithm is based on atom fragments, these linkers are regarded as the same pseudo-framework in the proposed algorithm.

In order to investigate the diversity of the structures with pseudo framework-based structure generation, we compared similarity among k nearest samples in both databases. An underlying idea of this analysis is to test whether structures from pseudo framework-based generation could represent the exhaustively generated structures. When looking into a single structure in both databases, the distances (dissimilarity) between surrounded structures and the structure we are focusing on can be a measurement of how sparse the databases are. A structure in a diversified database has longer distance to its neighbors than one in a non-diversified database. An ideal diversified dataset was constructed as follows; exhaustive structures were firstly generated, and then a sampling method was applied to choose smaller number of molecules. To mimic this ideal procedure, we sampled 624 molecules from 21,249 exhaustive ones with a diversity-oriented sampling method. MACCS key [48] consisting of 166 bits was used as a fingerprint of structures. Diversified sampling was conducted by the *RDKit Diversity Picker* [40] node on the KNIME [41] platform. The criterion of sampling was MaxMin [49]. In Fig. 16, averaged Tanimoto similarities among the k-nearest neighbor structures are plotted against k (i.e. the number of neighbors). As we expected, the line for pseudo framework-based generation is between the exhaustive one and diversified one (between 1 and 623). Averaged Tanimoto similarity in exhaustive structure pool converged to 0.282 when all the molecules in the pool were used for calculation. Figure 16 implies how much diversity the structures generated by the proposed strategy can acquire. It is just between the ideal and the worst case. Furthermore, we visualized how the structures generated by pseudo framework-based generation in the fingerprint space were distributed. The map was created with the data of exhaustive structure pool by multidimensional scaling (MDS). For MDS, the distance between the *i*th sample and the *j*th sample is (1 − Tanimoto similarity between *i*th and *j*th samples). The contribution of the axis 1 is 14.8 % and for the axis 2 is 11.4 % in terms of eigenvalues (Fig. 17). Black squares are the 624 structures from pseudo framework-based generation, whereas red circles are the ones from exhaustive generation. Some of the representative structures from the exhaustive generation pool are picked in Fig. 17 with their coordinates. Around those representatives on the map, there are no structures from pseudo framework-based generation. This is because those three structures shown in Fig. 17 have the same pseudo framework, fluorobenzene. Since it is in charge of representing the three structures, the diversity based on MACCS key is strictly limited. This is a limitation of the pseudo framework-based generation algorithm.

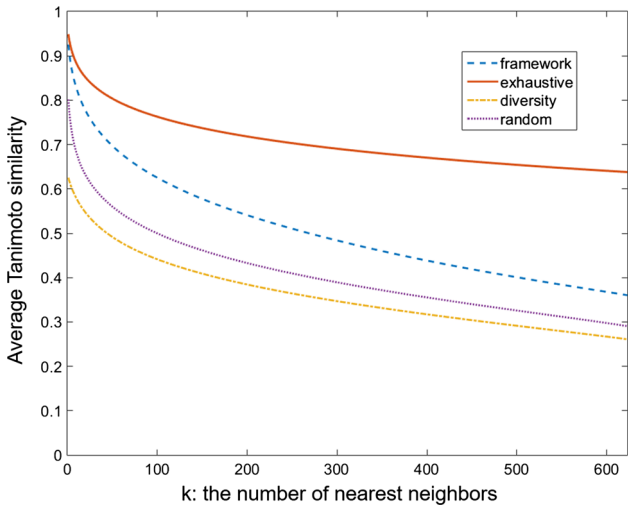

Fig. 16 Average pairwise Tanimoto similarity among *k* nearest neighbors, framework: pseudo-framework-based generation, exhaustive: pool of exhaustive structure generation, diversity: MaxMin sampling from the exhaustive structure pool, random: randomly picking from the exhaustive structure pool. 624 molecules were sampled for diversity and random cases

Stochastic generation

We tested estimation ability by stochastic generation, which prunes nodes in a generation tree with certain probabilities. 100 ring systems were selected randomly from 55,620 unique ring systems obtained by decomposing structures in the ChEMBL 20 database. Types of atom fragments were C, N, O, F, S (with 2 valence), Cl, Br, I. Probability of pruning varied based on the depth of generation tree (i.e. the number of fragments combined), such as $p_0 = 0$, $p_1 = 0.75$, $p_2 = 0.91$, $p_k = 0.95$ ($k > 2$). The results are shown in Fig. 18. The mean and the standard deviation for each number of fragments were calculated by sampling 10 times with different initial conditions for random number generation. Red line shows the numbers of structures that were exhaustively generated and, therefore, the correct numbers of structures. As it is apparent from Fig. 18, the sampling method was able to estimate the number of structures to be generated with adequate

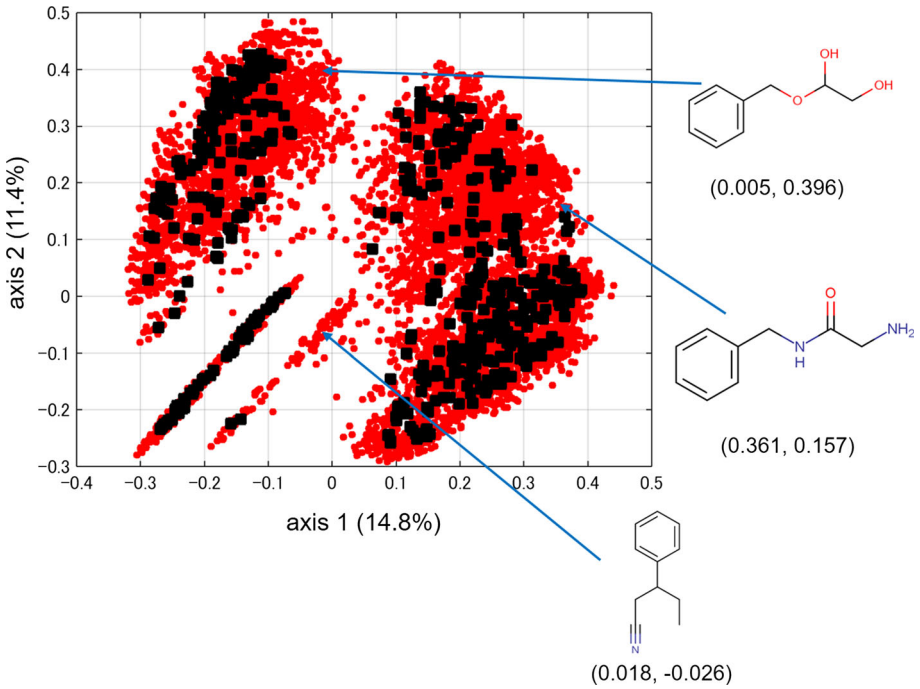

Fig. 17 MDS map with two data sets, *black squares* are structures from pseudo framework-based generation and *red circles* are ones from exhaustive generation. Three structures are extracted from the exhaustive generation pool where no structures from pseudo framework-based generation pool exist

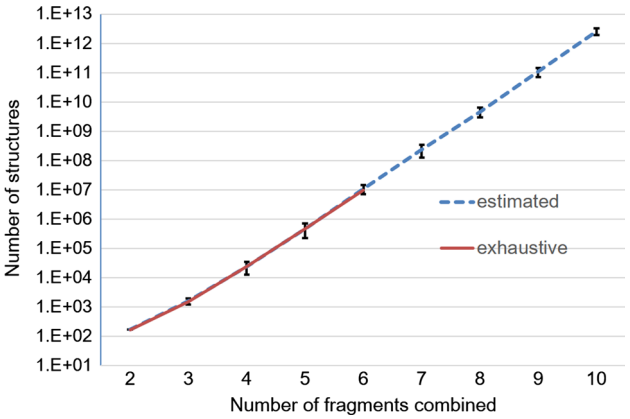

Fig. 18 Number of estimated and generated structures against the number of combined fragments (logarithmic scale). The *red line* is the number of structures that were actually generated without sampling method. *Blue dotted line* is the average estimated number of structures and the range of *error bars* is 2 standard deviations from the average values. 10 trials were conducted for each fragments

precision. We concluded that the stochastic generation was a trustworthy way for estimating the number of structures to be generated exhaustively.

Practical application: structure generation for mimicking rosiglitazone

The goal of this case study is to test Molgilla's generation ability through designing the chemical structures having pharmacophore profile similar to that of rosiglitazone. Rosiglitazone is a selective ligand of peroxisome proliferator-activated receptor (PPaR) gamma, and it is also a full agonist of PPaR gamma. Regardless of its potency as a full agonist, registration of rosiglitazone was withdrawn in Europe and in several other countries, because several adverse effects caused by the thiazolidinedione group in that drug have been reported [50].

One of the advantage of the proposed algorithm is that we can take several constraints into account during generation, not filtering out undesired structures after structure generation completed. The descriptors that can be employed as constraints are called monotonous changing descriptors (MCDs) [51, 52], such as molecular weight, Randic connectivity indices [53], and topological-based autocorrelation descriptors.

We used the summation of topological distance-based descriptors between PPPs (STDPs) for determining the similarity of structures. The definition of STDPs is the summation of all the topological distances between a pair of certain types of PPPs, which is inspired by the widely used chemically advanced template search (CATS) descriptors [54]. Although CATS consists not of the summation of all topological distances, but of the occurrences with a specific pre-determined distance between PPPs, we intentionally chose STDPs instead of CATS descriptors for keeping the calculation speed up. The STDPs of rosiglitazone and its structure are on Table 4 and Fig. 19, respectively. Table 5 shows the constraints for structure generation based on Table 4. In addition to STDPs, numbers of PPPs were also used as constraints. The number of

Table 4 STDPs and PPPs profile of rosiglitazone

LL	LA	LD	LP	LN	AA	AD	AP	AN	DD	DP	DN	PP	PN	NN	RL	RA	RD	RP	RN	RR
(a)																				
2	64	5	0	0	81	36	0	0	0	0	0	0	0	0	24	50	16	0	0	5
L				A				D				P				N				R
(b)																				
2				5				1				0				0				2

(a) STDPs and (b) the number of PPPs

L lipophilic point, *A* hydrogen bond acceptor, *D* hydrogen bond donor, *P* positively charged point, *N* negatively charged point, *R* aromatic ring

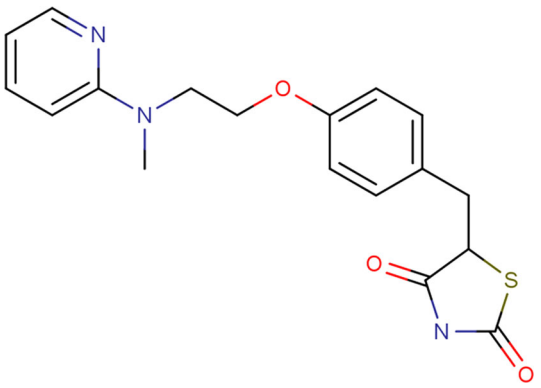

Fig. 19 Two dimensional structure of rosiglitazone. It is noted that the neutral form of rosiglitazone was used for structure generation

ring systems used for generation were 47 (on SI3.sdf as supplementary material). Atom fragments were, C, N, O, F, Cl, Br, and I. Sulfur atoms were only accepted when they are inside a ring system with the valence of two. The number of fragments to be combined was set 11. The prohibition rules that prohibit generating unstable and reactive structure, are as follows in Molgilla settings: nonC–nonC, =C=, C=N, C(=O)X, CH_2X, CH(=O), aliphatic-C(=O)–aliphatic, where nonC is an atom without carbon, X is a halogen, aliphatic is a non-aromatic atom, – is a single and = is a double bond. Furthermore, the generator was not allowed to combine ring systems directly. Also, it did not use = CH_2 for ring systems having access points with degree two, atom fragment C with four hands, atom fragment CH connected to 3 rings and atom fragment N connected to 3 rings.

Molgilla successfully generated 6790 structures satisfying the constraints during its investigation of 11,928,258,809 structures (the number of nodes in the generation tree). The generated structures, which are annotated with the summation of absolute distance errors to rosiglitazone, are on SI4.sdf as supplementary material. The STDPs profiles of generated structures as well as that of rosiglitazone are shown on Fig. 20. There are 12 structures identical to rosiglitazone in terms of STDPs. Among these structures, rosiglitazone was successfully retrieved by Molgilla. Among the top 100 structures close to rosiglitazone, only two structures were found to be actual compounds through SciFinder search (CAS Registry Number 122320-73-4 and 1190268-36-0) including rosiglitazone. Some of the structures contain enol-form as substructures, meaning these structures are unstable. With the current prohibition rules of Molgilla, this unstable form in tautomerism can also be generated. One simple way to overcome this limitation is to introduce the additional rule of not generating enol tautomers, such as by introducing (C=C(OH)) on the taboo list. It is also possible to avoid proposing these structures by applying additional filters to the generated structures. Furthermore, in order to evaluate

Table 5 Constraints for structure generation

LL	LA	LD	LP	LN	AA	AD	AP	AN	DD	DP	DN	PP	PN	NN	RL	RA	RD	RP	RN	RR
(a)																				
0–5	61–67	2–8	0	0	78–84	33–39	0	0	0	0	0	0	0	0	21–27	47–53	13–19	0	0	2–8
L				A				D				P				N				R
(b)																				
1–3				4–6				0–2				0–0				0–0				1–3

Lower and upper bounds of STDPs (a) and PPPs (b) were based on the values on Table 4

L lipophilic point, *A* hydrogen bond acceptor, *D* hydrogen bond donor, *P* positively charged point, *N* negatively charged point, *R* aromatic ring

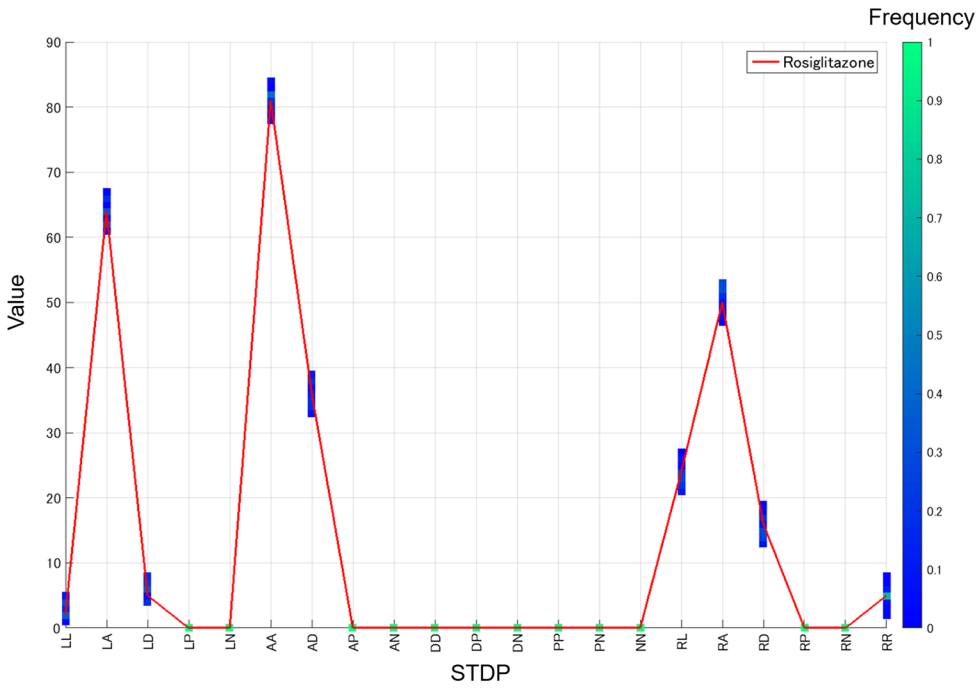

Fig. 20 STDPs profile of the generated 6790 structures. The *color scale* represents the frequency of occurrence of the number of structures. *Red line* shows the profile of rosiglitazone

generated structures from various aspects, additional filters should be applied to them, such as PAINS [55] for ligand design, and the criterion developed by Allu and Oprea for evaluating complexity and synthesizability of chemical structures [56]. Although there need many improvements for Molgilla to be used for practical applications, in particular descriptors, Molgilla may be useful for ligand-based molecular design.

Conclusion

We have described structure generation algorithms by combing ring systems and atom fragments. Efficient generation is expected by using the reduced graph corresponding to a ring system. Furthermore, an algorithm for diversity-oriented generation has been introduced. For testing the efficacy of the proposed algorithms, we implemented structure generator Molgilla, and tested three case studies. First, we showed how to construct isomorphic reduced graphs corresponding to ring systems from the ChEMBL database. From 55,654 structures, 34 ring systems could not find the proper reduced graph by simple template matching rules. We have also shown that the generation speed of Molgilla surpassed that of a simple fragment-combined-based generator in Chemish in a simple case study. This is an evidence that treating ring systems as reduced graphs and using the canonical construction path method can contribute to the reduction of generation time. Second, we compared the two sets of generated structures, one by pseudo framework-based generation and the other by exhaustive generation. We have confirmed that pseudo framework-based generation generated structures represent the exhaustive structures. It is noted that diversity is took part in during structure generation instead of sampling structures after generation. We also visualized those structures on the two dimensional map constructed by MDS and confirmed the method's limitation, where side chain diversities cannot be considered. Finally, we demonstrated the estimation ability of the number of structures that are to be generated by the stochastic generation method. Comparing estimation numbers with ones from actual numbers, it is fair to say that the method may be useful for the estimation. We showed a simple structure generation case study aiming at generating structures for rosiglitazone mimetic. In the generated 6790 structures, rosiglitazone was successfully retrieved along with novel structures.

One of the biggest problems in this type of structure generator is the lacking of synthesizability of proposed structures as well as proposing unstable or reactive structures. Considering synthesis paths during generation phase might not be feasible, because it reduces the calculation speed drastically. Finding retrosynthesis paths for proposed de novo structures is a feasible way to solve it. Although we have implemented several rules as a taboo list during structure generation, further improvement of the list is necessary for making the generator reliable.

Structure generator Molgilla has been developed and improved, where more descriptors were implemented

without reducing calculation speed. In the future, integrating retrosynthesis modules, such as AIPHOS [57], into the generator is a promising way to make Molgilla even more useful. Our immediate next step is to apply the generator to actual applications for de novo drug design.

Acknowledgments The authors are grateful to G. Schneider and D. Reker at the Department of Chemistry and Applied Biosciences, Institute of Pharmaceutical Sciences, ETH Zurich. G. Schneider supported the authors by giving valuable advice for the improvement of our structure generation algorithms, particularly the descriptor calculation and how to generate feasible structures in a chemistry point of view. D. Reker and the authors have discussed how to develop diversity-oriented generation algorithms. The authors also acknowledge the support of the Core Research for Evolutionary Science and Technology (CREST) Project 'Development of a knowledge-generating platform driven by big data in drug discovery through production processes' of the Japan Science and Technology Agency (JST). T.M. is a JSPS Research Fellow.

References

1. Faulon J-L, Bender A (2010) Handbook of chemoinformatics algorithms. CRC Press, Boca Raton
2. Pólya G, Read RC (1987) Combinatorial enumeration of groups, graphs, and chemical compounds. Springer, New York
3. Balaban AT, Kennedy JW, Quintas L (1988) The number of alkanes having N carbons and a longest chain of length D: an application of a theorem of Polya. J Chem Educ 65:304–313
4. Gugisch R, Kerber A, Laue R, Meringer M, Weidinger J (2000) MOLGEN-COMB, a software package for combinatorial chemistry. MATCH 41:189–203
5. Ruch E, Klein DJ (1983) Double cosets in chemistry and physics. Theor Chim Acta 63:447–472
6. Lindsay RK, Buchanan BG, Feigenbaum EA, Lederberg J (1993) DENDRAL: a case study of the first expert system for scientific hypothesis formation. Artif Intell 61:209–261
7. Sasaki S, Kudo Y (1985) Structure elucidation system using structural information from multisources: CHEMICS. J Chem Inf Comput Sci 25:252–257
8. Funatsu K, Miyabayashi N, Sasaki S (1988) Further development of structure generation in the automated structure elucidation system CHEMICS. J Chem Inf Comput Sci 28:18–28
9. Benecke C, Grüner T, Kerber A, Laue R, Wieland T (1997) MOLecular structure GENeration with MOLGEN, new features and future developments. Fresen J Anal Chem 359:23–32
10. Benecke C, Grund R, Hohberger R, Kerber A, Laue R, Wieland T (1995) MOLGEN+, a generator of connectivity isomers and stereoisomers for molecular structure elucidation. Anal Chim Acta 314:141–147
11. Grüner T, Laue R, Meringer M (1997) Algorithms for group actions: homomorphism principle and orderly generation applied to graphs. In: DIMACS Series in Discrete Mathematics and Theoretical Computer Science; American Mathematical Society, vol 28, pp 113–122
12. Faulon JL (1992) On using graph-equivalent classes for the structure elucidation of large molecules. J Chem Inf Comput Sci 32:338–348
13. Kawashita N, Yamasaki H, Miyao T, Kawai K, Sakae Y, Ishikawa T, Mori K, Nakamura S, Kaneko H (2015) <Review> A mini-review on chemoinformatics approaches for drug discovery. J Comput Aided Chem 16:15–29
14. Schneider G, Fechner U (2005) Computer-based de novo design of drug-like molecules. Nat Rev Drug Discov 4:649–663
15. Schneider G, Neidhart W, Giller T, Schmid G (1999) "Scaffold-Hopping" by topological pharmacophore search: a contribution to virtual screening. Angew Chem Int Ed 38:2894–2896
16. Lewell XQ, Judd DB, Watson SP, Hann MM (1998) RECAP-retrosynthetic combinatorial analysis procedure: a powerful new technique for identifying privileged molecular fragments with useful applications in combinatorial chemistry. J Chem Inf Comput Sci 38:511–522
17. Hartenfeller M, Zettl H, Walter M, Rupp M, Reisen F, Proschak E, Weggen S, Stark H, Schneider G (2012) DOGS: reaction-driven de novo design of bioactive compounds. PLoS Comput Biol 8:e1002380
18. Lessel U, Wellenzohn B, Lilienthal M, Claussen H (2009) Searching fragment spaces with feature trees. J Chem Inf Model 49:270–279
19. Rella M (2011) Software review of FTrees and FTrees-FS in pipeline pilot FTrees and FTrees-FS in pipeline pilot. BioSolveIT GmbH. An Der Zieglei 79, 53757 Sankt Augustin, Germany. http://www.biosolveit.de/FTrees. See Web Site for Pricing Information. J Am Chem Soc, vol 133, pp 17101–17102
20. Shimizu M, Nagamochi H, Akutsu T (2011) Enumerating tree-like chemical graphs with given upper and lower bounds on path frequencies. BMC Bioinform 12:1–9
21. Zhao Y, Hayashida M, Jindalertudomdee J, Nagamochi H, Akutsu T (2013) Breadth-first search approach to enumeration of tree-like chemical compounds. J Bioinform Comput Biol 11:1343007
22. Nakano S, Uno T (2005) Generating colored trees. In: Kratsch D (ed) Graph-theoretic concepts in computer science Lecture notes in computer science, vol 3787. Springer, Berlin, pp 249–260
23. Suzuki M, Nagamochi H, Akutsu T (2014) Efficient enumeration of monocyclic chemical graphs with given path frequencies. J Cheminform 6:31
24. Akutsu T, Fukagawa D, Jansson J, Sadakane K (2012) Inferring a graph from path frequency. Discrete Appl Math 160:1416–1428
25. McKay BD (1998) Isomorph-free exhaustive generation. J Algorithms 26:306–324
26. Jaworska J, Nikolova-Jeliazkova N, Aldenberg T (2005) QSAR applicability domain estimation by projection of the training set descriptor space: a review. ATLA 33:445–459
27. Miyao T, Kaneko H, Funatsu K (2014) Ring-system-based exhaustive structure generation for inverse-QSPR/QSAR. Mol Inform 33:764–778
28. Bemis GW, Murcko MA (1996) The properties of known drugs. 1. Molecular frameworks. J Med Chem 39:2887–2893
29. Wester MJ, Pollock SN, Coutsias EA, Allu TK, Muresan S, Oprea TI (2008) Scaffold topologies. 2. Analysis of chemical databases. J Chem Inf Model 48:1311–1324
30. Fisanick W, Lipkus AH, Rusinko A (1994) Similarity searching on CAS registry substances. 2. 2D structural similarity. J Chem Inf Comput Sci 34:130–140
31. Rarey M, Stahl M (2001) Similarity searching in large combinatorial chemistry spaces. J Comput Aided Mol Des 15:497–520
32. McKay BD, Royle G F (1985) Constructing the cubic graphs on up to 20 vertices. Department of Mathematics, University of Western Australia
33. Fink T, Reymond JL (2007) Virtual exploration of the chemical universe up to 11 atoms of C, N, O, F: assembly of 26.4 million structures (110.9 million stereoisomers) and analysis for new ring systems, stereochemistry, physicochemical properties, compound classes, and drug discove. J Chem Inf Model 47:342–353
34. Blum LC, Reymond J-L (2009) 970 million druglike small molecules for virtual screening in the chemical universe database GDB-13. J Am Chem Soc 131:8732–8733

35. Ruddigkeit L, van Deursen R, Blum LC, Reymond J-L (2012) Enumeration of 166 billion organic small molecules in the chemical universe database GDB-17. J Chem Inf Model 52:2864–2875
36. Miyao T, Arakawa M, Funatsu K (2010) Exhaustive structure generation for inverse-QSPR/QSAR. Mol Inform 29:111–125
37. Faulon JL (1996) Stochastic generator of chemical structure. 2. Using simulated annealing to search the space of constitutional isomers. J Chem Inf Comput Sci 36:731–740
38. Virshup AM, Contreras-García J, Wipf P, Yang W, Beratan DN (2013) Stochastic voyages into uncharted chemical space produce a representative library of all possible drug-like compounds. J Am Chem Soc 135:7296–7303
39. Bento AP, Gaulton A, Hersey A, Bellis LJ, Chambers J, Davies M, Krüger FA, Light Y, Mak L, McGlinchey S, Nowotka M, Papadatos G, Santos R, Overington JP (2014) The ChEMBL bioactivity database: an update. Nucleic Acids Res 42:1083–1090
40. Landrum G RDKit (2016) Open-source cheminformatics http://www.rdkit.org. Accessed 12 Mar 2016
41. Berthold MR, Cebron N, Dill F, Gabriel TR, Koetter T, Meinl T, Ohl P, Sieb C, Thiel K, Wiswedel B (2008) KNIME: the Konstanz information miner. In: Preisach C, Burkhardt H, Schmidt-Thieme L, Decker R (eds) Data analysis, machine learning and applications. Springer, Berlin, pp 319–326
42. Taylor RD, MacCoss M, Lawson ADG (2014) Rings in drugs. J Med Chem 57:5845–5859
43. Arakawa M, Yamada Y, Funatsu K (2005) Development of the computer software. J Comput Aided Chem 6:90–96
44. Chemish: Chemometorics Software (2016) http://www.cheminfonavi.co.jp/chemish. Accessed 12 Mar 2016
45. Rishton GM (1997) Reactive compounds and in vitro false positives in HTS. Drug Discov Today 2:382–384
46. Rishton GM (2003) Nonleadlikeness and leadlikeness in biochemical screening. Drug Discov Today 8:86–96
47. Pavlov D, Rybalkin M, Karulin B, Kozhevnikov M, Savelyev A, Churinov A (2011) Indigo: universal cheminformatics API. J Cheminform 3:4
48. Durant JL, Leland BA, Henry DR, Nourse JG (2002) Reoptimization of MDL keys for use in drug discovery. J Chem Inf Comput Sci 42:1273–1280
49. Ashton M, Barnard J, Casset F, Charlton M, Downs G, Gorse D, Holliday J, Lahana R, Willett P (2002) Identification of diverse database subsets using property-based and fragment-based molecular descriptions. Quant Struct Act Rel 21:598–604
50. Rizos CV, Elisaf MS, Mikhailidis DP, Liberopoulos EN (2009) How safe is the use of thiazolidinediones in clinical practice? Expert Opin Drug Saf 8:15–32
51. Miyao T, Kaneko H, Funatsu K (2016) Ring-system-based chemical structure enumeration for de novo design. Yakugaku Zasshi 136:101–106
52. Miyao T, Kaneko H, Funatsu K (2016) Inverse QSPR/QSAR analysis for chemical structure generation (from Y to X). J Chem Inf Model 56:286–299
53. Randic M (1975) Characterization of molecular branching. J Am Chem Soc 97:6609–6615
54. Reutlinger M, Koch CP, Reker D, Todoroff N, Schneider P, Rodrigues T, Schneider G (2013) Chemically advanced template search (CATS) for scaffold-hopping and prospective target prediction for "Orphan" molecules. Mol Inform 32:133–138
55. Baell JB, Holloway GA (2010) New substructure filters for removal of pan assay interference compounds (PAINS) from screening libraries and for their exclusion in bioassays. J Med Chem 53:2719–2740
56. Allu TK, Oprea TI (2005) Rapid evaluation of synthetic and molecular complexity for in silico chemistry. J Chem Inf Model 45:1237–1243
57. Funatsu K, Sasaki S (1988) Computer-assisted organic synthesis design and reaction prediction system, "AIPHOS". Tetrahedron Comput Methodol 1:27–37