JCIM JOURNAL OF CHEMICAL INFORMATION AND MODELING

Application Note

Cite This: J. Chem. Inf. Model. 2019, 59, 2516−2521

pubs.acs.org/jcim

CGRtools: Python Library for Molecule, Reaction, and Condensed Graph of Reaction Processing

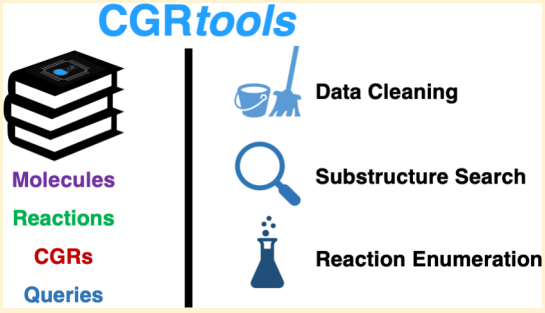
Ramil I. Nugmanov,[†] Ravil N. Mukhametgaleev,[†] Tagir Akhmetshin,[†] Timur R. Gimadiev,[†] Valentina A. Afonina,[†] Timur I. Madzhidov,*[,†] and Alexandre Varnek*[,#]

[†]Laboratory of Chemoinformatics and Molecular Modeling, A.M. Butlerov Institute of Chemistry, Kazan Federal University, 18 Kremlyovskaya Str., 420008 Kazan, Russia

[#]Laboratory of Chemoinformatics, Université de Strasbourg, 4 rue Blaise Pascal, 67000 Strasbourg, France

Supporting Information

ABSTRACT: CGRtools is an open-source Python library aimed to handle molecular and reaction information. It is the sole library developed so far which can process condensed graph of reaction (CGR) handling. CGR provides the possibility for advanced operations with reaction information and could be used for reaction descriptor calculation, structure−reactivity modeling, atom-to-atom mapping comparison and correction, reaction center extraction, reaction balancing, and some other related tasks. Unlike other popular libraries, CGRtools is fully written in Python with minor dependencies on other libraries and cross-platform. Reaction, molecule, and CGR objects in CGRtools support native Python methods and are comparable with the help of operations "equal to", "less than", and "bigger than". CGRtools supports common structural formats. CGRtools is distributed via an L-GPL license and available on https://github.com/cimm-kzn/CGRtools.



1. INTRODUCTION

Chemical reactions are important objects in chemoinformatics which have been attracting the attention of researchers in recent years.[1−3] Many different tasks emerging in chemical synthesis were elucidated using chemoinformatics technologies.[2] Computer representation of reaction is an important part of the modeling workflow, in large extent determining the modeling performance. Among many various methodologies of reaction encoding, the condensed graph of reaction (CGR)[4] approach recently demonstrated its efficiency in structure−reactivity modeling,[5−10] reaction condition prediction,[11,12] metabolic reaction product ranking,[13] atom-to-atom mapping error identification,[14] and substructure[11] and similarity search in reaction.[4] CGR is a molecular graph resulting from superposition of reactant and product molecules, in which atoms and bonds which changed their properties are specially labeled (Figure 1).

Handling chemical structures is an essential step in any chemoinformatics tools. This step could include duplicate searching and filtering, standardization of chemical structures, valence error identification and filtering, and descriptor calculation. There are several universal tools that were developed for handling molecular information, both open-source and commercial, such as OpenEye Toolkits,[15] ChemAxon JChem,[16] Indigo Toolkit,[17] RDKit,[18] CDK,[19] Open Babel,[20] and CACTVS.[21]

Nowadays, Python is one of the most used language in chemoinformatics and data mining (see Figure S1 of the Supporting Information) whose popularity is a direct result of its versatility, interoperability, simplicity to study, possession of different libraries for different tasks, and cross-platform nature.

Existing tools provide relatively weak support for reactions (see Table S1 of the Supporting Information). Only some simple operations for reactions are possible: extraction of reactants and products and some simple operations. To our knowledge, no public tool supporting CGR has been developed so far.

The rise in interest in chemical reactions in the last two−three years,[1,2] as well as growing incorporation of Python-based libraries, motivated us to develop the native Python based library CGRtools for handling chemical structures, which expands the object-oriented paradigm to molecules, reactions, and condensed graphs of reaction. Stereochemistry support is not implemented in the present version of CGRtools but will be added in the next release.

2. SOFTWARE OVERVIEW

CGRtools supports molecules and reaction as objects being the only tool supporting CGRs. CGRtools treats chemical objects similarly to standard Python data types like integers, strings, etc. Every chemical object is hashable due to atom numbering canonicalization (see the algorithm description in Appendix A2 in the Supporting Information). The objects support trans-

Received: February 1, 2019
Published: May 7, 2019

DOI: 10.1021/acs.jcim.9b00102
J. Chem. Inf. Model. 2019, 59, 2516−2521

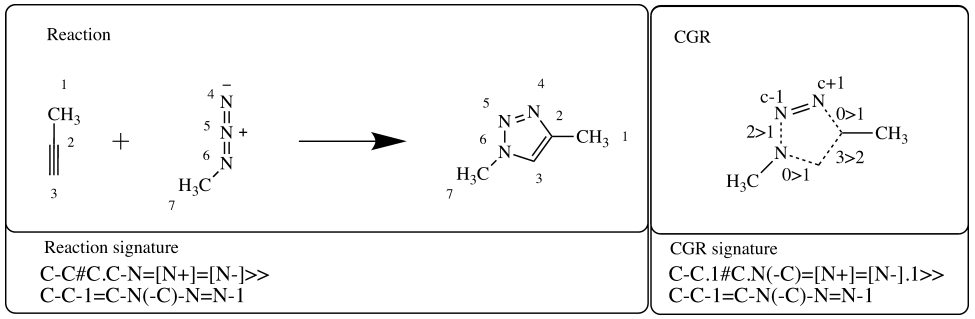


Figure 1. Chemical reaction (left) and its CGR (right). Dotted lines represent dynamic bonds. Indices "c−1" and "c+1" mean that formal atomic charges are lowered and increased by 1, respectively. Indices "0>1", "2>1", "3>2" denote formed single bond, double bond transformation to single and triple to double, correspondingly. On the bottom, signatures of the reaction and CGR are given.

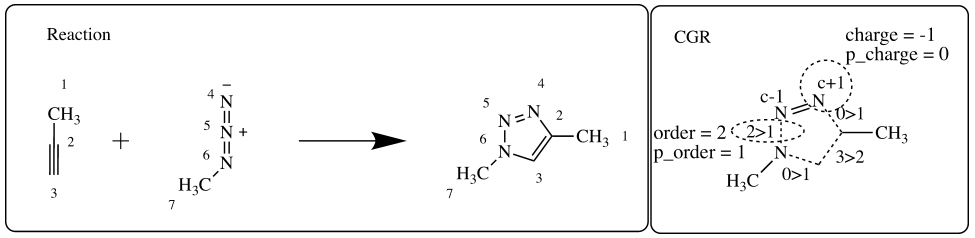


Figure 2. Atomic and bond labels in reagent and product part of CGR. For circled dynamic atoms and bonds, reagent and product properties are given.

parent class inheritance which augments existing functionalities—methods and attributes—without breaking up existing ones.

CGRtools has some dependencies: *networkx*[22] (for graph manipulation), *lxml*[23] (for XML file parsing), and optionally *coho*[24] (for SMILES parsing). Original DLLs for InChI read/write applicable to Windows and Linux are included in distribution. CGRtools efficiently process reaction data; see speed tests of some operations in Table S2, Supporting Information.

3. SOFTWARE APPLICATION AND FEATURES

Hereafter some basic features of CGRtools will be described. For the readers' convenience, the CGRtools tutorial and comparison of CGRtools with other tools (Table S1) are given in the Supporting Information.

3.1. Objects and Basic Operations with Them.

CGRtools supports molecules, reactions, and CGRs as objects. Every CGRtools' object consists of two parts: (i) a structural part, describing the structure of the chemical object as a graph using the NetworkX library,[22] (ii) associated information, accessible via the *meta* attribute. The latter is {key: value} storage for additional data given as a number or text.

Molecules are internally stored as *MoleculeContainer* (see part 1.1 of the tutorial, Supporting Information). Nodes and edges of graph represent separate class *Atom* and *Bond*, correspondingly. *Atom* objects store attributes of atoms (element symbol, charge, etc.). *Bond* objects store information about bond order. *MoleculeContainer* also has a dictionary of associated molecular properties (attribute *meta*). Atomic and bond attributes can be changed. A hash uniquely specifying atom or bond type can be generated.

Hybridization, number of explicit, implicit, or any hydrogen atoms as well as number of adjacent atoms could be calculated for every atom. These attributes of *Atom* objects are designed as additional restrictions in substructure search and not substitutable.

Widespread graph algorithms are implemented to handle molecules. Atoms, belonging to rings, can be identified using the smallest set of smallest ring (SSSR) algorithm.[25] Molecule in CGRtools could be represented by disconnected graphs. It is useful for salts representation as unity of anion(s) and cation(s). The lists of atoms belonging to connected components in the graph are accessible by the `connected_components` attribute and could be split into molecules containing only one component using the `split()` method.

Reactions in CGRtools are kept as *ReactionContainer* objects (see part 1.2 of the CGRtools tutorial, Supporting Information). It contains three lists of molecules: reactants, products, and reagents. Molecules are accessible as *MoleculeContainers*.

The *CGRContainer* object is similar to *MoleculeContainer* with some exceptions (see part 1.3 of the tutorial, Supporting Information). First, valence check and operations with implicit/explicit hydrogens are not supported since valence rules are not fulfilled for CGR. Standardization methods (see below) are also not applicable. On the other hand, one can access atoms and bonds considering that some properties could be dynamic, Figure 1. Element types and isotopes could not be changed in the reaction and thus are available using `element` and `isotope` attributes. But since formal charge, multiplicity, number of neighbors, and hybridization could be changed in the reaction, two series of attributes are used: `charge`, `multiplicity`, `neighbors`, and `hybridization` for the reactant part and `p_charge`, `p_multiplicity`, `p_neighbors`, and `p_hybridization` for atomic labels from the product part (Figure 2). Bonds also have related attributes `order` (bond order prior to reaction) and `p_order` (bond order after transformation).

DOI: 10.1021/acs.jcim.9b00102
J. Chem. Inf. Model. 2019, 59, 2516−2521

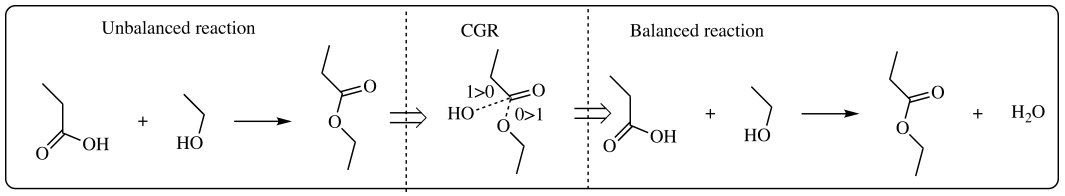


Figure 3. Reaction balancing using reaction transformation in CGR.

ReactionContainer could be transformed into CGR using the `compose()` method, opposite transformation could be done using the `decompose()` method. Note that, if a reaction is not balanced, its transformation to CGR and back will result in a new reaction that is balanced and does not coincide with the initial reaction, Figure 3. This feature of CGR is used for reaction balancing.[26] Since for some cases balancing could be imperfect, a special rule-based approach have been developed.[26]

CGRtools has extended scripting possibilities for molecule, reaction, and CGR construction from scratch (part 1.5 of the CGRtools tutorial, Supporting Information). It supports atom and bond label substitution and atom and bond creation/deletion. Molecules could be merged to reaction. There are special objects *QueryContainer* and *QueryCGRContainer* which build queries for substructure search. *QueryContainer* and *QueryCGRContainer* are very flexible objects with excluded integrity checks. Such labels like hybridization could be optionally created, deleted, or modified; lists of possible atom properties could be specified. *QueryContainer* could represent a molecule or reaction, and *QueryCGRContainer* represents a CGR.

CGRtools objects support simple class inheritance (part 1.6 of the CGRtools tutorial, Supporting Information). It is designed for upgrading current functionality.

3.2. Canonicalization of Objects and Signature Calculation. CGRtools has an embedded algorithm for molecular structure canonicalization (described in appendix A2 in the Supporting Information), which implies the Morgan idea of extended connectivity.[27] Instead of summation, CGRtools implements the Gasteiger−Ihlenfeldt algorithm on prime number multiplication,[28] which is free of the possible error that extended connectivity values could occasionally coincide, since there is only one possibility to factorize a number. The implemented canonicalization algorithm can be used for generation of unique signatures of chemical objects: molecules, reactions, and CGRs, that look like unique SMILES strings (see part 2 of the tutorial, Supporting Information). The procedure of signature generation is described in appendix A2 of the Supporting Information. Notice that signatures of chemical objects are sensitive to atom and bond types, i.e. aromaticity, tautomeric state, and molecule standardization.

MoleculeContainer, *ReactionContainer*, and *CGRContainer* objects include signatures as additional internal representation. They could be printed out using a `str(object)` command. If one compares CGRtools objects using "==" (equality operator), it returns `True/False` based on correspondence of the object signatures (see part 2 of the CGRtools tutorial, Supporting Information). The expression `molecule1==molecule2` returns `True` only if two molecules are isomorphic. Moreover, the signatures are used when hashes of CGRtools objects are calculated. Thus, *MoleculeContainer*, *ReactionContainer*, and *CGRContainer* are hashable objects. They could be used directly as keys in Python *dictionary* and elements of *set* data type. The latter operation is a simple way to delete duplicated objects.

3.3. Standardization. CGRtools includes several simple procedures for molecule standardization (see part 3 of the tutorial in the Supporting Information). *MoleculeContainer* and *ReactionContainer* (but not *CGRContainer*) have `standardize()` and `aromatize()` methods. The former performs the transformation of the most widespread groups into standard form. Aromatization is performed using embedded aromatization rules. Quinones are transformed into dearomatized form using the same rule set. Fused rings and N-, O-, P-heterocycles are supported well, but 7- and more membered rings are not supported in the present version. The `explicify_hydrogens()` and `implicify_hydrogens()` options add or delete explicit hydrogens. For reactions, standardization is applied independently to individual molecules in them.

For every atom, violation of valence rules could be checked by the `check_valence()` method. The valence rules for Periodic Table groups 1, 2, 13, 14, 15, 16, 17, and 18 were taken from Marvin Applets (version 14.7.21.0).[29]

3.4. Substructure Isomorphism. CGRtools includes possibilities for substructure search in molecules and reactions (see part 4 of the tutorial, Supporting Information). It uses the state-of-the-art VF2[30] algorithm. Full enumeration of possible isomorphic embeddings is possible.

CGRtools applies a simple and intuitive API for substructure isomorphism. Command *query* < *molecule* returns True only if query is substructure of molecule. Additional labels on atoms such as hybridization, number of neighbors are considered. To assign them the `reset_query_marks()` method is implemented. If a query contains labels of hybridization and number of atom neighbors, then substructure must have the same labels. Special CGRtools objects *QueryContainer* and *QueryCGRContainer* were developed to represent query in isomorphic embedding which gives freedom in the query specification.

Reactions could not be directly used for substructure search: its components (reactants, products and reagents) must be enumerated. If one wants to find a reaction involving reactant with some substructure, all reactants need to be returned followed by substructure search in them.

At the same time, CGR can be used for substructural search, Figure 4. It gives also an opportunity to perform subgraph isomorphism considering atom-to-atom mapping (AAM). Due to application of CGR technology, the AAM-aware substructural search for reaction incorporates the same embedding algorithm as that for the molecules.

3.5. Transformation Rule Extraction. CGRtools can be used for extraction and application of transformation rules (see part 5 of the tutorial, Supporting Information), which could be further used in synthesis design projects (i.e., retrosynthesis and related tasks).

DOI: 10.1021/acs.jcim.9b00102
J. Chem. Inf. Model. 2019, 59, 2516−2521

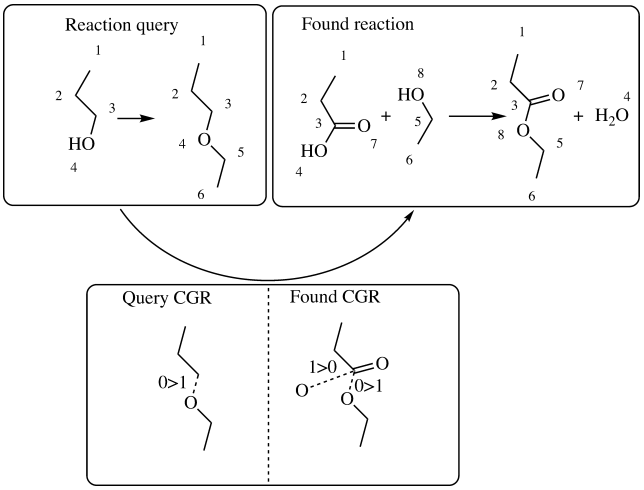


Figure 4. Example of substructure search in reactions using CGR.

A *CGRContainer* has attributes that return list of atoms (`center_atoms`) or bonds (`center_bonds`) belonging to reaction center. Atoms, which change formal charge or multiplicity, will appear as dynamic atoms in the list of reaction centers. The attribute `centers_list`, which returns the list of distant reaction centers, considers atoms linked by dynamic bonds as one reaction center, Figure 5. Obtained lists of atom numbers can be used to extract transformation rules including the environment of reaction centers.

The number of dynamic bonds in CGR called chemical distance[31] can be used for assessment of AAM quality. According to minimal chemical distance principle,[31] the larger the number of dynamic bonds, the higher the probability that AAM is incorrect. We used chemical distance for selecting the best AAM out of several possibilities proposed by different programs, that was called consensus mapping approach.[32]

3.6. Reactor. *Reactor* generates molecules applying transformation rules to some template molecule (see part 6.1 in the tutorial, Supporting Information). Transformation rule is *ReactionContainer* which in reagent side contains query for substructural matching and in product side patch for updating matched atoms and bonds with given atom-to-atom mapping. *Reactor* is a callable object that is initialized with transformation rule and then applied to a molecule or mixture of reactants. Results of transformation rule application could be united into reaction with known AAM. *Reactor* supports enumeration of a specified number of products for a given molecule (if possible) as well as generation of all possible products.

A unique feature of *Reactor* is an opportunity to apply the transformation rules to CGR. This option is particularly useful for fixing widespread AAM errors. In part 6.2 of the tutorial (Supporting Information), we demonstrated how wrong AAM could be fixed. This approach also provides the possibility to transform reaction of one type into another (for example, to generate concurrent reactions, see part 6.2 of the tutorial, Supporting Information).

In such a way, CGRtools can both extract the transformation rules, apply them to arbitrary molecule(s), and fix automatically the AAM errors.

3.7. Supported File Formats. CGRtools support reading chemical information from many common file formats for reactions and molecules: MOL/SDF/RXN/RDF (v2000 and v3000), SMILES (using *coho* library), ChemAxon MRV format, and InChI string. MDL and MRV are supported in basic format without S-groups and polymer processing. Data parsers transparently support any data sources, which can be a file or a network page on HTTP/FTP resource. Data in string format can be read from stream using StringIO wrapper. Data in gz-files or zip archives can be decompressed on the fly. Parsers work in streams, which saves memory.

CGRtools supports export of its objects into MDL SDF, RDF v2000, and MRV formats. Writer also supports compression and network transparency.

CGR could be read and saved in SDF format adapted in our laboratory (see description in appendix A3 of the Supporting Information). Produced SDF file could be used for fragment descriptors calculation using ISIDA Fragmentor software.[33]

Examples of file read/write operations is given in part 7 of the tutorial in the Supporting Information.

4. CONCLUSIONS

CGRtools is a powerful open source library for chemoinformatics of general utility with special attention paid to chemical reaction handling. Being written fully in Python and incorporating its object-oriented features, CGRtools is Python-friendly and transparent and supports class inheritance, that makes working with chemical objects as simple as with numbers.

CGRtools is available via an L-GPL license on https://github.com/cimm-kzn/CGRtools. Documentation is available on https://cgrtools.readthedocs.io/.

■ ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge on the ACS Publications website at DOI: 10.1021/acs.jcim.9b00102.

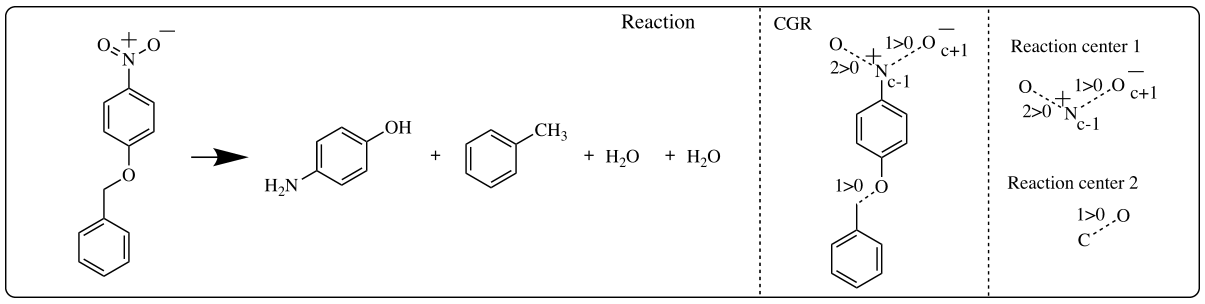


Figure 5. Example of reaction centers extracted by CGRtools.

DOI: 10.1021/acs.jcim.9b00102
J. Chem. Inf. Model. 2019, 59, 2516−2521

Comparison of the popularity of different programming languages in chemoinformatics, some examples of CGRtools application, description of developed canonicalization algorithm, the CGR file format description, and the CGRtools tutorial (PDF)

AUTHOR INFORMATION

Corresponding Authors

*Email: Timur.Madzhidov@kpfu.ru.
*Email: varnek@unistra.fr.

ORCID

Ramil I. Nugmanov: 0000-0002-8541-9681
Ravil N. Mukhametgaleev: 0000-0002-5619-3468
Tagir Akhmetshin: 0000-0002-2549-6431
Timur R. Gimadiev: 0000-0001-5012-0308
Valentina A. Afonina: 0000-0002-5291-8636
Timur I. Madzhidov: 0000-0002-3834-6985
Alexandre Varnek: 0000-0003-1886-925X

Notes

The authors declare no competing financial interest.

ACKNOWLEDGMENTS

We thank Dr. D. Horvath, Dr. G. Marcou, and Dr. F. Bonachera (University of Strasbourg) for valuable advice and fruitful discussions. Development of CGRtools was initiated within a project funded by the Russian Science Foundation (grant no. 14-43-00024). The work was funded from the Russian Government Program of Competitive Growth of Kazan Federal University, the program of State Assignments for Science (project nos. 4.1493.2017/4.6 and 4.5151.2017/6.7) funded by the Ministry of Science and Higher Education of the Russian Federation.

ABBREVIATIONS

CGR,condensed graph of reaction; AAM,atom-to-atom mapping; API,application programming interface

REFERENCES

(1) Engkvist, O.; Norrby, P.-O.; Selmi, N.; Lam, Y.; Peng, Z.; Sherer, E. C.; Amberg, W.; Erhard, T.; Smyth, L. A. Computational Prediction of Chemical Reactions: Current Status and Outlook. *Drug Discovery Today* **2018**, *23*, 1203−1218.

(2) Baskin, I. I.; Madzhidov, T. I.; Antipin, I. S.; Varnek, A. A. Artificial Intelligence in Synthetic Chemistry: Achievements and Prospects. *Russ. Chem. Rev.* **2017**, *86*, 1127−1156.

(3) Coley, C. W.; Green, W. H.; Jensen, K. F. Machine Learning in Computer-Aided Synthesis Planning. *Acc. Chem. Res.* **2018**, *51*, 1281.

(4) Varnek, A.; Fourches, D.; Hoonakker, F.; Solov'ev, V. P. Substructural Fragments: An Universal Language to Encode Reactions, Molecular and Supramolecular Structures. *J. Comput.-Aided Mol. Des.* **2005**, *19*, 693−703.

(5) Madzhidov, T. I.; Gimadiev, T. R.; Malakhova, D. A.; Nugmanov, R. I.; Baskin, I. I.; Antipin, I. S.; Varnek, A. A. Structure−reactivity Relationship in Diels−Alder Reactions Obtained Using the Condensed Reaction Graph Approach. *J. Struct. Chem.* **2017**, *58*, 650−656.

(6) Nugmanov, R. I.; Madzhidov, T. I.; Khaliullina, G. R.; Baskin, I. I.; Antipin, I. S.; Varnek, A. A. Development of "Structure-Property" Models in Nucleophilic Substitution Reactions Involving Azides. *J. Struct. Chem.* **2014**, *55*, 1026−1032.

(7) Polishchuk, P.; Madzhidov, T.; Gimadiev, T.; Bodrov, A.; Nugmanov, R.; Varnek, A. Structure−reactivity Modeling Using Mixture-Based Representation of Chemical Reactions. *J. Comput.-Aided Mol. Des.* **2017**, *31*, 829−839.

(8) Madzhidov, T. I.; Bodrov, A. V.; Gimadiev, T. R.; Nugmanov, R. I.; Antipin, I. S.; Varnek, A. A. Structure−reactivity Relationship in Bimolecular Elimination Reactions Based on the Condensed Graph of a Reaction. *J. Struct. Chem.* **2015**, *56*, 1227−1234.

(9) Glavatskikh, M.; Madzhidov, T.; Horvath, D.; Nugmanov, R.; Gimadiev, T.; Malakhova, D.; Marcou, G.; Varnek, A. Predictive Models for Kinetic Parameters of Cycloaddition Reactions. *Mol. Inf.* **2019**, *38*, 1800077.

(10) Gimadiev, T. R.; Madzhidov, T. I.; Nugmanov, R. I.; Baskin, I. I.; Antipin, I. S.; Varnek, A. Assessment of Tautomer Distribution Using the Condensed Reaction Graph Approach. *J. Comput.-Aided Mol. Des.* **2018**, *32*, 401−414.

(11) Lin, A. I.; Madzhidov, T. I.; Klimchuk, O.; Nugmanov, R. I.; Antipin, I. S.; Varnek, A. Automatized Assessment of Protective Group Reactivity: A Step Toward Big Reaction Data Analysis. *J. Chem. Inf. Model.* **2016**, *56*, 2140−2148.

(12) Marcou, G.; Aires de Sousa, J.; Latino, D. A. R. S.; de Luca, A.; Horvath, D.; Rietsch, V.; Varnek, A. Expert System for Predicting Reaction Conditions: The Michael Reaction Case. *J. Chem. Inf. Model.* **2015**, *55*, 239−250.

(13) Madzhidov, T. I.; Khakimova, A. A.; Nugmanov, R. I.; Muller, C.; Marcou, G.; Varnek, A. Prediction of Aromatic Hydroxylation Sites for Human CYP1A2 Substrates Using Condensed Graph of Reactions. *Bionanoscience* **2018**, *8*, 384−389.

(14) Muller, C.; Marcou, G.; Horvath, D.; Aires-de-Sousa, J.; Varnek, A. Models for Identification of Erroneous Atom-to-Atom Mapping of Reactions Performed by Automated Algorithms. *J. Chem. Inf. Model.* **2012**, *52*, 3116−3122.

(15) *OpenEye Toolkits*; OpenEye Scientific Software, Santa Fe, NM, 2018; http://www.eyesopen.com.

(16) *JChem Calculator Plugins 19.2.0*, ChemAxon, 2019.

(17) *Indigo Toolkit*; EPAM Systems, Inc., 2014; http://lifescience.opensource.epam.com/indigo/index.html.

(18) *RDKit: Open-Source Cheminformatics*; 2018; http://www.rdkit.org.

(19) Willighagen, E. L.; Mayfield, J. W.; Alvarsson, J.; Berg, A.; Carlsson, L.; Jeliazkova, N.; Kuhn, S.; Pluskal, T.; Rojas-Chertó, M.; Spjuth, O.; Torrance, G.; Evelo, C. T.; Guha, R.; Steinbeck, C. The Chemistry Development Kit (CDK) v2.0: Atom Typing, Depiction, Molecular Formulas, and Substructure Searching. *J. Cheminf.* **2017**, *9*, 33.

(20) O'Boyle, N. M.; Banck, M.; James, C. A.; Morley, C.; Vandermeersch, T.; Hutchison, G. R. Open Babel: An Open Chemical Toolbox. *J. Cheminf.* **2011**, *3*, 33.

(21) *CACTVS*; Xemistry GmbH, 2019.

(22) Hagberg, A. A.; Schult, D. A.; Swart, P. J. Exploring Network Structure, Dynamics, and Function Using NetworkX. In *Proceedings of the 7th Python in Science Conference*; Varoquaux, G., Vaught, T., Millman, J., Eds.; Pasadena, CA, USA, 2008; pp 11−15.

(23) Richter, S. lxml - XML and HTML with Python. https://lxml.de/index.html (accessed Mar 22, 2019).

(24) Cornett, B. Coho: SMILES parser. https://github.com/cornett/coho (accessed Mar 22, 2019).

(25) Lee, C. J.; Kang, Y.-M.; Cho, K.-H.; No, K. T. A Robust Method for Searching the Smallest Set of Smallest Rings with a Path-Included Distance Matrix. *Proc. Natl. Acad. Sci. U. S. A.* **2009**, *106*, 17355−17358.

(26) Nugmanov, R. I.; Madzhidov, T. I.; Antipin, I. S.; Varnek, A. A. An Approach for Automated Detection of Missing Reagents and Products in Chemical Reaction Equations. *Uchenye Zap. Kazan. Univ. Seriya Estestv. Nauk.* **2018**, *160*, 32−39.

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

(28) Ihlenfeldt, W. D.; Gasteiger, J. Hash Codes for the Identification and Classification of Molecular Structure Elements. *J. Comput. Chem.* **1994**, *15*, 793−813.

DOI: 10.1021/acs.jcim.9b00102
J. Chem. Inf. Model. 2019, 59, 2516−2521

(29) Marvin Applets, version 14.7.21.0. http://onlinelibrarystatic.wiley.com/marvin/ (accessed Jan 25, 2019).
(30) Cordella, L. P.; Foggia, P.; Sansone, C.; Vento, M. A (Sub)Graph Isomorphism Algorithm for Matching Large Graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* **2004**, *26*, 1367−1372.
(31) Jochum, C.; Gasteiger, J.; Ugi, I. The Principle of Minimum Chemical Distance (PMCD). *Angew. Chem., Int. Ed. Engl.* **1980**, *19*, 495−505.
(32) Madzhidov, T. I.; Nugmanov, R. I.; Gimadiev, T. R.; Lin, A. I.; Antipin, I. S.; Varnek, A. Consensus Approach to Atom-to-Atom Mapping in Chemical Reactions. *Butlerov Commun.* **2015**, *44*, 170−176.
(33) Varnek, A.; Fourches, D.; Horvath, D.; Klimchuk, O.; Gaudin, C.; Vayer, P.; Solov'ev, V.; Hoonakker, F.; Tetko, I. V.; Marcou, G. ISIDA - Platform for Virtual Screening Based on Fragment and Pharmacophoric Descriptors. *Curr. Comput.-Aided Drug Des.* **2008**, *4*, 191−198.

DOI: 10.1021/acs.jcim.9b00102
J. Chem. Inf. Model. 2019, 59, 2516−2521