Efficient Algorithms To Enumerate Isomers and Diamutamers with More Than One Type of Substituent

Markus van Almsick,† Hans Dolhaine,‡ and Helmut Hönig*,§

Am Markuskreuz 6, 45133 Essen, Germany, Henkel KGaA, Henkelstrasse 67, 40191 Düsseldorf, Germany, and Institute for Organic Chemistry, Technical University Graz, Stremayrgasse 16, 8010 Graz, Austria

Received September 16, 1999

In this paper we describe numeric as well as symbolic algorithms for the enumeration of substitutional isomers with an unlimited number of different achiral substituents. We consider three different scenarios: first, the enumeration of diamutamers with a given set of ligand types and ligand multiplicity, second, the enumeration of diamutamer libraries with a given ligand assortment pattern, and, third, the enumerations of libraries with diamutamers exhibiting a limited number of ligands.

1. INTRODUCTION

Chemical compounds that exhibit identical molecular formulas but differ in constitution are called *isomers*. Isomers with constitutions that exhibit the same central skeleton but differ in the arrangement of ligands are called *diamutamers*.[1] A set of diamutamers that one assembles given a central skeleton and an unlimited supply of *n* ligand types is called a *library*. The algorithms introduced below efficiently calculate the number and library size of diamutamers with one or more types of achiral ligands of specified, unspecified, and partially specified multiplicity.

Prior to any isomer, diamutamer, or library enumeration, one has to determine two criteria. First, one has to define the ensemble of chemical compounds to be considered in the enumeration. This is usually done by listing the common properties, such as the number and types of atoms or subgroups, constitutions, and configurations. Second, one has to specify the conditions under which two compounds are to be regarded equivalent. This is done with the help of symmetry transformations that define an equivalence relation between identical isomers. Obviously, many different kinds of isomer enumerations are possible. Here, we limit our considerations to configurational diamutamers with achiral substituents.

The enumeration algorithms described in this paper are implemented in a program called ISOMERS.[2] We assume that the reader is a potential user of this program and thus proceed as follows. We begin with the input data. As mentioned above, the input data consist of diamutamer ensemble specifications and of applicable spatial symmetry transformations determining the isomer equivalence. The ensembles are defined through a diamutamer skeleton, the binding sites, and the types, number, and assortment of achiral ligands. The symmetry transformations are provided as binding site permutations. The construction and classification of the binding site permutations, the derivation of Pòlya cycle indices, and a point group identification scheme for input verification are the subject of section 2. We conclude section 2 with the Cauchy−Frobenius lemma, a prerequisite for all the enumeration algorithms that follow. Section 3 introduces an algorithm that enumerates diamutamers with given ligand types and multiplicities. This algorithm utilizes the Pòlya theorem[3] by efficiently calculating the coefficients of expanded Pòlya polynomials. In section 4 we generalize the previous algorithm to enumerate diamutamer libraries with specified ligand assortment patterns. This method can also be used to enumerate libraries of diamutamers with a limited number of attached ligands. However, another procedure introduced in section 5 is far superior. It is especially tailored for counting diamutamers with ligands of arbitrary composition but fixed number. Section 6 introduces the program ISOMERS, written in the computer algebra language MATHEMATICA.[4] An example calculation of cyclohexane demonstrates the handling of the program. Other more elaborate diamutamer enumerations illustrate the scope of the software.

2. PÒLYA CYCLE INDICES

The main ingredient for all our enumeration algorithms are Pòlya cycle indices. These cycle indices encode the relevant information about a diamutamer and its symmetry and number of binding sites. In this section we discuss the generation of Pòlya cycle indices. We start with a parent compound and a set of substituent types. The parent compound, also referred to as the skeleton, should exhibit a limited number of binding sites, to each of which at most one substituent may be attached in just one way. The supply of an arbitrary number of achiral substituents may consist of a given number of different substituent types.

2.1. Symmetry Group. We want to count all possible ways to arrange and attach substituents to a parent compound. This task is not trivial, because spatial symmetries of the parent compound render some substituent patterns equivalent. Two or more substituent patterns are indistinguishable, if there exists a symmetry transformation of the parent com-

† Am Markuskreuz 6. E-mail: m.van.almsick@cityweb.de.
‡ Henkel KGaA. E-mail: hans.dolhaine@denotes.henkel.de.
§ Technical University Graz. E-mail: hoenig@orgc.tu-graz.ac.at.

pound, typically a rotation, that maps the two or more substituent patterns into one another (e.g., 1,3-dibromobenzol and 3,5-dibromobenzol, where the latter does not obey the rules of nomenclature). These symmetry transformations partition the set of all possible substituent patterns into distinct equivalence classes, each encompassing all those compounds that can be transformed into each other. The task is to enumerate these isomer equivalence classes. They represent the distinguishable isomers.

To count the isomer equivalence classes, we have to identify the symmetry transformations that define them. Symmetry transformations are maps that take each atom and each bond of a compound into an image that is congruent with the original compound. Hence, each atom maps into an atom of the same type, and each bond maps into a bond of the same kind.

We distinguish three kinds of symmetry transformations. (1) Rotations: rotations are symmetry transformations that rotate a compound as a whole in three-dimensional Euclidean space around one or more axes. (2) Reflections/inversions: reflections map each part of a parent compound across a plane into a mirror image. Inversions flip every (atom) position vector across an origin into the opposite direction. (3) Conformational transformations: conformational transformations are symmetry transformations that affect the chemical bonds of a parent compound by twisting, or rotating, them. Rotations and reflections as well as inversions may be part of a conformational transformation.

Rotations do not alter chemical compounds in any way. One can always perform a rotation on an actual chemical compound. It may, however, not be possible to obtain a reflected or inverted diamutamer without altering the structure of the parent compound. Any molecule that cannot be rotated into its mirror image is a chiral molecule, hence the distinction between rotations and reflections/inversions. Nevertheless, rotations and reflections/inversions have one property in common. They are spatial symmetry transformations that leave one fix point invariant, thus the name *point group transformation*.

The symmetries of rigid structures are limited to point group symmetries. Chemical bonds, however, may not always be viewed as rigid links. Some bonds can be twisted; some dihedral angles along σ-bonds can be rotated. This allows some molecules to change their conformation. In combination with rotations and reflections, we thereby obtain symmetry transformations that extend the set of point group transformations. Since a molecule may have to overcome an energy barrier to change its conformation, we treat conformational transformations separately, so that one can include or exclude them in an isomer enumeration.

We apply the above symmetry considerations to a diamutamer skeleton and use methane CH_4 as an instructive example. The central carbon atom of CH_4 forms the skeleton. Each of the four hydrogen ligands H may be substituted and represents a substituent site. We label these substituent sites with labels 1, 2, 3, and 4. The central C atom of methane and its sp^3 hybrid orbitals exhibit a simple tetrahedral symmetry as depicted in Figure 1. For larger skeletons the symmetry identification may be more difficult and may require the help of a computer program such as SYMMOL.[5]

Mathematically, symmetries are encoded as invariances under symmetry transformations g. These symmetry transformations can be classified according to the three categories above: rotations, reflections/inversions, and conformational transformations.

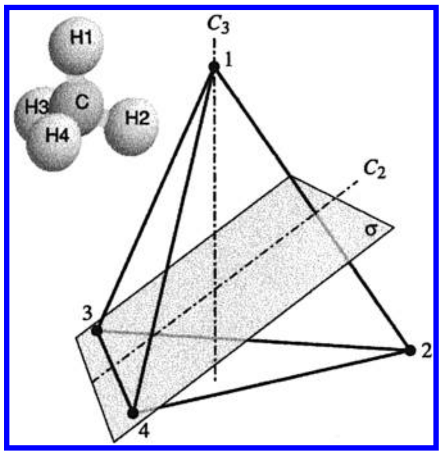


Figure 1. Methane CH_4 and its symmetries depicted by a tetrahedron with three of its point group symmetry transformations, C_2, C_3, and σ.

Each symmetry transformation g induces a permutation p_g of substituent sites. All these site permutations have to be taken into account. Fortunately, it is sufficient to identify the permutations of only those symmetry transformations with which all others can be performed via concatenation. A set of such symmetry transformations is called a generator set. In the case of methane three generators are needed. Two generators are rotations, one 3-fold and the other 2-fold. Their axes,[6] C_3 and C_2, are shown in Figure 1. The third generator, a reflection, is depicted by a reflection plane,[7] σ. No conformational transformations are present in this example. The resulting permutations of methane's four substituent sites are

$$p_{C_3} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 3 & 4 & 2 \end{pmatrix}$$

$$p_{C_2} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 3 \end{pmatrix}$$

$$p_{\sigma} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 3 & 4 \end{pmatrix}$$

These generators form permutation group P_G with 24 elements p_g representing the 24 transformations g of methane's symmetry group G.

2.2. Point Group Identification. The program ISOMERS automatically generates the permutation group P_G. With P_G at hand, it is possible to derive the necessary group properties to obtain the Pòlya cycle index and, equally important, to identify the symmetry group G in retrospect, thereby verifying the permutation group generators. To achieve this, we determine the conjugate equivalence classes and cycle structures of permutation group P_G, as presented for methane in Table 1.

The conjugate equivalence relation

$$p_a \equiv p_b \Leftrightarrow \exists\, p_c \in P_G, \quad \text{with} \quad p_c \cdot p_a \cdot {p_c}^{-1} = p_b$$

partitions the set of group elements into conjugate equiva-

Table 1. Conjugate Equivalence Classes and Point Group Cycle Structures of the Tetrahedral Point Group T_d[a]

conjugate class size	point group cycle structure	permutation sample in cyclic notation
1	E	(1) (2) (3) (4)
8	C_3 (4), C_3^2 (4)	(1) (342)
3	C_2 (3)	(21) (43)
6	σC_4 (3), σC_4^3 (3)	(2341)
6	σ (6)	(21) (3) (4)

[a] The first column lists the number of permutations in each conjugate class. The second column displays the point group cycle structures. The notation is explained in the text. The third column lists permutation examples whose partition into permutation cycles is a property of their conjugate class.

lence classes. The number of group elements in each of these classes (for methane, see the first column of Table 1) is the first criterion to identify the point group.

Furthermore, rotations around an n-fold axis give rise to a group cycle $\{C_n, C_n^2, ..., E = C_n^n\}$, where C_n denotes a rotation of $(1/n)360°$, $C_n^2 = C_n{\cdot}C_n$ represents a rotation of $(2/n)360°$, and C_n^n stands for a 360° rotation, which is equivalent to the identity transformation E. We determine the group cycles in the permutation subgroup representing pure rotations. These group cycles allow us to identify the rotations and to label the subgroup elements according to C_n^p (for variations in notation found in the literature see ref 8). For point group elements containing a reflection or inversion, we proceed accordingly and place a σ in front of C_n^p to indicate a pseudorotation. We count the number of group elements with a given label in each equivalence class and obtain a point group cycle structure like the one in the second column of Table 1. For each label the number of corresponding group elements is appended in parentheses. The point group cycle structure is the second criterion to identify the point group.

The number and size of equivalence classes and the cycle structure of P_G are almost sufficient to uniquely identify the corresponding point group G. However, there are point groups G that have isomorphic permutation groups P_G exhibiting the same equivalence classes as well as cycle structures. These point groups are in Schoenflies notation:[9]

$$\begin{gathered} S_2 \equiv C_{1h} \equiv C_1 \equiv C_s \\ C_{2h} \equiv C_{2v} \\ \left.\begin{aligned} S_{4n+2} &\equiv C_{(2n+1)h} \\ D_{(2n+1)d} &\equiv D_{(2n+1)h} \end{aligned}\right\} n \in \mathrm{IN} \end{gathered} \tag{1}$$

It is impossible to discriminate the point groups in (1) given the site permutations representing rotations and reflections/inversions.

The program ISOMERS renders the identified point groups G in Schoenflies notation, which, in the case of methane, is T_d. This extra feature of ISOMERS is not needed for a diamutamer enumeration, but greatly helps to eliminate errors in the input data.

2.3. Pòlya Cycle Index Construction. The most compact mathematical encoding of the skeleton symmetry data is a Pòlya cycle index. Here we explain how to construct such an index. The motivation and theoretical background will be given in the next section.

To derive a Pòlya cycle index $Z(P_G)$, we take the substituent site permutations p_g and determine the length of each permutation cycle. This only needs to be done for one permutation p_g per equivalence class, since the length of the permutation cycles does not vary within a conjugate equivalence class. The results for methane are displayed in the third column of Table 1. To construct a Pòlya cycle index $Z(P_G)$, one replaces each permutation cycle $c_i(p_g)$ of length $|c_i(p_g)|$ with an index variable $f_{|c_i(p_g)|}$; in this fashion one converts each substituent permutation p_g into a product of index variables, one sums over all permutations p_g, and finally divides the sum by the number of permutation group elements $|P_G|$. The result is a Pòlya cycle index

$$Z(P_G) = \frac{1}{|P_G|}\sum_{p_g \in P_G}\left(\prod_i f_{|c_i(p_g)|}\right) \tag{2}$$

For methane we can determine two Pòlya cycle indices, one excluding the reflection symmetry σ, and one including it. The Pòlya cycle index for methane excluding reflections is

$$Z(\text{methane}) = (1/12)(f_1^4 + 8f_1f_3 + 3f_2^2)$$

The Pòlya cycle index for methane including reflections is

$$Z_\sigma(\text{methane}) = (1/24)(f_1^4 + 8f_1f_3 + 3f_2^2 + 6f_4 + 6f_1^2f_2)$$

This concludes the survey of input data. Most of the steps above are automated in the ISOMERS program. The Pòlya cycle index is the starting point for all enumeration schemes that follow.

2.4. Cauchy−Frobenius Lemma. The Cauchy−Frobenius lemma is the cornerstone of many enumeration schemes, including methods utilizing Polya's polynomial. To state the lemma of Cauchy and Frobenius, we introduce a few group theoretical terms.

(1) P_G denotes a finite group consisting of permutations p_g, which permute the elements x of ordered set X. The permutations p_g are bijective maps p_g:$X \rightarrow X$ representing the symmetry transformations g of point group G with p_g: $x \rightarrow p_g(x)$, $(p_g{\cdot}p_{g'})(x) = p_{g'}(p_g(x))$, $p_{g^{-1}}(x) = p_g^{-1}(x)$, and $p_\epsilon(x) = x$. X will denote the set of distinguishable as well as indistinguishable diamutamers.

(2) $\omega_G(x)$ represents the set $\{p_g(x)|p_g \in P_G\}$ for a given $x \in X$. The $\omega_G(x)$ with $x \in X$ form equivalence classes with respect to the equivalence relation $x \equiv x' \Leftrightarrow \exists\, p_g \in P_G$, with $p_g(x) = x'$. The $\omega_G(x)$ will represent the isomer equivalence classes representing the indistinguishable isomers.

(3) $\Omega_G(X)$ is the set of all classes $\omega_G(x)$ with $x \in X$ generated by P_G. Thus, $\Omega_G(X)$ is the set of disjoint equivalence classes in X. $\Omega_G(X)$ will denote the set of all distinguishable isomers.

(4) X_{p_g} is the set of fix points $\{x \in X|p_g(x) = x\}$ under a single permutation p_g.

(5) $(P_G)_x$ is the stabilizer $\{p_g \in P_G|p_g(x) = x\}$ of $x \in X$. $(P_G)_x$ is the largest subgroup of P_G that leaves x invariant.

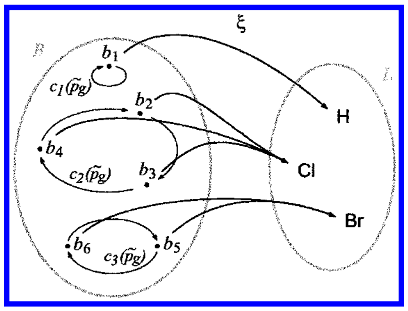


Figure 2. A diamutamer is represented by a map ξ matching the susbtituent site in B with the ligand types in L. The circles in B represent the cycles $c_i(\tilde{p}_g)$ generated by a site permutation $\tilde{p}_g$. The mapping ξ shown is invariant under permutation $\tilde{p}_g$ and thus a fix point.

In an isomer enumeration we determine the cardinality $|\Omega_G(X)|$, which is the number of elements of the set $\Omega_G(X)$ of all isomers equivalence classes $\omega_G(X)$. Instead of counting all $\omega_G(x) \in \Omega_G(X)$ the Cauchy-Frobenius lemma (3) allows us to enumerate the fix points X_{p_g}.

$$|\Omega_G(X)| = \frac{1}{|P_G|}\sum_{p_g \in P_G} |X_{p_g}| \tag{3}$$

A proof can be found in ref 10.

We apply the Cauchy–Frobenius lemma (3) to a diamutamer enumeration as follows. Initially, we are given a diamutamer skeleton with s substituent sites and a set L containing n ligand types. A diamutamer is represented by a map ξ: $B \rightarrow L$, associating every substituent site in B with a ligand type in L (see Figure 2). We assume the set L^B of all possible maps ξ to be the set X.

Point group transformations g rotate and transform a diamutamer skeleton and thereby permute the substituent sites in B according to a site permutation $\tilde{p}_g(B)$. Each site permutation $\tilde{p}_g$ induces a permutation of maps $\xi \in X$.

$$p_g(\xi)(B) = \xi(\tilde{p}_g(B))$$

We thereby have obtained all the necessary ingredients to apply the Cauchy–Frobenius lemma (3): an ordered set X and a permutation group P_G operating on X. The sets $\omega_G(\xi)$ are the isomer equivalence classes. To enumerate the diamutamers, we have to determine the number of classes $|\Omega_G(X)|$.

According to the Cauchy–Frobenius lemma (3), we only have to count the number of fix points $|X_{p_g}|$ for all $p_g \in P_G$. This turns out to be a simple task, since fix point maps $\xi \in X_{p_g}$ are easily constructed. To obtain an invariant map ξ for a given site permutation $\tilde{p}_g$, one has to ensure that all the substituent sites within a permutation cycle $c_i(\tilde{p}_g)$ are mapped onto the same ligand type as shown in Figure 2. The permutations of substituent sites within a cycle do not affect such a map. Site permutations between cycles do not occur. Hence, we are looking at an invariant map ξ, a fix point of permutation p_g.

The following three sections address the enumeration of fix points $|X_{p_g}|$ under three different enumeration constraints. The section about Pòlya polynomial expansion focuses on the enumeration of invariant maps from B into L with given multiplicity for each ligand type. We ease the constraint of ligand multiplicity in the subsequent section by specifying ligand assortment patterns instead. The third section features an algorithm that enumerates libraries of diamutamers with limited numbers of attached ligands.

3. PÒLYA POLYNOMIAL EXPANSION

Before addressing the problem of Pòlya polynomial expansion, we quickly recapitulate the concept of Pòlya polynomials and the relation to the Cauchy–Frobenius lemma.

3.1. Pòlya Polynomial. To count the number of fix points $|X_{p_g}|$, one can simply state the different ways on how to construct an invariant map ξ. Taking Figure 2 as an example, we could attach one H, one Cl, or one Br to b_1, and connect three H, three Cl, or three Br atoms to b_2, b_3, and b_4, and, finally, assign two H, two Cl, or two Br atoms to b_5 and b_6. Rewriting this in Boolean logic, we obtain

$$\begin{aligned}
&\left(\begin{bmatrix} b_1\\ \downarrow\\ \mathrm{H}\end{bmatrix} \vee \begin{bmatrix} b_1\\ \downarrow\\ \mathrm{Cl}\end{bmatrix} \vee \begin{bmatrix} b_1\\ \downarrow\\ \mathrm{Br}\end{bmatrix}\right)\\
\wedge &\left(\begin{bmatrix} b_2 & & b_3 & & b_4\\ \downarrow & \wedge & \downarrow & \wedge & \downarrow\\ \mathrm{H} & & \mathrm{H} & & \mathrm{H}\end{bmatrix} \vee \begin{bmatrix} b_2 & & b_3 & & b_4\\ \downarrow & \wedge & \downarrow & \wedge & \downarrow\\ \mathrm{Cl} & & \mathrm{Cl} & & \mathrm{Cl}\end{bmatrix}\right.\\
&\qquad\left.\vee \begin{bmatrix} b_2 & & b_3 & & b_4\\ \downarrow & \wedge & \downarrow & \wedge & \downarrow\\ \mathrm{Br} & & \mathrm{Br} & & \mathrm{Br}\end{bmatrix}\right)\\
\wedge &\left(\begin{bmatrix} b_5 & & b_6\\ \downarrow & \wedge & \downarrow\\ \mathrm{H} & & \mathrm{H}\end{bmatrix} \vee \begin{bmatrix} b_5 & & b_6\\ \downarrow & \wedge & \downarrow\\ \mathrm{Cl} & & \mathrm{Cl}\end{bmatrix} \vee \begin{bmatrix} b_5 & & b_6\\ \downarrow & \wedge & \downarrow\\ \mathrm{Br} & & \mathrm{Br}\end{bmatrix}\right)
\end{aligned} \tag{4}$$

According to the distributive law of Boolean logic[11] one can expand (4) to

$$\begin{aligned}
&\left(\begin{bmatrix} b_1\\ \downarrow\\ \mathrm{H}\end{bmatrix} \wedge \begin{bmatrix} b_2 & & b_3 & & b_4\\ \downarrow & \wedge & \downarrow & \wedge & \downarrow\\ \mathrm{H} & & \mathrm{H} & & \mathrm{H}\end{bmatrix} \wedge \begin{bmatrix} b_5 & & b_6\\ \downarrow & \wedge & \downarrow\\ \mathrm{H} & & \mathrm{H}\end{bmatrix}\right)\\
\vee &\left(\begin{bmatrix} b_1\\ \downarrow\\ \mathrm{Cl}\end{bmatrix} \wedge \begin{bmatrix} b_2 & & b_3 & & b_4\\ \downarrow & \wedge & \downarrow & \wedge & \downarrow\\ \mathrm{H} & & \mathrm{H} & & \mathrm{H}\end{bmatrix} \wedge \begin{bmatrix} b_5 & & b_6\\ \downarrow & \wedge & \downarrow\\ \mathrm{H} & & \mathrm{H}\end{bmatrix}\right)\\
\vee &\left(\begin{bmatrix} b_1\\ \downarrow\\ \mathrm{Br}\end{bmatrix} \wedge \begin{bmatrix} b_2 & & b_3 & & b_4\\ \downarrow & \wedge & \downarrow & \wedge & \downarrow\\ \mathrm{H} & & \mathrm{H} & & \mathrm{H}\end{bmatrix} \wedge \begin{bmatrix} b_5 & & b_6\\ \downarrow & \wedge & \downarrow\\ \mathrm{H} & & \mathrm{H}\end{bmatrix}\right)\\
\vee &\left(\begin{bmatrix} b_1\\ \downarrow\\ \mathrm{H}\end{bmatrix} \wedge \begin{bmatrix} b_2 & & b_3 & & b_4\\ \downarrow & \wedge & \downarrow & \wedge & \downarrow\\ \mathrm{Cl} & & \mathrm{Cl} & & \mathrm{Cl}\end{bmatrix} \wedge \begin{bmatrix} b_5 & & b_6\\ \downarrow & \wedge & \downarrow\\ \mathrm{H} & & \mathrm{H}\end{bmatrix}\right) \ldots\\
\vee &\left(\begin{bmatrix} b_1\\ \downarrow\\ \mathrm{Br}\end{bmatrix} \wedge \begin{bmatrix} b_2 & & b_3 & & b_4\\ \downarrow & \wedge & \downarrow & \wedge & \downarrow\\ \mathrm{Br} & & \mathrm{Br} & & \mathrm{Br}\end{bmatrix} \wedge \begin{bmatrix} b_5 & & b_6\\ \downarrow & \wedge & \downarrow\\ \mathrm{Br} & & \mathrm{Br}\end{bmatrix}\right)
\end{aligned} \tag{5}$$

Every line in (5) is an invariant map ξ, and the expansion of (4) delivers all invariant maps under p_g.

The expansion of a Boolean expression corresponds to the expansion of a polynomial if one replaces the $\wedge$'s by multiplication and the $\vee$'s by addition. In this way one can apply polynomial calculations to the enumeration of invariant maps. To do so, we simply replace each map $b_i \rightarrow$ X by the corresponding ligand variable X, all $\wedge$'s by $\times$, and all $\vee$'s by $+$. The boolean expression (4) is represented by

$$(\mathrm{H} + \mathrm{Cl} + \mathrm{Br})(\mathrm{H}^3 + \mathrm{Cl}^3 + \mathrm{Br}^3)(\mathrm{H}^2 + \mathrm{Cl}^2 + \mathrm{Br}^2) \tag{6}$$

and expands to

$$H^6 + H^5Cl + H^5Br + H^4Cl^2 + H^4Br^2 + 2H^3Cl^3 + H^3Cl^2Br + ... + ClBr^5 + Br^6 \quad (7)$$

Every monomial in (7) stands for an invariant map ξ. The variables of the monomials indicate the assortment of ligand types.[12] The monomial H^4Cl^2, for example, represents an invariant map with four hydrogen and two chlorine atoms. The coefficient in front of H^3Cl^3 indicates the existence of 2 invariant maps with three hydrogen and three chlorine atoms. In this fashion, the polynomials can be used as generating functions to count the numbers $|X_{p_g}|$ of invariant maps under permutation p_g. The generating function of $|X_{p_g}|$ for a given assortment of ligand types is

$$\prod_i (\sum_{X \in L} X^{|c_i(p_g)|}) \quad (8)$$

The Cauchy−Frobenius lemma (3) sums over $|X_{p_g}|$ for all $p_g \in P_G$. Thus, we obtain a generating function for $|\Omega_G(X)|$ by substituting (8) for $|X_{p_g}|$ in (3).

$$\frac{1}{|P_G|} \sum_{p_g \in P_G} \prod_i (\sum_{X \in L} X^{|c_i(p_g)|}) \quad (9)$$

(9) is called the *Pòlya polynomial*. Comparing the Pòlya polynomial (9) with the Pòlya cycle index (2), we find the simple connection.

$$f_{|c_i(p_g)|} = \sum_{X \in L} X^{|c_i(p_g)|} \quad (10)$$

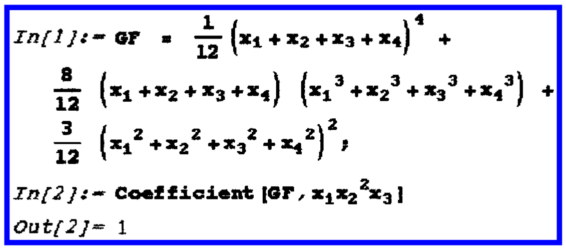
Consequently Pòlya cycle indices can be used to construct the corresponding Pòlya polynomials. Expanding a Pòlya ploynomial, we find as coefficients the number of diamutamers $|\Omega_G(X)|$ for a given assortment of ligand types as indicated by the monomials.

To complete this short introduction, we enumerate the number of diamutamers for methane, substituting three of the four hydrogens H with two ligands A and one ligand B. Disregarding the set of possible ligands $L = \{A, B, H\}$, we first replace the index variable f_k by $(x_1^k + x_2^k + x_3^k + x_4^k)$, because methane has four substituent sites and can accept up to four ligands x_1, x_2, x_3, and x_4. We then take the Pòlyas polynomial resulting from the methane Pòlya cycle index (excluding reflections) and expand it. We extract the coefficient of monomial $x_1x_2^2x_3$,x_1 representing one hydrogen atom H, x_2^2 denoting two ligands A, and x_3 standing for one ligand B. The coefficient we find is 1. Hence, there is only one possible configuration and consequently only one stereoisomer for CHA_2B.

Considering Pòlyas cycle indices and polynomials with or without reflections and with or without conformational transformations, one can enumerate configurational or conformational stereoisomers as well as configurational or conformational diastereoisomers.

3.2. Polynomial Expansion. Given a generating polynomial, we need to find its expansion coefficients. For small polynomials this task can be handled by a built-in MATHEMATICA command called Coefficient.

For the previous methane example we enter



```
In[1]:= GF  =  1/12 (x1 + x2 + x3 + x4)^4 +
     8/12 (x1 + x2 + x3 + x4) (x1^3 + x2^3 + x3^3 + x4^3) +
     3/12 (x1^2 + x2^2 + x3^2 + x4^2)^2;

In[2]:= Coefficient[GF, x1 x2^2 x3]
Out[2]= 1
```

The built-in Coefficient routine can process any kind of polynomial, but may demand large amounts of memory and processor time to do so. We therefore introduce a far more efficient algorithm that determines the coefficients of Pòlya polynomials in a fraction of the time, especially for compounds with many substituent sites. The increase in processing speed for methane with four substituent sites is 40%, cyclohexane with 12 substituents is calculated 4 times faster, and kekulene with 24 sites is processed already 80 times faster.

We were able to achieve this tremendous increase in speed by specializing on Pòlya polynomials. All generating functions encountered are linear combinations of the following polynomial type:

$$S(\vec{\alpha};\vec{x}) := (x_1 + x_2 + ... + x_n)^{\alpha_1} \cdot (x_1^2 + x_2{}^2 + ... + x_n{}^2)^{\alpha_2} ... (x_1{}^m + x_2{}^m + ... + x_n{}^m)^{\alpha_m}$$

$$= \prod_{k=1}^{m} (x_1{}^k + x_2{}^k + ... + x_n{}^k)^{\alpha_k} \quad (11)$$

$$\vec{\alpha} := \{\alpha_1, \alpha_2, ..., \alpha_m\}$$

$$\vec{x} := \{x_1, x_2, ..., x_n\}$$

With respect to (11) we can rewrite the Pòlya polynomial GF for methane in the MATHEMATICA input above as

$$GF(\vec{x}) = \frac{1}{12} S(\{4,0,0,0\};\vec{x}) + \frac{3}{12} S(\{0,2,0,0\};\vec{x}) + \frac{8}{12} S(\{1,0,1,0\};\vec{x})$$

The algorithm, introduced here, determines the expansion coefficients of $S(\vec{\alpha};\vec{x})$. Once the expansion coefficients of $S(\vec{\alpha};\vec{x})$ are calculated, one can assemble the expansion coefficients of the complete generating function GF linearly (see (25)).

We determine the expansion coefficients of $S(\vec{\alpha};\vec{x})$ by rewriting polynomial $S(\vec{\alpha};\vec{x})$ in an expanded form as depicted in (12). We can, thus, extract a formula for the expansion coefficients $\chi_{(\alpha_1,\alpha_2,...,\alpha_m)}^{(\mu_1,\mu_2,...,\mu_n)}$.

$$S(\vec{\alpha};\vec{x}) = \sum_{\mu_1,\mu_2,...,\mu_n} \chi_{(\alpha_1,\alpha_2,...\alpha_n)}^{(\mu_1,\mu_2,...,\mu_n)} \cdot x_1{}^{\mu_1} \cdot x_2{}^{\mu_2} ... x_n{}^{\mu_n} \quad (12)$$

First, we expand the polynomial factors $(x_1{}^k \cdot x_2{}^k ... x_n{}^k)^{\alpha_k}$ in (11) utilizing the formula for multinomials in (13) and (14).

$$(x_1 + x_2 + ... + x_n)^{\alpha} = \sum_{\substack{a_1,a_2,...,a_n \\ \sum_{j=1}^{n} a_j = \alpha}} \frac{\alpha!}{a_1! a_2! ... a_n!} \cdot x_1^{a_1} \cdot x_2^{a_2} ... x_n{}^{a_n} \quad (13)$$

Evidently, the above equation remains valid after substituting x_i^k for x_i.

$$(x_1^k + x_2^k + ... + x_n^k)^\alpha = \sum_{\substack{a_1,a_2,...,a_n \\ \sum_{j=1}^n a_j=\alpha}} \frac{\alpha!}{a_1!a_2!...a_n!}\cdot x_1^{k\cdot a_1}\cdot x_2^{k\cdot a_2}...x_n^{k\cdot a_n} \quad (14)$$

To distinguish the m polynomial factors of $S(\vec{\alpha};\vec{x})$, we attach to exponent α and to all nonnegative integers a_j in (14) an additional index k. Inserting (14) into (11) yields

$$S(\vec{\alpha};\vec{x}) = \prod_{k=1}^m \left(\sum_{\substack{a_{k1},...,a_{kn} \\ \sum_{j=1}^n a_{kj}=\alpha_k}} \frac{\alpha_k!}{a_{k1}!...a_{kn}!}\cdot x_1^{k\cdot a_{k1}}...x_n^{k\cdot a_{kn}} \right) \quad (15)$$

For convenience we will use the following notation for multinomial coefficients.

$$\binom{\alpha_k}{\vec{a}_k} := \frac{\alpha_k!}{a_{k1}!a_{k2}!...a_{kn}!} \quad (16)$$

$$\vec{a}_k := \{a_{k1}, a_{k2}, ..., a_{kn}\}$$

$$a_k := a_{k1} + a_{k2} + ... + a_{kn}$$

In the next step we expand the product of multinomial factors by applying the generalized distribution law (17).

$$\prod_{k=1}^n \left(\sum_{h_k=1}^{m_k} q_{h_k} \right) = \sum_{h_1=1}^{m_1} ... \sum_{h_n=1}^{m_n} \left(\prod_{k=1}^n q_{h_k} \right) = \sum_{...,h_k=1,...}^{...,m_k,...} \left(\prod_{k=1}^n q_{h_k} \right) \quad (17)$$

Setting $q_{h_k} = \binom{\alpha_k}{\vec{a}_k} x_1^{k\cdot a_{k1}}\cdot x_2^{k\cdot a_{k2}}...x_n^{k\cdot a_{kn}}$ and $h_k = \vec{a}_k$ with $\sum_{j=1}^n a_{kj} = \alpha_k$ yields

$$S(\vec{\alpha};\vec{x}) = \prod_{k=1}^n \left(\sum_{\substack{a_{k1},...,a_{kn} \\ \sum_{j=1}^n a_{kj}=\alpha_k}} \binom{\alpha_k}{\vec{a}_k}\cdot x_1^{k\cdot a_{k1}}...x_n^{k\cdot a_{kn}} \right)$$

$$= \sum_{\substack{...,a_{kj},... \\ \sum_{j=1}^n a_{kj}=\alpha_k}} \left(\prod_{k=1}^m \binom{\alpha_k}{\vec{a}_k}\cdot x_1^{k\cdot a_{k1}}...x_n^{k\cdot a_{kn}} \right) \quad (18)$$

Each summand of (18) consists of a product with numerous $x_j^{k\cdot a_{kj}}$ factors. These factors are indexed by k and j so that one can rewrite the $x_j^{k\cdot a_{kj}}$ products in condensed form as a double product.

$$S(\vec{\alpha};\vec{x}) = \sum_{\substack{...,a_{kj},... \\ \sum_{j=1}^n a_{kj}=\alpha_k}} \left(\prod_{k=1}^m \binom{\alpha_k}{\vec{a}_k} \right)\cdot\left(\prod_{j=1}^n \prod_{k=1}^m x_j^{k\cdot a_{kj}} \right) \quad (19)$$

We leave the multiplication with respect to j untouched and convert the product with respect to k into a summation of the exponent of x_j.

$$S(\vec{\alpha};\vec{x}) = \sum_{\substack{...,a_{kj},... \\ \sum_{j=1}^n a_{kj}=\alpha_k}} \left(\prod_{k=1}^m \binom{\alpha_k}{\vec{a}_k} \right)\cdot\left(\prod_{j=1}^n x_j^{(\sum_{k=1}^m k\cdot a_{kj})} \right) \quad (20)$$

We abbreviate the exponent of each x_j with $\mu_j := \sum_{k=1}^m k\cdot a_{kj}$. Then we collect all monomials exhibiting the same exponent pattern. We, thus, split the summation in (18) into a summation over μ_j and a subsummation over all a_{kj} with $\sum_{j=1}^n a_{kj} = \alpha_k$ as well as $\sum_{k=1}^m k\cdot a_{kj} = \mu_j$.

$$S(\vec{\alpha};\vec{x}) = \sum_{\substack{\mu_j \\ \sum_{j=1}^n \mu_j=\sigma}} \underbrace{\left(\sum_{\substack{...,a_{kj},... \\ \sum_{j=1}^n a_{kj}=\alpha_k \\ \sum_{k=1}^m k\cdot a_{kj}=\mu_j}} \prod_{k=1}^m \binom{\alpha_k}{\vec{a}_k} \right)}_{\chi^{(\mu_1,\mu_2,...,\mu_n)}_{(\alpha_1,\alpha_2,...,\alpha_m)}} \cdot \left(\prod_{j=1}^n x_j^{\mu_j} \right) \quad (21)$$

Note that given $\mu_j = \sum_{k=1}^m k\cdot a_{kj}$ the sum of exponents $\sum_{j=1}^n \mu_j$ is a constant value $\sigma := \sum_{k=1}^m k\cdot\alpha_k$ as can be seen in (22), where $\sum_{j=1}^n a_{kj} = \alpha_k$ encountered in (16) is used.

$$\sigma = \sum_{j=1}^n \mu_j = \sum_{j=1}^n \sum_{k=1}^m k\cdot a_{kj} = \sum_{k=1}^m k\cdot\left(\sum_{j=1}^n a_{kj} \right) = \sum_{k=1}^m k\cdot\alpha_k \quad (22)$$

Comparing the expanded sum in (21) with the polynomial in (12), we can extract the terms for the coefficients $\chi^{(\mu_1,\mu_2,...,\mu_n)}_{(\alpha_1,\alpha_2,...,\alpha_m)}$.

$$\chi^{(\mu_1,\mu_2,...,\mu_n)}_{(\alpha_1,\alpha_2,...,\alpha_m)} := \sum_{\substack{...,a_{kj},... \\ \sum_{j=1}^n a_{kj}=\alpha_k \\ \sum_{k=1}^m k\cdot a_{kj}=\mu_j}} \prod_{k=1}^m \binom{\alpha_k}{\vec{a}_k} \quad (23)$$

Hence, to find a coefficient $\chi^{(\mu_1,\mu_2,...,\mu_n)}_{(\alpha_1,\alpha_2,...,\alpha_m)}$ for a given tuple $\vec{\alpha}$ and $\vec{\mu}$, we have to sum over products of multinomials for all nonnegative integers a_{jk} satisfying the conditions $\sum_{j=1}^n a_{kj} = \alpha_k$ and $\sum_{k=1}^m k\cdot a_{kj} = \mu_j$.

3.3. Algorithm. The heart of the algorithm, introduced here, consists of a recursive strategy that generates all admissible sets of nonnegative integers a_{kj}. To explain the strategy, we rewrite the conditions $\sum_{j=1}^n a_{kj} = \alpha_k$ and $\sum_{k=1}^m k\cdot a_{kj} = \mu_j$ in the form of a table (24).

The kth row $k\cdot a_{k1}$ $k\cdot a_{k2}$ ... $k\cdot a_{kn}$ $\overset{\Sigma}{=}$ α_k stands for $\sum_{j=1}^n k\cdot a_{kj} = \alpha_k$. Accordingly, the jth column stands for $\sum_{k=1}^m k\cdot a_{kj} = \mu_j$.

The α_k's and μ_k's and m and n are given. The nonnegative integers a_{kj} must be found. The strategy consists of two recursions, one nested within the other. The outer recursion

$$\begin{array}{ccccccccc}
a_{11} & a_{12} & \cdots & a_{1j} & \cdots & a_{1n} & \overset{\Sigma}{=} & \alpha_1 \\
2\cdot a_{21} & 2\cdot a_{22} & \cdots & 2\cdot a_{2j} & \cdots & 2\cdot a_{2n} & \overset{\Sigma}{=} & \alpha_2 \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \vdots \\
k\cdot a_{k1} & k\cdot a_{k2} & \cdots & k\cdot a_{kj} & \cdots & k\cdot a_{kn} & \overset{\Sigma}{=} & \alpha_k \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \vdots \\
m\cdot a_{m1} & m\cdot a_{m2} & \cdots & m\cdot a_{mj} & \cdots & m\cdot a_{mn} & \overset{\Sigma}{=} & \alpha_m \\
\overset{\Sigma}{=} & \overset{\Sigma}{=} & \cdots & \overset{\Sigma}{=} & \cdots & \overset{\Sigma}{=} & \ddots & \overset{\Sigma}{=} \\
\mu_1 & \mu_2 & \cdots & \mu_j & \cdots & \mu_n & \overset{\Sigma}{=} & \sigma
\end{array} \tag{24}$$

reduces the equation table (24) by one row at a time, from bottom to top. The inner recursion eliminates one column in each recursion step, doing so from right to left.

3.4. Outer Recursion. We commence with the outer recursion. Given a tuple $\vec{\alpha}$ and a tuple $\vec{\mu}$, we first determine the maximum nonnegative integer values admissible for the a_{mj} entries in row m. Since $m{\cdot}a_{mj}$ is part of the jth column that forms the sum μ_j, and since all summands in that sum are nonnegative, we must observe the inequality $m{\cdot}a_{mj} \le \mu_j$. Hence, the largest possible value for a_{mj} is given by $b_j^{(m)} = \lfloor \mu_j/m \rfloor$. The expression $\lfloor \mu_j/m \rfloor$ stands for the greatest integer value smaller than or equal to μ_j/m. Equipped with α_m and tuple $\vec{b}^{(m)}$, the inner recursion, defined in a separate function and described below, generates all admissible tuples $\vec{a}_m$ with $a_{mj} \le b_j^{(m)}$ and $\sum_{j=1}^{n} a_{mj} = \alpha_m$. For each of these admissible tuples $\vec{a}_m$, we now perform a recursion step from row m to $(m - 1)$. Be aware that at this point the recursion splits into one or more branches, one branch for each admissible tuple $\vec{a}_m = \{a_{m1}, a_{m2}, ..., a_{mn}\}$. First, we store $\vec{a}_m$ as part of one of the final solutions. Second, we subtract $m{\cdot}\vec{a}_m$ from $\vec{\mu}$ and keep the result as the new tuple $\vec{\mu}$. Third, we remove the mth row from the equation table (24) and repeat the above procedure, this time for row $(m - 1)$.

There are two scenarios which cause the outer recursion to halt. In the first scenario, the recursion reaches the first row with $m = 1$. We thus obtain a complete set of $\{\vec{a}_1, \vec{a}_2, ..., \vec{a}_m\}$ values with which we can calculate a summand in (11). In the second scenario, the inner recursion renders no solution for $\vec{a}_k$ in a kth outer recursion step. Then the prospective solution $\{\vec{a}_1, \vec{a}_2, ..., \vec{a}_k\}$ of the affected outer recursion branch must be discarded.

Remember that the outer recursion splits into one or more branches at each recursion step. Hence, we obtain a tree structure of solutions $\vec{a}_k$. For every leaf of this tree structure we can extract a complete solution $\{\vec{a}_1, \vec{a}_2, ..., \vec{a}_m\}$ by collecting tuples $\vec{a}_k$ at each branch as we go from the root to a given leaf. The expansion coefficient χ is zero, if the outer recursion fails to produce any solution.

3.5. Inner Recursions: Bounded Distributions. We still have to describe the inner recursion. As noted above, the inner recursion starts with an integer value for α and an n-tuple of nonnegative integer upper bounds $\vec{b}$. Here we omit index k for easier reading. The inner recursion has to render all tuples $\vec{a}$ that comply with $\sum_{j=1}^{n} a_j = \alpha$ and $a_j \le b_j$ for all $j = 1, ..., n$. Setting it differently, we are looking for all distributions of α items into n bins with upper bounds b_j. We therefore refer to the inner recursion as the *bounded distribution function*. We start the inner recursion by calculating the range of admissible values of a_1; we then proceed to deal with a_2 and so forth. An upper bound for admissible a_1 values is given either by α, due to $\sum_{j=1}^{n} a_j = \alpha$, or by b_1, whichever is smaller. Consequently, $a_1 \le \min(\alpha, b_1)$. A lower bound for a_1 is 0, since a_1 has to be a nonnegative integer. Furthermore, a lower bound for a_1 is given when all other summands $a_{j\ge 2}$ assume their maximum values $b_{j\ge 2}$. Then a_1 has to assume its smallest value to allow for $a_1 + \sum_{j=1}^{n} b_j = \alpha$. Thus, $a_1 \ge \alpha - \sum_{j=2}^{n} b_j$. The lower bound is therefore given by $a_1 \ge \max(0, \alpha - \sum_{j=2}^{n} b_j)$. Each integer value within the interval $[\max(0, \alpha - \sum_{j=2}^{n} b_j), \min(\alpha, b_1)]$ is admissible for a_1. For each admissible a_1 value we perform a separate recursion step. Hence, the inner recursion may split into several branches, just as the outer recursion. For every branch we save the corresponding a_1 value and remove it from the sum $\sum_{j=1}^{n} a_j = \alpha$ by subtracting a_1 from α. Then, we repeat the inner recursion for a_2 by setting $\vec{a} = \{a_2, a_3, ..., a_n\}$ and $\vec{b} = \{b_2, b_3, ..., b_n\}$, and taking the reduced α into account. The inner recursion halts with $a_n = \alpha - \sum_{j=1}^{n-1} a_j$. The inner recursion never stops prematurely if a range of admissible values for a_1 exists. This is obvious since we have chosen the lower interval boundary of a_1 to allow for a solution of all a_j's. If one cannot find an admissible value for a_1, no solution exists.

Again, as with the outer recursion, we obtain a tree structure of solutions. For every leaf of this tree structure we can extract a complete solution $\vec{a} = \{a_1, a_2, ..., a_n\}$ by collecting the admissible a_j values at each branch going from the root of the tree structure to the given leaf.

3.6. Assembly of Expansion Coefficients. We now return to (11). Any generating function that we encounter consists of summands $S(\vec{\alpha};\vec{x})$ with different tuples $\vec{\alpha}$. We obtain the expansion coefficients $\gamma^{(\vec{\mu})}$ of such a generating function by summing up the respective multiples of all corresponding summand coefficients $\chi_{(\vec{\alpha})}^{(\vec{\mu})}$ with the same exponent signature $(\vec{\mu})$.

$$\begin{aligned}
\mathrm{GF}(\vec{x}) &= cS(\vec{\alpha};\vec{x}) + c'S(\vec{\alpha}';\vec{x}) + c''S(\vec{\alpha}'';\vec{x}) + ... \\
&= \sum_{\mu_1,\mu_2,...,\mu_n} \gamma^{(\mu_1,\mu_2,...,\mu_n)}{\cdot}x_1^{\mu_1}{\cdot}x_2^{\mu_2}...x_n^{\mu_n} \qquad (25) \\
\gamma^{(\vec{\mu})} &:= c\chi_{(\vec{\alpha})}^{(\vec{\mu})} + c'\chi_{(\vec{\alpha}')}^{(\vec{\mu})} + c''\chi_{(\vec{\alpha}'')}^{(\vec{\mu})} + ...
\end{aligned}$$

With this result we conclude the discussion of our Pólya polynomial expansion algorithm. The expansion coefficients $\gamma^{(\vec{\mu})}$ of $\mathrm{GF}(\vec{x})$ in (25) render the number of equivalence classes in $\Omega_G(X)$ and thus the isomer count for diamutamers with specified ligand multiplicity.

4. LIBRARIES WITH SPECIFIED LIGAND ASSORTMENT PATTERNS

In the previous section we performed the enumeration of isomer ensembles. Every diamutamer consisted of the same central skeleton and the same set of ligands. We now turn to the enumeration of diamutamer libraries. The ensembles (or libraries) to be enumerated consist of diamutamers that exhibit the same central compound but not necessarily the same set of ligands. Instead we assume an unlimited supply of substituents with a specified range of ligand types. The substituents of the diamutamers are all taken from the same supply, but the ligand selection may be different. Within one library one may therefore encounter diamutamers with different molecular formulas, hence chemical compounds that are not isomeric.

Table 2. Substituent Patterns and the Corresponding Combinatorial Factors of Methane (First Two Columns) and The Number of Isomers for the Substituent Pattern in Column One (Last Column)

pattern	multiplication factor	no. of isomers	pattern	multiplication factor	no. of isomers
CX_4	$\binom{n}{1}$	1	CX_2YZ	$\binom{n}{1}\binom{n-1}{2}$	1
CX_3Y	$\binom{n}{1}\binom{n-1}{1}$	1	CXYZW	$\binom{n}{4}$	2
CX_2Y_2	$\binom{n}{2}$	1			

In this section we enumerate libraries consisting of diamutamers with a specified ligand assortment pattern. The emphasis lies on the word *pattern*. We already considered diamutamers with a specified ligand assortment in the previous section. A ligand assortment pattern is the partition of attached substituents into subsets. Each subset denotes a substituent group of an unspecified ligand type. Different subsets represent different ligand types. We thereby limit an ensemble of diamutamers to compounds that exhibit a certain molecular formula pattern. All possible ligand assortment patterns for methane are listed in the first column of Table 2 as an example.

Given a ligand assortment pattern and a supply of ligand types, one has to be aware of the many combinations with which the available ligand types can be distributed into the ligand assortment pattern. Determining this number of combinations we can convert an enumeration of libraries with ligand assortment patterns into an enumeration of isomer ensembles consisting of diamutamers with a specified set of ligands.

We solve this task step by step considering more and more assortment patterns.

First, we address an assortment pattern that equally partitions the set of attached substituents into k different ligand types, each partition encompassing m substituents. Obviously, k needs to be smaller than n, the number of supplied ligand types. A substituent selection such as $A_mB_mC_m...K_m$ satisfying the pattern is uniquely specified by the k-element subset A, B, C, ..., K. The order of ligand types in the selection is irrelevant, since a ligand assortment pattern refers to the Chemical Abstract Formula Index, which may have a sorted notation convention, but which does not imply any structural order (e.g., formula $C_6H_2Br_2Cl_2$ is equivalent to $C_6H_2Cl_2Br_2$ or $C_6Br_2Cl_2H_2$). To find the number of possible substituent selections such as $A_mB_mC_m...K_m$, we have to determine the number of k-element subsets in the set of n ligand types. The solution to this well-known problem is the binomial coefficient $\binom{n}{k}$. Consequently, having access to a supply of n ligand types, one can create $\binom{n}{k}$ different selections of k elements. Each ligand selection leads to the same diamutamer enumeration, since these enumerations are independent of the particular ligand types. Only the abstract partitions into different ligand types are relevant.

Second, we approach assortment patterns with partitions of different sizes via recursion. We begin by considering partitions of a certain size only. We assume to find k_1 of these partitions. We can then distribute k_1 of the n available ligand types into the k_1 partition bins. As mentioned above, there are $\binom{n}{k_1}$ combinations. We initiate a recursion step by extracting all partitions of another size. If there are k_2 partitions of this other size, we can distribute k_2 of the $n - k_1$ ligand types not yet in use. The resulting combinatorial factor is $\binom{n-k_1}{k_2}$. In the next recursion step, we extract k_3 partitions of still another size, distribute the $n - (k_1 + k_2)$ unused ligand types, and obtain the combinatorial factor $\binom{n-(k_1+k_2)}{k_3}$. We can proceed this way until all partition sizes are accounted for.

The resulting combinatorial factors for the methane example are listed in the second column of Table 2. We can infer from the fourth row of Table 2 that one can assemble $\binom{n}{1}\binom{n-1}{2}$ methane diamutamers of type CX_2YZ given a supply of n ligand types. According to the last row, there are two $\binom{n}{4}$ methane diamutamers of type CXYZW. Note that in the last case every ligand selection amounts to two diamutamers.

We can utilize this library enumeration scheme also for libraries with no or partially specified ligand assortment patterns. We simply have to sum over all admissible assortment patterns. This approach may, however, be very tedious, if the number of admissible assortment patterns is large. For example, to enumerate an unrestricted library for methane, we must sum over all five patterns listed in Table 2, obtaining $\binom{n}{1} + \binom{n}{1}\binom{n-1}{1} + \binom{n}{2} + \binom{n}{1}\binom{n-1}{2} + 2\binom{n}{4} = n^2(11 + n^2)/12$ diamutamers. This might be acceptable, but the enumeration of the C_{60} fullerene diamutamer library results in a summation of over 966 467 ligand assortment patterns,[13] a task that takes today's personal computers days to complete. In such a case the algorithm introduced in the next section is far superior.

5. LIBRARIES WITH A SPECIFIED NUMBER OF ATTACHED LIGANDS

We continue with the enumeration of diamutamer libraries. However, we no longer specify the ligand assortment pattern. The only constraint upon the library ensemble is now the number of attached ligands. We achieve an enumeration of such an ensemble not via Pòlya polynomials as in sections 3 and 4, but through a new algorithm counting the number of fix points X_{p_g} in the Cauchy−Frobenius lemma (3) under the new constraint.

We start with the special case in which the number of attached ligands s is the same as the number of binding sites $|B|$. This scenario is equivalent to the enumeration of unrestricted libraries. We recall the construction of invariant diamutamer maps ξ as discussed in section 2.4 and Figure 2. A map ξ is invariant under a site permutation $\tilde{p}_g$, if all the sites within a permutation cycle are mapped onto the same ligand type. Hence, we can choose to map every permutation cycle, or rather its elements, onto $|L|$ different ligand types. We use $\eta(p_g)$ to denote the number of permutation cycles in B. The number of p_g-invariant maps is then based on $\eta(p_g)$ ligand type choices, amounting to $|L|^{\eta(p_g)}$ possibilties. Inserting this number of fix points $|X_{p_g}|$ into the Cauchy−Frobenius lemma (3), we derive

$$|\Omega_G(X)| = \frac{1}{|PG|}\sum_{p_g\in P_G} |L|^{\eta(p_g)} \qquad (26)$$

(26) is the known enumeration formula for unrestricted diamutamer libraries.

Pòlya indices (2) are very helpful when constructing (26) for a particular skeleton. Recall that every Pòlya index variable $f_{|c_i(p_g)|}$ denotes a permutation cycle $c_i(p_g)$. Replacing $f_{|c_i(p_g)|}$ by $|L|$, we obtain $|L|^{\eta(p_g)}$ for $\prod_i f_{|c_i(p_g)|}$ and (26) for a Pòlya index (2).

The enumeration of diamutamer libraries becomes more subtle if one reduces the number of substituents attached to the central skeleton. We introduce a new ligand type with the symbol V, representing *vacancies*, to account for this modification. Binding sites mapped upon V are now considered unsubstituted.

As with unrestricted libraries we have to count the invariant maps $\xi{:}B \rightarrow L \cup \{V\}$, but with the proviso that $|B| - s$ binding sites are mapped onto V. Hence, the set of admissible diamutamer maps is $(L|_s \cup \{V\})^B := \{\xi|\xi{:}B \rightarrow (L \cup \{V\}) \vee |\xi^{-1}(L)| = s\}$. The number of occupied binding sites is s, fixed by the condition $|\xi^{-1}(L)| = s$.

To determine the number of diamutamers, we have to find the number of fix points $|((L|_s \cup \{V\})^B)_{p_g}|$ for each site permutation p_g, the same counting strategy as before. Now, however, we have to ensure that for all diamutamers exactly $(|B| - s)$ sites are vacant, i.e., mapped upon V.

Under this constraint, we can only find invariant diamutamer maps ξ for a site permutation p_g, if p_g exhibits a subset of permutation cycles with altogether exactly $(|B| - s)$ elements. The elements of these permutation cycles can then be invariantly mapped upon the $(|B| - s)$ ligands of type V. After all, any permutation cycle with only part of its elements mapped upon V would not be invariant under permutation p_g, hence the guideline "all cycle elements or none".

Once again we use the Pòlya index variables $f_{|c_i(p_g)|}$ to denote the permutation cycles $c_i(p_g)$ of a site permutation p_g. The product $\prod_i f_{|c_i(p_g)|}$ represents the number and length of all cycles of site permutation p_g faithfully. We rewrite the product of index variables $f_{|c_i(p_g)|}$ by gathering permutation cycles of the same length $l = |c_i(p_g)|$, and denoting their multiplicity by an exponent e. We thereby obtain an expression of the form $f_{l_1}^{e_1}\cdot f_{l_2}^{e_2}...f_{l_j}^{e_j}$. We thus characterize a permutation p_g with e_1 cycles of length l_1, ..., and e_j cycles of length l_j.

To determine the number of fix points $|((L|_s \cup \{V\})^B)_{p_g}|$ by recursive means, we introduce a new function

$$\mathrm{NF}_s{:}f_{l_1}^{e_1}...f_{l_j}^{e_j} \rightarrow \mathrm{IN}$$

The function NF_s will render the number of fix points under a permutation with a Pòlya index variable product $f_{l_1}^{e_1}...f_{l_j}^{e_j}$.

As a recursion anchor we consider site permutations p_g with permutation cycles of equal length l. Such site permutations are represented by a Pòlya index variable product f_l^e. $\mathrm{NF}_s(f_l^e)$ is equal to 0 if no invariant map can be formed. This is the case if the s occupied binding sites cannot be assigned to a subset of complete permutation cycles as described in Figure 2. Thus, a necessary condition for any invariant assignment is $s \bmod l = 0$. The number of occupied binding sites s has to be divisible by the cycle length l. Furthermore, there should not be more binding sites s than the $(e\cdot l)$ elements of all permutation cycles: $s \le el$.

Once these preconditions are met, there are e permutation cycles, s/l of which are mapped onto ligand types in L. The rest will be assigned to V. Obviously there are $\binom{e}{s/l}$ possibilities to select s/l permutation cycles out of a total of e cycles. For each of these $\binom{e}{s/l}$ selections there are $|L|^{s/l}$ possibile assignments of the s/l cycles to the $|L|$ ligand types in L. This is the same reasoning as for unrestricted libraries. Hence, we obtain the following recursion anchor:

$$\mathrm{NF}_s(f_l^e) = \begin{cases} \binom{e}{s/l}\,|\mathrm{L}|^{s/l}, & \text{if } s \bmod l = 0 \vee s \le el,\\ 0, & \text{otherwise} \end{cases}$$

To apply the function NF_s to permutations p_g with j different cycle lengths, we construct a recursion step that reduces the case with j different cycle lengths to the case with $(j - 1)$ different cycle lengths. The permutations p_g with j different cycle lengths will exhibit a Pòlya index variable product $f_{l_1}^{e_1}...f_{l_{j-1}}^{e_{j-1}}\cdot f_{l_j}^{e_j}$. We begin with the ligand assignments to the permutation cycle of the last Pòlya variable $f_{l_j}^{e_j}$. The ligand assignments to the remaining permutation cycles will be handled by the following recursion steps.

Up front, we do not know how many of the $f_{l_j}^{e_j}$ cylces will be assigned to ligand types in L, and how many will be assigned to V, and, thus, stay vacant. We therefore have to sum over all possible scenarios. We can assign 0 or up to e_j but no more than $\lfloor s/l_j \rfloor$ cycles to ligand types in L. Assume we take an integer k with $0 \le k \le \min(e_j, \lfloor s/l_j \rfloor)$, and map k permutation cycles of length l_j onto L. We obtain, as in the case of the recursion anchor, $\binom{e_j}{k}|L|^k$ possible invariant maps. However, we still have the cycles of length l_1 to l_{j-1}. The number of invariant maps for these cycles is evaluated by the following recursion steps, which we obtain via $\mathrm{NF}_{s-k}(f_{l_1}^{e_1}...f_{l_{j-1}}^{e_{j-1}})$. Note that the number of binding sites to be occupied has been reduced from s to $(s - k)$. The (partial) maps of the permutation cycles of different lengths will combine to complete invariant maps. Hence, the enumerations of all the possible partial maps within a recursion have to be multiplied. We thus multiply the $\mathrm{NF}_{s-k}(f_{l_1}^{e_1}...f_{l_{j-1}}^{e_{j-1}})$ possibilities onto the number of invariant maps with cycles of length l_j. Summing over all admissible values of k, we derive the recursion step

$$\mathrm{NF}_s(f_{l_1}^{e_1}...f_{l_{j-1}}^{e_{j-1}}\cdot f_{l_j}^{e_j}) = \sum_{k=0}^{\min(\lfloor s/l_j \rfloor, e_j)} \binom{e_j}{k}|L|^k \mathrm{NF}_{s-k}(f_{l_1}^{e_1}...f_{l_{j-1}}^{e_{j-1}}) \quad (27)$$

Obviously the recursion step (27) reduces the number of cycles with different length. Consequently, the recursion terminates with the anchor derived above.

So far, function NF_s helps us to determine the number of fix points $|((L|_s \cup \{V\})^B)_{p_g}|$ for a single site permutation p_g. According to the Cauchy–Frobenius lemma (3), we have to perform this calculation for every site permutation p_g of group P_G, add the results, and divide it by $|P_G|$ to complete the diamutamer library enumeration. We can simplify this procedure. Note that a Pòlya index is a linear combination of Pòlya index variable products representing the summation and division just mentioned above. Hence, we simply define function NF_s as a linear operator and apply it to a Pòlya

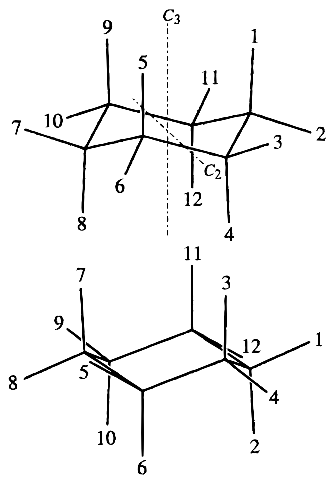


Figure 3. Cyclohexane in its chair conformation. The numbers enumerate the hydrogen ligands, the possible substituent sites. The two displayed conformations are related by a ring flip. The conformations are invariant under rotations around a 2-fold axis, C_2, and a 3-fold axis, C_3, as well as under reflection σ between the foreground and background.

index as a whole. We instantly obtain the result of the diamutamer library enumeration.

$$|\Omega_G(L|_s \cup \{V\})| = \mathrm{NF}_s\left(\frac{1}{|P_G|}\sum_{p_g\in P_G}\left(\prod_i f_{|c_i(p_g)|}\right)\right) \quad (28)$$

6. THE PROGRAM

The program package ISOMERS.M is written in MATHEMATICA[4] and readily available to MATHEMATICA users on the Internet.[2] We will try to find a sponsor for a MATHEMATICA-driven Web page for those who do not have access to the MATHEMATICA kernel needed to run ISOMERS.M. The core of ISOMERS.M consists of an adaptation, a refinement, and a considerable extension of a program[14] given earlier in FORTRAN. ISOMERS.M automates all the calculations and algorithms introduced in this paper. The program makes intensive use of the powerful list and symbolic manipulating features of the computer algebra language MATHEMATICA. This allows us to enhance the capacity of the algorithms. We can render symbolic as well as arbitrarily large, exact numeric results. ISOMERS.M comes with detailed on-line help and explanations built into MATHEMATICA's Help Browser.

Here, we provide some sample calculations to illustrate the handling of the program. We take cyclohexane in its chair conformation (Figure 3) to compare our results with those of J. Leonard.[15]

After loading the ISOMERS.M package, we start by providing the substituent site permutations according to the three categories (1) rotations, (2) reflection/inversion, and (3) conformational transformations.

The symmetry group of cyclohexane can be generated with four generators. Two of them are rotations, a 2-fold one with site permutation (8, 7, 6, 5, 4, 3, 2, 1, 12, 11, 10, 9), and a 3-fold one with site permutation (5, 6, 7, 8, 9, 10, 11, 12, 1, 2, 3, 4). Furthermore, cyclohexane is invariant under reflection with site permutation (1, 2, 11, 12, 9, 10, 7, 8, 5, 6, 3, 4). Besides these point group symmetries, cyclohexane also exhibits a conformational symmetry transformation. Figure 3 displays the cyclohexane chair conformation before and after a ring flip. A ring flip in combination with a 60° rotation around the 3-fold vertical axis is our fourth symmetry transformation with site permutation (3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 1, 2). We implement these properties of cyclohexane with the command (note that the subscript σ identifies the site permutation of a reflection and c the site permutation of a conformational transformation)

```
In[1]:=
   DefineParentCompound[
     Cyclohexane,
     {ℛ[8,7,6,5,4,3,2,1,12,11,10,9],
      ℛ[5,6,7,8,9,10,11,12,1,2,3,4],
      ℛσ[1,2,11,12,9,10,7,8,5,6,3,4],
      ℛc[3,4,5,6,7,8,9,10,11,12,1,2]}
   ]
```

Now, Cyclohexane is a MATHEMATICA symbol with which we can retrieve the symmetry generators from our database. To check the input, we verify the symmetry of cyclohexane.

```
In[2]:= PointGroupReport[Cyclohexane]
```

The group cycles of the point group are :

$(\mathcal{E}),(3C_2),(3\sigma),\left(\sigma C_6^3\right),\left(\sigma C_6,\sigma C_6^5\right),\left(C_6^2,C_6^4\right)$

The corresponding point groups are : D_{3d}, D_{3h}.

The group cycles including conformational transformations are :

$(\mathcal{E}),(3C_2),(3C_2),(3\sigma),(3\sigma),\left(C_6^3\right),\left(\sigma C_6^3\right),\left(\sigma C_6^3\right),\left(C_6,C_6^5\right),\left(\sigma C_6,\sigma C_6^5\right),\left(\sigma C_6,\sigma C_6^5\right),\left(C_6^2,C_6^4\right)$

Such a group cycle structure would correspond to : D_{6h}.

The automatically identified point groups D_{3d} and D_{6h} agree with those in ref 15. However, some Pólya cycle indices determined with our program contradict those in ref 15. Simple consistency checks[16] indicate that a few results of ref 15 must be partially wrong.

The Pólya cycle index for a "rigid" cyclohexane with point group symmetry D_3 and without any conformational ring flips is (different in ref 15)

```
In[3]:= CycleIndex[
           Cyclohexane,
           Reflections → False,
           Conformations → False
         ]
```

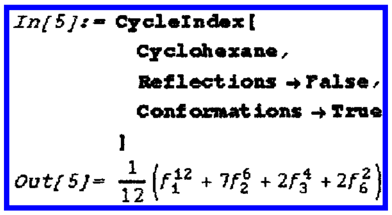
$\mathrm{Out[3]}= \frac{1}{6}\left(f_1^{12}+3f_2^6+2f_3^4\right)$

Including reflections as in the point group D_{3d}, we obtain (same as in ref 15)

```
In[4]:= CycleIndex[
           Cyclohexane,
           Reflections → True,
           Conformations → False
         ]
```

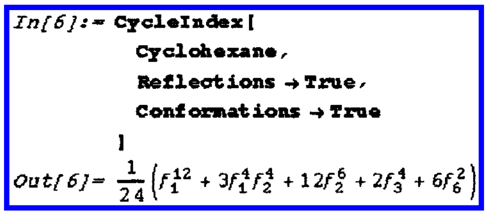
$\mathrm{Out[4]}= \frac{1}{12}\left(f_1^{12}+3f_1^4f_2^4+4f_2^6+2f_3^4+2f_6^2\right)$

The Pólya cycle index for a "nonrigid" cyclohexane with a symmetry corresponding to D_6 is (same as in ref 15)



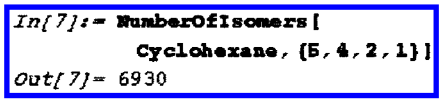
```
In[5]:= CycleIndex[
           Cyclohexane,
           Reflections → False,
           Conformations → True
         ]
```
Out[5]= $\frac{1}{12}\left(f_1^{12} + 7f_2^6 + 2f_3^4 + 2f_6^2\right)$

Including reflections as in the point group D_{6h}, we obtain (different in ref 15)



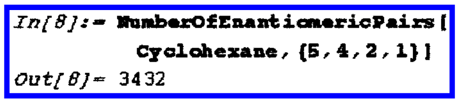
```
In[6]:= CycleIndex[
           Cyclohexane,
           Reflections → True,
           Conformations → True
         ]
```
Out[6]= $\frac{1}{24}\left(f_1^{12} + 3f_1^4f_2^4 + 12f_2^6 + 2f_3^4 + 6f_6^2\right)$

Due to the discrepancy in Pólya cycle indices, we obviously obtain different results in our isomer enumerations. For example, if we count the number of isomers of cyclohexane with 12 ligands of type A, B, D, and E, we calculate for $C_6A_5B_4D_2E$



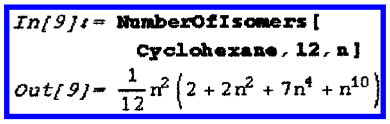
```
In[7]:= NumberOfIsomers[
            Cyclohexane, {5,4,2,1}]
Out[7]= 6930
```

Here we are still in agreement, since the Pólya cycle index of *Out[5]* is in agreement with ref 15. However, we find different results for the number of enantiomeric pairs, since *Out[6]* contradicts ref 15.



```
In[8]:= NumberOfEnantiomericPairs[
            Cyclohexane, {5,4,2,1}]
Out[8]= 3432
```

We conclude the cyclohexane example with a symbolic calculation related to ref 17. The number of compounds with a cyclohexane skeleton assembled with 12 ligands of at most *n* different types is



```
In[9]:= NumberOfIsomers[
            Cyclohexane, 12, n]
```
Out[9]= $\frac{1}{12}n^2\left(2 + 2n^2 + 7n^4 + n^{10}\right)$

7. RESULTS AND DISCUSSION

The enumeration of diamutamers is not only of particular interest in spectroscopy and theoretical chemistry,[18] but also in the expanding field of combinatorial chemistry to estimate library sizes.[19]

With the program described above one can derive general formulas for the number of isomers of multisubstituted derivatives. This has been done for several organic molecules in ref 17. Additional examples of larger molecules can be found at our Web site.[20] There, we provide isomer enumeration results for many types of fullerenes, which are of general interest, and which were previously treated to lesser extent due to restraints in computer memory and time.[21] The algorithm described in section 5 of this paper drastically reduces the time needed to derive symbolic formulas for any diamutamer. For example, initially the derivation of all isomer enumeration formulas of C_{60} fullerene[17] took several days. This initial approach was based on the algorithm described in section 3. The algorithm of section 5 handles the task in less than 30 s.[22]

With this program we wish to provide chemists with an easy to use tool for diamutamer enumeration. We hope the algorithms will prove to be beneficial to a large number of scientists.

REFERENCES AND NOTES

(1) *IUPAC Commission on Nomenclature of Organic Chemistry, Section E: Pure And Applied Chemistry*; Pergamon Press: Oxford, 1976; Vol. 45, pp 11−30.

(2) ISOMERS.M ftp site and pointers to related Web sites are located at http://www-orgc.tu-graz.ac.at/hoegroup.

(3) Pólya, G. Kombinatorische Anzahlbestimmungen für Gruppen, Graphen und Chemische Verbindungen. *Acta Math.* **1937**, *68*, Stockholm 145−254. In English: Pólya, G. *Combinatorial Enumeration of Groups, Graphs, and Chemical Compounds*; Springer: New York, 1987.

(4) Wolfram, S. *The Mathematica Book*, 4th ed.; Wolfram Media/ Cambridge University Press: Cambridge, U.K., 1999.

(5) Pilati, T.; Forni, A. SYMMOL: a program to find the maximum symmetry group in an atom cluster, given a prefixed tolerance. *J. Appl. Crystallogr.* **1998**, *31*, 503−504.

(6) We regret the confusion due to standard group theoretical and chemical notations. Here, C_n stands for an *n*-fold rotation axis, which represents a group cycle of length *n*.

(7) In chemistry the tetrahedral point group T_d usually includes an S_4-symmetry transformation that, for example, with respect to Figure 1, consists of a 90° rotation around C_2 and a reflection σ_h across a plane that lies parallel to edge (1−2) and edge (3−4). This reflection σ_h is only a symmetry transformation in combination with the corresponding 90° rotation. It should not be confused with reflection σ. An S_4 symmetry transformation can also be generated via a reflection σ and a 120° rotation around C_3. We therefore do not introduce reflection σ_h or S_4.

(8) The notations found in the literature may differ slightly: $C_n^p = C_n^{n-p}$ and $C_n^p = C_{n/q}^{p/q}$, if *q* is a common denominator for *n* and *p*.

(9) Schoenflies, A. *Theorie der Kristallstruktur*; Bomtraeger: Berlin, 1923.

(10) Kerber, A. *Applied Finite Group Actions*; Algorithms & Combinatorics 19; Springer-Verlag: New York, 1999; Chapter 2.

(11) The distributative law of Boolean logic is $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$. Note the analogy to basic algebra: $a\cdot(b + c) = a\cdot b + a\cdot c$.

(12) One can include vacant substituent sites in an isomer enumeration by simply introducing a *vacant* substituent type, typically hydrogen.

(13) 966 467 is the number of integer partitions of 60. This is the number of different substituent patterns for a C_{60} fullerene.

(14) Dolhaine, H. A Computer Program for the Enumeration of Substitutional Isomers. *Comput. Chem.* **1981**, *5*, 41−48. See also the note by Dolhaine, H.; Garavelli, J. S.; Leonard, J. E. Comments on Papers Concerning Computer Enumeration of Permutation Isomers. *Comput. Chem.* **1986**, *10*, 239−240.

(15) Leonard, J. E. Isomer Numbers of Nonrigid Molecules, The Cyclohexane Case. *J. Phys. Chem.* **1977**, *81* (23), 2212.

(16) Consistency checks for Pólya cycle indices: Replacing all Pólya variables f_n^p in a Pólya cycle index by 1 (assuming no substituents) should yield a sum equal to 1. For each product $\prod f_{n_i}^{p_i}$ in a Pólya cycle index, the sum $\sum n_i p_i$ should render the number of substituent sites. Consistency checks for isomer enumerations: The number of diamutamers should be the number of diastereoisomers plus the number of enantiomeric pairs. The number of diastereoisomers should be the number of enantiomeric pairs plus the number of achiral diamutamers.

(17) Dolhaine, H.; Hönig, H.; van Almsick, M. Sample Applications of an Algorithm for the Calculation of the Number of Isomers With More Than One Type of Achiral Substituent, MATCH communication in mathematical and in computer chemistry, March 1999, Vol. 39, pp 21−37, http://www.mathe2.uni-bayreuth.de/match.

(18) See e.g.: Balasubramanian, K. Applications of Combinatorics and Graph Theory to Spectroscopy and Quantum Chemistry. *Chem. Rev.* **1985**, *85*, 599−618.

(19) http://www.combinatorial.com/.

(20) http://www-orgc.tu-graz.ac.at/institut/softnew.htm.

(21) Fujita, S. Unit Subduced Cycle indices with and without Chirality Fittingness for Ih Group. An Application to Systematic Enumeration of Dodecahedrane Derivatives. *Bull. Chem. Soc. Jpn.* **1990**, *21*, 141−157.

(22) All calculations and benchmarks were performed on a 300 MHz PC x86 Pentium processor under Windows NT.

CI990121M