Abstract—Many areas of chemical biology, such as drug discovery, rely heavily on chemical libraries offering compounds usable in the industrial processes. However, the "universe" containing all possible compounds, the so-called chemical space, is vast, and therefore, the libraries store only its representative parts. Thus, to explore the whole chemical space and to identify all its promising parts containing e.g., drug-like molecules computational methods have to be developed and employed.

In this paper, we propose a method for traveling in the chemical space called Molpher. Given two molecules, Molpher is intended to find a sequence of related compounds, called path in the chemical space, leading from the starting molecule to the target one. The path is generated by iterative application of the so-called morphing operators corresponding to simple chemical operations such as adding or removing an atom or a bond. The molecules on the resulting path represent a focused library that can be used as a starting point for other experiments. We also propose a testbed for examining qualities of algorithms such as Molpher. The testbed is used to describe Molpher’s qualities in terms of ability of finding a path in the space in the given time.

Keywords—chemical space; drug discovery; molecule morphing

I. INTRODUCTION

Modern medicine critically depends on small molecule drugs that interfere with specific biochemical targets. These drugs are generally organic molecules containing up to 25 atoms (not counting hydrogens), mostly carbon C with nitrogen N, oxygen O, sulphur S, phosphorus P, and other elements. Researchers now routinely screen millions of compounds in the search for some that are biologically active, i.e. they show beneficial or adverse effects on living matter. The number of possible organic compounds is huge and limited only by chemist’s imagination. These compounds form the so-called chemical space that can be viewed, according Lipinski and Hopkins [1], as a parallel to the universe with chemical compounds taking up a place instead of stars. Different estimates have been proposed for the size of chemical space varying between between $10^{18}$ and $10^{200}$ [2]. However, even the compound libraries of the largest pharmaceutical companies offer only a fraction of the chemical space. In addition, not all compounds from the chemical space are biologically active, and not all biologically active molecules have the desired physico-chemical properties to be a drug (e.g. they can be too greasy to be orally absorbed, they can be toxic, or they can be easily degraded by metabolizing systems). Compounds that are supposed to yield safe (non-toxic) orally bioavailable medicines are termed as "drug-like". But even biologically active compounds not having properties required of a drug can still be very useful as "tools" for analysing biological mechanisms and testing hypotheses in model systems. The branch of biology trying to discover such "tools" for every existing protein is known as the chemical biology or the chemical genetics/genomics [3].

Apparently, the key question for drug discoverer or chemical biologist is to identify the regions of the chemical space that are likely to contain biologically active compounds, i.e. biologically relevant chemical space. These regions can serve as a storage for discovering molecules that have never been reported previously. A major unknown is if these areas are sparsely distributed (and thus difficult to find), or if the most of the chemical space is empty. The key role in the chemical space investigation play computational methods allowing to mine chemical space for molecules with desired properties. However, automatic generation of chemical space is limited due to the combinatorial explosion - while the GDB-11 database contains 26.4 millions molecules of up to 11 atoms of C, N, O, and F [4], the GDB-13 contains already 970 million small organic compounds with up to 13 atoms of C, N, O, A and Cl [5]. To enumerate all possible molecules up to 25 atoms (or even larger, if the aim is not to identify a potential drug, but rather to find a useful molecular "tool" for which the restrictions are not so stringent) is clearly unfeasible, and thus alternative methods must be pursued.

In the present work, we suggest a method of "molecular morphing", that generates various molecules lying on the path between a start and target structure. In this way we are able to generate a focused library of compounds covering small yet relevant subset of chemical space. We suggest, that compounds encountered along the way provide valuable starting points for other experiments (either in silico, or in vitro/in vivo) assessing their biological activities. In addition, our method can be used as a useful tool for the study of the chemical space as such, its characteristics and properties,
including the investigation of the shapes and positions of regions of biologically active compounds.

Similar method called chemical space travel as implemented in the chemical spaceship program was developed in the research group of prof. Reymond [6]. However, the inner workings of this approach were not well described, and because the chemical spaceship program is not publicly available, the details of the algorithm remain unclear. In addition no information about time required for finding the solution was published, and also the space requirements in relation to the given hardware configuration are not known. The present work describes not only an alternative approach to the chemical space investigation, but we also propose a benchmark sets that could be useful in further chemical space exploration studies.

II. MOLPHER

Molpher’s goal is to provide a path \( P \) in the chemical space beginning in a source molecule \( M_S \) and ending in a target molecule \( M_T \). The path consists of molecules \( \mathcal{P} = \{ M_S = M_1, ..., M_n = M_T \} \) where \( M_{i+1} \) is created from \( M_i \) by the application of a morphing operator (sec. II-B). The details of the algorithm describes section II-A. Here, we briefly describe the basic idea together with the description of some of the restrictions.

The main idea of Molpher presents Fig. 1. The algorithm is based on an iterative process where at each step \( i \) a set of molecules \( M_i \) is used as a prescription for the set \( M_{i+1} \). To each of the molecules from \( M_i \) random morphing operators are applied several times to generate candidate morphings. The morphs are merged and sorted into a list with respect to the distance to \( P_T \) (sec II-C). A given number of candidates are accepted for \( M_{i+1} \). The process starts with \( M_0 = \{ M_S \} \) and terminates when \( M_T \in M_i+1 \). The sequence of morphs starting in \( M_S \) and ending in \( M_T \) defines the resulting path \( \mathcal{P} \).

A. Molpher’s Algorithm

The baseline of the algorithm demonstrates in pseudocode Alg. 1. The code formalizes the description of Molpher from the previous section and therefore here we will only describe some of the details hidden in the code.

The code of Alg. 1 mentions three functions which should be described in a bit more detail. One of the most important functions in the code is GenerateMorphings. Inside the function, the morphs of a given molecule are generated. The number of morphs generated for each molecule is dictated by the \( \text{cnt\_morphs} \) and \( \text{cnt\_morphs\_det} \) parameters. The former one defines number of morphs if the input molecule is in the distance at least \( \text{dist\_det} \) from \( M_T \). The latter one represents the number of morphs if the considered molecule is closer to the target than \( \text{dist\_det} \). \( \text{cnt\_morphs\_det} \) should by set noticeably higher than \( \text{cnt\_morphs} \). In that way we are able to quickly guide the

---

**Algorithm 1** Molpher’s algorithm (Computes path \( P \) between source molecule \( M_S \) and target molecule \( M_T \))

\[
\mathcal{M}_0 \leftarrow \{ M_S \} \\
P \leftarrow \emptyset \\
\text{for } i = 1 \text{ to } \text{params.cnt\_max\_iterations} \text{ do} \\
\quad M_i \leftarrow \emptyset \\
\quad \text{for all } M \in \mathcal{M}_{i-1} \text{ do} \\
\quad\quad /* Generate only \text{params.cnt\_accept} \text{ morphs from } M \text{ being closest to } M_T. */ \\
\quad\quad \text{accept} \quad \text{params.cnt\_accept}\text{\_max} \quad \text{morphs from } M \text{ depending on the distance of } M \text{ to } M_T \text{ and the value of } \text{params.dist\_det}. */ \\
\quad\quad M_{aux} \leftarrow \text{GenerateMorphings}(M, \\
\quad\quad \text{params.cnt\_morphs,} \\
\quad\quad \text{params.cnt\_morphs\_det,} \\
\quad\quad \text{params.dist\_det}) \\
\quad\quad \text{for all } M_{aux} \in M_{aux} \text{ do} \\
\quad\quad\quad M_{aux}\_pred \leftarrow M \\
\quad\quad \text{end for} \\
\quad\quad \mathcal{M}_i \leftarrow \mathcal{M}_i \cup M_{aux} \\
\quad\quad \text{end for} \\
\quad \text{if } \exists M \in M_i : \text{IsIdentical}(M, M_T) \text{ then} \\
\quad\quad /* Generate the path based on the } M.\text{pred information. */ \\
\quad\quad P = \text{GeneratePath}(M) \\
\quad\quad \text{break} \\
\quad \text{end if} \\
\quad \text{Sort}(M_i, \delta, M_T) \\
\quad /* Accept only \text{params.cnt\_accept} \text{ morphs from } M_i \text{ being most similar to } M_T. */ \\
\quad \text{The rest of the molecules will be accepted with a probability based on their position in the sorted list. */ \\
\quad \mathcal{M}_i \leftarrow \text{Accept}(M_i, \text{params.cnt\_accept}) \\
\quad /* Only \text{params.cnt\_accept\_max} \text{ morphs can be accepted. */ \\
\quad \mathcal{M}_i \leftarrow \text{Prefix}(M_i, \text{params.cnt\_accept\_max}) \\
\quad /* Remove those molecules } M \text{ (and all their successors) having no successor closer to } M_T \text{ than } M \text{ for } \text{params.cnt\_it\_prune} \text{ number of iterations. If } \text{params.cnt\_morphs\_max} \text{ directions have been explored for a given molecule, remove it from the exploration process. */ \\
\quad \text{Prune}(\text{params.cnt\_it\_prune}, \\
\quad \text{params.cnt\_morphs\_max, } \mathcal{M}_1, ..., M_i) \\
\quad \text{end for} \\
\quad \text{return } P 
\]
Each of the morphs is generated by application of one of the morphing operators to the input molecule. The selection of the operator type is random. The operator types are described in detail in sec. II-B.

The second important function which deserves attention is Accept. In step $i$, not all the morphs can be accepted for $M_{i+1}$ considering both time and space constraints. Therefore, the candidate set is ordered according to the distance to the target and only $\text{cnt}_{\text{accept}}$ nearest candidates are accepted automatically. The rest of the candidates are accepted with probability being proportional to their position in the ordered set. The function assessing distance to a pair of molecules is described in sec. II-C. We do not employ some more elaborate solutions such as simulated annealing [7] (SA) which might seem as a more suitable solution. That is the consequence of the domain in which the space exploration is applied. SA is suitable for finding suboptimal solutions which follows from the decreasing probability of acceptance of a candidate with decreasing temperature (increasing time). The decreasing temperature is not applicable in case of Molpher. Molpher does not recognize "suboptimal solution", the target is either found or not. Therefore the probability of accepting a candidate molecule should be constant during the whole process.

The third function worth detail description is Prune. The motivation for pruning is to avoid local optima in the space. Without pruning, one of the branches of the computation might find itself in a part of the space where all the molecules are close to the solution in terms of the distance function but there is not the possibility to actually reach the target without crossing large part of space with worse score than the actual position. If a substantial part of $M_i$ gets into such part of the space than probability of acceptance of other path (possibly leading to the target) becomes very low. The low probability is the consequence of restriction of the number of accepted morphs to $\text{cnt}_{\text{accept}}$.

Therefore, if a molecule $M$ does not (transitively) generate a morph being closer to the target than $M$ for $\text{cnt}_{\text{it}}$ iterations, the molecule and all its morphs are discarded. Such a solution helps to evade sticking of the process in a local optima. In order to avoid returning to the same area which had once been examined and rejected, a global list of examined directions is stored for each molecule and it is not allowed to go to those directions more than ones. We speak about global list since it is possible to get to one molecule (point) in the space in many different ways. To uniquely identify a molecule in the space independently on the order of its atoms or chemical groups we employ canonical SMILES [8]. To speed up the exploration and to avoid repetitive exploration of not promising areas of the space, we introduce second parameter into the pruning process $\text{cnt}_{\text{morphs}}$. This parameter serves as a global guard. If the number of morphs (counted globally) generated from a particular molecule exceeds this value, the molecule is removed from the exploration process and no other future path can cross this point. The $\text{cnt}_{\text{morphs}}$ can be understood as the estimate of the maximal possible number of morphs for each molecule. Since to compute such a number is difficult and dependent on the morphing operators and their implementation we decided to use a constant value.

The last detail of the code we cover here is the structure called $\text{params}$. $\text{params}$ significantly influences the qualitative characteristics of the Molpher method. Therefore, we will list the components of the structure here and describe their effects. In the brackets we show values of the parameters used in our experiments if not stated otherwise:

- $\text{cnt}_{\text{max}}$ iterations (100) The number of iterations forms a stop condition only so that the computation does not run forever in situation where Molpher fails to find a path.
- $\text{cnt}_{\text{morphs}}$ (20), $\text{cnt}_{\text{morphs}}$ det (100), dist det (0.2) The number of morphs to be generated represents a trade-off between time and quality. Low number of morphs generated from each molecule in each step might increase the time needed to find a solution. On the other hand, low number of morphs might increase the number of steps needed to find the solution or the solution might not be found at all in given number of iterations. Therefore we introduce the parameters $\text{cnt}_{\text{morphs}}$ det and dist det which allow to switch to more detailed space exploration ($\text{cnt}_{\text{morphs}}$ det) when a molecule gets close enough (dist det) to the target.
- $\text{cnt}_{\text{accept}}$ (30), $\text{cnt}_{\text{accept}}$ max (60) The number of morphs to be accepted for the next itera-
tion represents the trade-off between time and space constraints on one side and quality on the other. By storing larger set of candidates we increase the required memory for the method and also its runtime. The higher runtime results in larger candidate set and thus higher number of morphs to be generated in the next step. On the other hand, higher number of candidates allows us to browse larger area of the space and thus increases the probability of finding a solution.

- **cnt_it_prune(5)** The number of iterations after which a molecule and all its morphs is discarded influences the ability to find a solution. If the parameter is set too low a solution which needs to cross some larger suboptimal parts of the space would never be identified. On the other hand, if set too high the probability of being stuck in a local optima increases and so does the runtime.

- **cnt_morphs_max(300)** The global maximal number of morphs which can be generated from a molecule differs from cnt_morphs and cnt_morphs_det which represent number of morphs for given iteration. Low value of this parameter can cause missing a path which might lead to the target. On the other hand, increasing this value causes increase of the runtime.

### B. Morphing Operators

To generate a new molecule $M_{i+1}$ from an existing one $M_i$, Molpher employs a set of following morphing operators. The resulting molecule is a subject to the basic conformation rules, e.g. complying with the maximum bond order for each of the atom types.

- **Add atom** ($O_{AA}$). The $O_{AA}$ operator adds a random atom with a type present in $M_T$ to $M_i$ and connects it using the single bond to a random atom in $M_i$.
- **Remove atom** ($O_{RA}$). The $O_{RA}$ operator removes a random atom from $M_i$ so that $M_{i+1}$ does not decompose into two molecules (the graph of $M_{i+1}$ remains connected).
- **Add bond** ($O_{AB}$). The $O_{AB}$ operator adds a bond between a random pair of atoms of $M_i$. If the selected atoms are already connected, the bond order is increased.
- **Remove bond** ($O_{RB}$). The operator $O_{RB}$ decreases the bond order of a random bond found in $M_i$. If the chosen bond is single, the bond is deleted. Only such single bond can be selected whose remove does not cause decomposing of $M_{i+1}$ into two molecules (the graph of $M_{i+1}$ remains connected).
- **Mutate atom** ($O_{MA}$). The $O_{MA}$ operator substitutes an atom type of a random atom in $M_i$ for a random atom type from $M_T$.
- **Interject atom** ($O_{IA}$). The $O_{IA}$ operator inserts a random atom to $M_{i+1}$ between two random atoms being connected in $M_i$. Thus the bond connecting the select atoms is split into two.
- **Bond reroute** ($O_{BR}$). The $O_{BR}$ operator reconnects a random bond in $M_i$ by selecting a random end of the bond and rerouting it to a random atom (the graph of $M_{i+1}$ remains connected).
- **Bond contraction** ($O_{BC}$). The $O_{BC}$ operator removes a random bond connecting atoms with identical types. Moreover, the respective atoms are merged so that the number of atoms in $M_{i+1}$ is one less than the number of atoms in $M_i$.

All the operators restrict new atom types to those present in $M_T$. The behavior when only atom types from $M_T$ are considered suits the optimization purposes only and can be easily changed if paths crossing specific areas of the chemical space are interesting for given task.

### C. Pairwise Molecule Similarity

To guide the mutation process to reach the target molecule, a distance measure is required. The selected mea-
sure operates on chosen descriptors (characteristics) of the molecules being compared. Molpher utilizes a combination of two distance measures and underlying descriptors.

First, so called hashed fingerprints\(^1\) are utilized as molecular descriptors and Tanimoto coefficient [9], [10] is applied to compare a pair of descriptors (molecules). Molecular fingerprinting is a common method which basically produces a bitset that contains information on the molecular connectivity and its associated atomic elements. To generate a fingerprint, we consider path-distances for each vertex, beginning in 0, to a user-defined path-length (11 in Molpher). Then we annotate the atomic symbol (or other atomic property) of each vertex along all paths of each length. Each single path results in a string that is converted to a number which is, in turn, hashed into a value ranging from 0 to the length of the fingerprint. Finally the respective bit of the fingerprint is turned on.

Second, we extract a set of statistical characteristics describing the molecule such as the number of atoms of each atom type, the number of bonds of order one to four and the connection table (number of bonds in the molecule for each pair of atom types). These characteristics result in a vector of integers on top of which we define distance function \( \tau = 1 - T \) so that we can safely aggregate the two measures.

The two measures are aggregated to obtain the final distance \( \delta \) for molecules \( M_1, M_2 \):

\[
\delta(M_1, M_2) = \frac{\tau(F_H(M_1), F_S(M_2)) + \xi \times L_1(F_S(M_1), F_S(M_2))}{1 + \xi}
\]

(1)

The \( F_H \) and \( F_S \) functions represent the hashed fingerprints and statistics extraction procedures. Using the parameter \( \xi \) we are able to favor the \( L_1 \) distance over the Tanimoto coefficient distance. This might be useful in some cases where hashed fingerprints fail human intuitive of being similar. Imagine an example depicted in Fig. 3. Let us denote 3a as \( M_1 \), 3b as \( M_2 \) and 3c as \( M_T \) and let \( M_T \) be the desired target molecule. Although clearly \( M_2 \) looks more similar to \( M_T \) than \( M_1 \) in reality \( \tau(F_H(M_1), F_H(M_T)) < \tau(F_H(M_2), F_H(M_T)) \) (at least for all the implementations of hashed fingerprints we are aware of). We solved this issue by introducing the \( \delta \) aggregation function with suitable setting of the \( \xi \) parameter. For this specific example, \( \xi = 2 \) leads to the desired result i.e., \( \delta(M_1, M_T) > \delta(M_2, M_T) \).

\(^1\)We use the implementation in Open Babel denoted as FP2.

III. EXPERIMENTAL RESULTS

Molpher’s abilities were evaluated against a testbed introduced in sec. III-A. All experiments were carried out on a machine with 4 3GHz processors Intel(R) Xeon(R) E5450 (although only one computational unit was utilized for finding a single path) with 4 GB RAM and 64-bit OS Windows Server 2008 R2.

Currently, Molpher is limited to utilize compounds containing only carbon C, oxygen O and nitrogen N atoms (explicit hydrogens, though present in structures, are suppressed). In addition, these compounds are not allowed to be charged, and Molpher also can not treat stereoisomerism properly. This means, that stereoisomers — compounds having the same molecular formula and sequence of bonded atoms (constitution), but differing only in the three-dimensional orientations of their atoms in space — will not be present on the morphing path.

For manipulations with molecules Molpher utilizes the open-source "chemical toolbox" Open Babel (http://openbabel.org).

A. Compound Selection

To assess the quality of the Molpher algorithm, three sets of molecular structures differing by the similarity between the start and target structures were prepared. The compounds within the start/target pairs were selected from the publicly available database PubChem [11]. The selection was based on the representation of start and target structures by their PubChem substructure fingerprints (ftp://ftp.ncbi.nlm.nih.gov/pubchem/specifications/pubchem_fingerprints.txt), and the similarity was quantified as the Tanimoto score between these fingerprints. The first set, termed \( D_1 \), contains 20 start-target pairs with the similarity between start and target from the interval 0.7–0.8. The second set, \( D_2 \), contains pairs more dissimilar, the start/target similarities are from the interval 0.5–0.6. Finally, the third set, named \( D_3 \), contains pairs with start/target similarities between 0.3–0.4.

The structure of the test set is intended to examine Molpher’s qualities on different difficulty levels. We emphasize that the sets were chosen randomly and therefore the results might slightly differ on differently chosen sets. In the future, we plan to propose a bigger set. Here, we have to take into account the expected time complexity of the solution (see next section) which complicates carrying out more extensive experiments.
Table I: Path Finding Results

<table>
<thead>
<tr>
<th></th>
<th>$D_1$</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>$D_2$</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>$D_3$</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CID</td>
<td>CID</td>
<td>#it</td>
<td>path</td>
<td>time (m)</td>
<td>CID</td>
<td>CID</td>
<td>#it</td>
<td>path</td>
<td>time (m)</td>
<td>CID</td>
<td>CID</td>
<td>#it</td>
<td>path</td>
<td>time (m)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.</td>
<td>4550</td>
<td>3140</td>
<td>?</td>
<td>?</td>
<td>5018</td>
<td>2749</td>
<td>?</td>
<td>?</td>
<td>7517</td>
<td>4201</td>
<td>76</td>
<td>30</td>
<td>452</td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>8711</td>
<td>7747</td>
<td>1</td>
<td>1</td>
<td>5808</td>
<td>3140</td>
<td>19</td>
<td>19</td>
<td>7873</td>
<td>5385</td>
<td>24</td>
<td>16</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>7756</td>
<td>7913</td>
<td>3</td>
<td>3</td>
<td>9932</td>
<td>4550</td>
<td>?</td>
<td>?</td>
<td>8773</td>
<td>4362</td>
<td>14</td>
<td>14</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>8999</td>
<td>6518</td>
<td>18</td>
<td>18</td>
<td>3018</td>
<td>2749</td>
<td>?</td>
<td>?</td>
<td>7512</td>
<td>3081</td>
<td>?</td>
<td>7</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>6219</td>
<td>3042</td>
<td>7</td>
<td>8</td>
<td>9508</td>
<td>2944</td>
<td>7</td>
<td>7</td>
<td>8773</td>
<td>5385</td>
<td>24</td>
<td>16</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>5094</td>
<td>3276</td>
<td>7</td>
<td>7</td>
<td>8983</td>
<td>8041</td>
<td>7</td>
<td>7</td>
<td>7512</td>
<td>3081</td>
<td>22</td>
<td>22</td>
<td>31</td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>9679</td>
<td>2574</td>
<td>6</td>
<td>6</td>
<td>7747</td>
<td>6119</td>
<td>4</td>
<td>4</td>
<td>6491</td>
<td>4687</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td>6168</td>
<td>3454</td>
<td>38</td>
<td>18</td>
<td>8646</td>
<td>4644</td>
<td>17</td>
<td>6</td>
<td>5537</td>
<td>4198</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>10.</td>
<td>8393</td>
<td>3242</td>
<td>36</td>
<td>33</td>
<td>7956</td>
<td>6275</td>
<td>2</td>
<td>2</td>
<td>2749</td>
<td>2421</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>11.</td>
<td>4983</td>
<td>2570</td>
<td>9</td>
<td>9</td>
<td>8273</td>
<td>7913</td>
<td>4</td>
<td>4</td>
<td>9822</td>
<td>3830</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>13.</td>
<td>9362</td>
<td>5955</td>
<td>12</td>
<td>12</td>
<td>8274</td>
<td>4655</td>
<td>5</td>
<td>5</td>
<td>9312</td>
<td>4774</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>14.</td>
<td>6211</td>
<td>5781</td>
<td>4</td>
<td>4</td>
<td>8393</td>
<td>4990</td>
<td>36</td>
<td>31</td>
<td>9312</td>
<td>4593</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>15.</td>
<td>4558</td>
<td>3344</td>
<td>8</td>
<td>8</td>
<td>5712</td>
<td>5578</td>
<td>?</td>
<td>?</td>
<td>9213</td>
<td>2377</td>
<td>28</td>
<td>28</td>
<td>441</td>
<td></td>
</tr>
<tr>
<td>16.</td>
<td>5842</td>
<td>5838</td>
<td>7</td>
<td>11</td>
<td>5684</td>
<td>4735</td>
<td>?</td>
<td>?</td>
<td>7105</td>
<td>3331</td>
<td>17</td>
<td>17</td>
<td>31</td>
<td></td>
</tr>
</tbody>
</table>

B. Experiments

The ability of Molpher to find its way in the space demonstrates Table I. For each dataset, four columns are shown. The first two columns represent in order PubChem compound ID of the starting and of the target molecule. The next column represents the number of iterations needed by Molpher to find a path from the starting molecule to the target one. If Molpher did not find a path, a question mark is present instead. Then the length of the resulting path is presented and final column shows the time in minutes needed to find the path.

The contradiction between the number of the iterations and the path length results from the pruning process. In case when the values do not correspond there has to be at least one molecule (point) on the path whose neighbor has not been generated in the first pass through that point. Thus, first a wrong neighborhood was explored and when it turned out that the direction is not promising, a new, possibly correct, direction leading from that point was investigated.

If the amount of the time in which Molpher had to find the path was not limited, the path would be found between each two molecules. But since the chemical space is immense, we had to restrict the time in which the solution had to be found to 8 hours. If the path had not been found until that time, the process was designated as unsuccessful. If the time was restricted to less than 8 hours, the results on each of the datasets would deteriorate. The same stands for increase of the time amount. Therefore we emphasize that the results are only illustrative and can hardly be understood as an objective quantifier of Molpher’s effectiveness. To demonstrate this fact we ran Molpher on the first pair of the $D_1$ dataset without restricting its runtime or number of iterations. It turned out that Molpher is able to find the path of length 27 in 765 minutes (less than 13 hours). The length of the path indicates that by different setting of the algorithm’s parameters (such as cnt_morphs, cnt_accept, ...) the path could be found in less time.

In general, it can be seen that Molpher’s ability to find a way is proportional to the difficulty of the dataset, as expected. Specifically, Molpher found a path in given time (8 h) in 75 % cases for the $D_1$ set, in 60 % cases for the $D_2$ set and in 35 % cases for the $D_3$ set.

IV. DISCUSSION

By investigating the pairs where Molpher was unsuccessful, we analyzed the main determinants of the ability of finding a path in reasonable time.

Probably the most important factor is the proper molecule descriptor selection together with suitable choice of the distance function. This is evident when realizing that the descriptor and the distance function represent the only tool for navigating in the space. The hashed fingerprints presented in section II-C represent a proper choice although without introducing the statistical component in the final score, the fingerprints alone are not sufficient. One problem when the fingerprints do not fit the human intuitive was already discussed in section II-C.

Another drawback, we encountered, was the presence of...
false positives. In such a case a molecule is pronounced to be the target (zero distance to the target) although it differs from it. One can prevent false positives by declaring a molecule as the target one when its canonical SMILES string is identical to the canonical SMILES string of the target. In that way, we are able to avoid the false positives but at that point we absolutely lose control over the guiding process and the correct path can be found only by pure chance. False positives often occur when the target consists of many identical substructures. Then, if the tested molecule shares all the shapes of the bigger, target, structure, it is wrongly declared as identical with the target. This incorrect behavior was also corrected by introducing the statistics $\delta$.

Another inconvenient feature of the hashed fingerprints follows from their preference of large and, more importantly, complicated molecules. Large and complicated molecules have higher probability to contain paths existing also in the target molecule. Even if they do not correspond to the target molecule entirely, their distance to the target is low. This behavior is partially corrected by the introduction of the statistics into $\delta$ but the tendency is still present. Fig. 4 demonstrates this problem. The left molecule is molecule with the best score reached when trying to find a path between molecules from pair 9 from $D_3$. Their distance is quite low (0.16) but the structures are clearly very different. This favoring of complicated molecules increases the probability to slip into local optima since it "pulls" the computation in wrong way. The pruning procedure ensures that finally the error will be corrected but the restricted time causes the path-finding process to fail.

Another possibility is to utilize so-called counted fingerprints which count the number of occurrences of the substructures. But implement also derived or aggregated operators then these operators may behave as shortcuts in the space taking us over suboptimal parts of the space. The starting and target molecule of 5-th pair in $D_1$ can serve as an example (see Fig. 5). According to Tab. I, the number of operations needed for converting the starting molecule to the target one was 8. This definitely does not represent the shortest path because only three operators suffice (MA,BR,MA). But since such a path goes through non-optimal parts of the space Molpher was not able to identify it with given parameter settings. If we implemented "interchange atoms and their neighborhoods" operator to find the path, a single interchange of the oxygen atom for the carbon (together with the double bonded oxygen) would be sufficient. Although it might look simple, such an operation is not so trivial to implement since it is not obvious what part of the molecule should be moved together with the atom being interchanged.

![Figure 4](image1.png)  
**Figure 4:** The closest identified molecule (a) when searching for a path to the target (b) between the 9-th pair in $D_3$.

The set of morphing operators should be also carefully chosen. The suitable choice of operators can decrease the runtime of the algorithm. If we do not stick to basic operators but implement also derived or aggregated operators then these operators may behave as shortcuts in the space taking us over suboptimal parts of the space. The starting and target molecule of 5-th pair in $D_1$ can serve as an example (see Fig. 5). According to Tab. I, the number of operations needed for converting the starting molecule to the target one was 8. This definitely does not represent the shortest path because only three operators suffice (MA,BR,MA). But since such a path goes through non-optimal parts of the space Molpher was not able to identify it with given parameter settings. If we implemented "interchange atoms and their neighborhoods" operator to find the path, a single interchange of the oxygen atom for the carbon (together with the double bonded oxygen) would be sufficient. Although it might look simple, such an operation is not so trivial to implement since it is not obvious what part of the molecule should be moved together with the atom being interchanged.

![Figure 5](image2.png)  
**Figure 5:** The starting (a) and target (b) molecule (b) between the 5-th pair in $D_1$. With appropriate morphing operator, the resulting path might be only one step long.

V. _Conclusion and Future Work_

In this paper we presented a solution called Molpher for chemical space exploration. Molpher is the only documented algorithm, we are aware of, solving the problem. It is designed to find a way in the chemical space from a starting molecule to a target one by sequence of morphing operators defining individual modifications of the starting molecule on its way to the target one. We described in detail the morphing operators together with the algorithm for navigation in the space. To be able to quantify Molpher’s qualities in terms of ability to find a way in the space, we proposed three different difficult sets of start/target molecules. These datasets, or their extensions, can be used in the future to qualitatively quantify impacts of future modifications of Molpher or to compare it to other solutions.

The presented approach forms a self-contained algorithm which can be used as is. On the other hand we see many directions in which it can be improved to provide both more efficient and effective service.

As mentioned above, the descriptors characterizing particular molecules and the distance used for guiding the
morphing process are essential for finding a path from source to target. Therefore, in the future we would like to focus on designing descriptors well suited for the needs of navigation in the chemical space. Qualities of the proposed descriptors and distance can then be easily tested using Molpher.

To allow Molpher to find a solution more effectively we would like to extend the set of operators. We would like to implement more high-level operators such as mutual substructure substitution, bond sticking, etc.

We would also like to test Molpher against a bigger database using parallel approaches in the candidate morphs generation process. By introducing parallelism, we might be able to scan much larger part of the chemical space in acceptable time. The reasonable runtime of finding a path is inevitable before a usable interactive version of the algorithm can be developed.

We also plan to extend Molpher’s chemical intelligence by enriching the set of allowed atoms (to S, P, Cl, F, ...), and by including the stereochemistry and chirality rules.

Finally, we plan to extend usability of Molpher by implementing the idea of interactive guiding through the chemical space using sets of decoys. Then not only the distance to the target is taken into account but also the distance to the molecules from the decoy set. In such a concept the user could specify subsets of the space or a set of molecules to which she would like to get close during the morphing process. We plan to provide the user with the ability to interactively switch these sets on or off and thus interactively guide the exploration process. Such a functionality might be useful, for example, when the user would like to find a path being close to existing compounds or more specifically, to biologically active compounds.

ACKNOWLEDGMENT

This work was supported by Czech Science Foundation project Nr. 201/09/0683 and by the Ministry of Education of the Czech Republic MSM6046137302.

REFERENCES


