TARGET-ORIENTED GENERIC FINGERPRINT-BASED MOLECULAR REPRESENTATION

Petr Skoda and David Hoksza

Faculty of Mathematics and Physics,
Charles University in Prague, Prague, Czech Republic
skoda@ksi.mff.cuni.cz
hoksza@ksi.mff.cuni.cz

ABSTRACT

The screening of chemical libraries is an important step in the drug discovery process. The existing chemical libraries contain up to millions of compounds. As the screening at such scale is expensive, the virtual screening is often utilized. There exist several variants of virtual screening and ligand-based virtual screening is one of them. It utilizes the similarity of screened chemical compounds to known compounds. Besides the employed similarity measure, another aspect greatly influencing the performance of ligand-based virtual screening is the chosen chemical compound representation. In this paper, we introduce a fragment-based representation of chemical compounds. Our representation utilizes fragments to represent a compound where each fragment is represented by its physico-chemical descriptors. The representation is highly parametrizable, especially in the area of physico-chemical descriptors selection and application. In order to test the performance of our method, we utilized an existing framework for virtual screening benchmarking. The results show that our method is comparable to the best existing approaches and on some data sets it outperforms them.

KEYWORDS

Virtual screening, Molecular representation, Molecular fingerprints

1. INTRODUCTION

The main method to identify new leads in the drug discovery process has traditionally been medium or high-throughput screening (HTS). In this experimental process, a large number of chemical compounds can be screened against a specific target to identify compounds which trigger a response in this target. Some of the HTS approaches can guarantee throughput up to about 100.000 compounds per second [1] by using the combinatorial libraries. Obviously, the throughput in such cases is not an issue anymore. However, management of such large libraries can be difficult and economically unfeasible since every new compound brought into the screening process increases its price.

The in-silico answer to the growing size of chemical databases is the so-called high-throughput virtual screening (HTVS). It allows fast screening of large libraries, which may contain up to tens of millions chemical compounds, without the need of physically own the compounds. An additional bonus which relates to the HTVS is the ability to screen even virtual libraries. I.e., one can easily predict bioactivity of compounds residing in not yet well explored parts of the chemical space [2].

While the limits of HTS are given by the technology, the HTVS, since it is a simulation of a real world approach, is limited by the available information about ligands [3]. The available knowledge then dictates how HTVS is utilized. Since unlike HTS, HTVS can suffer by both false positives and false negatives it is commonly used as a pre-step to the standard HTS in the early-stages in the drug discovery pipeline. The HTVS is used to prioritize large chemical libraries which narrows down the set of compounds to be forwarded to HTS. Usefulness of complementing HTS with HTVS has been supported by several studies [4], [5]. The virtual screening approaches can be classified as ligand-based virtual screening (LBVS) and structure-based virtual screening (SBVS) [6], [7]. The choice of which approach to utilize depends on information about the task at hand. If we know the three-dimensional structure of the biological target we can use SBVS methods [8], [9]. The SBVS is based on docking and includes two steps: positioning the ligand into the target active site (docking) and scoring the pose. However, this information is often not available in sufficient quality or it is not available at all. In such a case the ligand-based virtual screening method is the method of choice.

1.1 Ligand-based virtual screening

In LBVS, only the information about known bioactive ligands (triggering response in the given biological target) is required. The LBVS is built around the concept that similar structures carry out similar functions more often than dissimilar ones. This assumption is based on the shape and physicochemical complementary of the ligand and target commonly called key-and-lock principle [10] or similar property principle [11]. Thus, given the known active (and possibly also inactive) compounds LBVS methods prioritize compounds that are more likely to have desired functionality/features, based on the similarity to the known active molecules.

In the first step of LBVS, a computer-based representation is calculated for the known bioactive ligands as well as for all the molecules in a library to be searched for new bioactive compounds. In the second step, the representation of ligands can be aggregated and used as a query or individual representations are used directly for searching the library. As the last step, the library is sorted with respect to the similarity to the query ligand(s). It is assumed that the high-scoring compounds bind to the target with high probability due to the similarity principle.

One can come up with various classification of LBVS approaches. For example, Taboureau at. al. [7] divide LBVS into five classes based on the utilized molecular features: alignment-based, descriptor-based, graph-based, shape-based and pharmacophore-based.

While the methods might differ in the specifics of how to approach the identification of bioactive compounds, most of them employ a feature extraction step where the molecular descriptors are identified and encoded into some kind of representation. This is then used as a representation of the molecule in the virtual screening. Among the commonly used features are those which reflect structure or capture computed or experimentally measured physico-chemical properties. Currently, there exists a plethora of descriptors to be utilized in virtual screening [12], [13]. They differ not only in their semantics but also in the computational complexity. The excess of descriptors is the consequence of the fact that none of the descriptors can be generally declared as superior to the rest. The features discriminating active and inactive compounds simply depend on the specific target which varies in every screening campaign. It follows that it is vital to the success of a virtual screening campaign to capture such features which represent the molecules well in terms of their discriminative capability. This is the main motivation for our work. It is out of question that the correct choice of features greatly influences the outcome of a virtual screening campaign. However, we moreover believe that the choice of descriptors should be context-aware that is it should be dependent on the investigated target. Therefore, in this paper we propose a general framework which allows the user to parameterize the molecular representations.

1.2 Fingerprints

A common type of descriptors are the 2D fingerprints (fingerprints) capturing the structure of given chemical compound in the form of a bitstring. Every structural feature is mapped to a position in the string. Such representation is suitable for large-scale virtual screening campaigns since it allows fast comparison of two molecules (bitstrings).

Thus, the main idea behind the fingerprints is to encode the existence of a given (structural, pharmacophore, …) feature to a position in a bitstring. The features to be encoded commonly include molecular fragments which are continuous substructures of a given molecule. There are two main approaches to fragment extraction: path-based (Topological Torsions fingerprints), neighbourhood-based (Circular Fingerprints or Extended connectivity fingerprints).

The Topological torsions fingerprints [14] (TT) use paths of length four (quaternions). The information about types, nonhydrogen connections and number of pi-electrons is used to calculate the index of given path.

In Extended Connectivity Fingerprints (ECFPs) and Functional Connectivity Fingerprints (FCFPs) an atom is described in terms of its neighbouring atoms up to a certain radius. Hert [15] has shown that such descriptors can be effective in similarity searching applications. The extended connectivity of an atom is calculated using a modified version of the Morgan algorithm [16] where the atom code is combined with the codes of its neighbours to establish the final atom description.

To map a fragment into a position in the bitstring representation, a mapping function needs to be utilized. The simplest solution is the dictionary-based approach where a predefined dictionary of fragments and their mapping into the bitstring is utilized. However, this allows to represent only a limited set of fragments in the bitstring. Another solution is to map every possible fragment into a constant-sized bitstring. However, since the size of a bitstring representation uses to be an order of magnitude smaller in comparison to the number of all possible fragments, typically a modulo function is applied. This allows to obtain a bitstring position for every possible fragment. On the other hand, two different fragments with different indexes can be mapped to the same position in a bitstring. This situation is called the collision. The fingerprints that utilize this approach form the family of hashed fingerprints.

2. METHOD OUTLINE

In this work we introduce vector fingerprints (VectorFp), a new approach to the representation of chemical compounds and their comparison. As mentioned above, our goal is to provide a modular molecular representation for LBVS allowing to be parametrized based on the task at hand. The basis of VectorFp molecular representation form structural fragments. But unlike other descriptors, VectorFp allows the fragments to be labeled by user-defined physico-chemical properties. Moreover, the representation was designed with the emphasis on the ability to use it with existing similarity measures for bitstrings. Thus, VectorFp is designed as a generic representation that needs proper parametrization before it can be used.

2.1. VectorFp structure

In order to maintain compatibility with existing fingerprint methods we decided to choose the bitstring as the representation for VectorFp. The advantage is that we can utilize existing well-established similarity measures, LBVS processes, and benchmarking platforms in order to get comparison of our method to the other fingerprints.

VectorFp (Figure 1) is basically an array (outer array), where each cell represents one (or more in case of a collision) fragment(s). Each cell of the main outer array contains another array (inner array). The purpose of the inner array is to store the selected descriptors of a respective fragment(s). As mentioned, these descriptors are physico-chemical properties of fragments that are converted into bitstring representation.

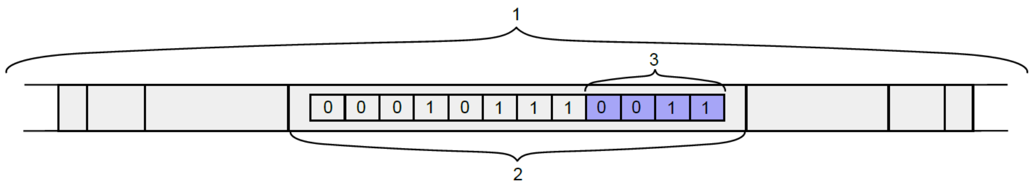


Figure 1. Structure of VectorFp. 1 - outer array, 2 - cell with inner array, 3 - bits representing single descriptor

Generally, physico-chemical descriptors can take various ranges of values being typically integer or float data types. The process of conversion of descriptors into a bitstring is secured by so called conversion methods. In our current implementation we use the same conversion method for all descriptors. It gets minimum and maximum value for a given descriptor and then uses binning which results in an integer value to be used as the descriptor value to be stored. The integer value is then encoded into a bit array using unary encoding. The binning is the formation of a set of disjoint intervals (bins) that represent the possible values. The bin index is finally encoded into a binary representation. In VectorFp we decided to use unary coding. The choice of unary coding instead of, e.g. classical binary coding, stems from the typical choice of similarity functions used when comparing bitstring molecular representations. The most commonly used similarity functions basically assess similarity to a pair of bit strings based on the number of common and differing bit positions. These measures assume that the bits are independent which holds when every bit corresponds to the existence or nonexistence of a molecular substructure. However, when the binary image of a substructure spans multiple bit positions (inner array) the positions are dependent. Using the binary coding with such similarity measure is then not valid. Let us consider a situation when the binary representation takes 4 bits. Then if the distance/similarity is based on the number of common bits bin 4 (0100) is from bin 1 (0001) in the same distance as, e.g., bin 2 (0010). Which should not hold since the bin indexes approximate quantitative characteristics. However, when using unary coding bin 1 gets the code 1000, bin 2 gets the code 1100 and bin 4 gets the code 1111. Then, using the same similarity measure, bin 2 is more similar to bin 1 than bin 4 as one would expected.

2.2. VectorFp generation

1. The VectorFp representation computation for a given molecule consists of five main steps:
2. Extract fragments from the molecule and compute indexes (positions in the bitstring representation) for those fragments.
3. For each fragment compute its physico-chemical descriptors.
4. Convert all fragments descriptors into bitstrings.
5. Create the fragment bitstring representation from its respective descriptor representations.

6. Combine the individual fragments representations (bitstrings) together and assemble the representation of the molecule. The representations of fragments are stored into cells determined by the index computed in step 1.

As the VectorFp size is limited, the index computed in step 1 must be modified by application of the modulo operation (hashing). As a consequence a collision may occur. In order to solve collisions VectorFp utilizes the bitwise logical or to merge representations of multiple fragments together. The advantage of this method is simplicity and the fact that the results (fragment bitstring) are the same for different permutations of the same fragments. The drawback of selected approach is, that created fragment representation does not have to represent existing fragment. This can be problem during similarity comparison of two VectorFps. If both molecules (their VectorFps representations) have the same fragments in single cell, then everything is in order, but if one molecule has difference number of fragments in given cell then the other molecule, for example one and two, the comparison still compare fragment representation to fragment representation. In this case we compare existing fragment to some imaginary aggregated fragment.

3. PARAMETERIZATION

From the description of VectorFp one can notice that there are places where the approach is not fully specified: fragment extraction, descriptor selection and conversion. The named areas and some more create space for parameterization of VectorFp. VectorFp can be seen as a generic representation or frame. The parameterization determines the efficiency of the final VectorFp-based molecular representation.

3.1 VectorFp size

One of the parameters is the size of VectorFp. The size is determined by two variables: size of the inner array and the number of cells in the outer array. The final size of vectorFp representation is therefore *size of inner array * size of outer array*. So for example if we use 1024 cells for the outer array, then a 4 bit increase of the inner array size will result in 4096 bit increase of the resulting representation size.

3.2 Fragment extraction

In the current implementation we utilize RDKit's [17] algorithm to extract the fragments from a molecule get their positions in the bitstring. The algorithm uses RDKit's Morgan Fingerprint which is based on the Morgan algorithm. Morgan Fingerprints use the following features to calculate a fragment's position in the bitstring: donor, acceptor, aromatic, halogen, basic, acidic. The RDKit provides the possibility to modify this feature list and thus change the fragment indexes. This can be also viewed as a possible parameterization of VectorFp. Another possibility is to use paths (like TT fingerprints) instead of neighbourhoods.

3.3 Fragment representation

Each fragment is represented by an inner array (bitstring). The size of this array determines how many information can be stored about each fragment. By setting the size of the inner array to one we get the classical fingerprints. The selection of used descriptors, conversion method and number of bits in inner array is also part of the parameterization. The descriptor selection and conversions are in our opinion the most important parts of the parameterization and have a great influence on the performance of the method. The selected descriptors include, for example, the number of heavy atoms, logP, the presence of a fragment or, in an extreme case, other fingerprint can be used as a fragment's descriptor and inserted into VectorFp. There is also the possibility to stress certain descriptor by multiplying its value. For example, let us have two different descriptors, we use them both in our parameterization but we replicate one of them. In this case, the replicated descriptor has more weight and can be seen as the main one. The second one (nonreplicated) descriptor can serve as a fine tuning mechanism.

4. EXPERIMENTS

For experimental evaluation we used the recently published framework for benchmarking LBVS approaches by Riniker et al. [18]. The framework is written in Python [19] and uses RDKit [17] as the underlying chemical framework. It comes with a predefined set of fingerprints, similarity methods (Dice, Tanimoto, Cosine, Russel, Kulczynski, McConnaughey, Manhattan, RogotGoldberg) and quality measurement methods (Area Under Curve (AUC) of Receiver Operating Characteristic curve (ROC), Enrichment Factor (EF), Robust Initial Enhancement (RIE) [20], Boltzmann-Enhanced Discrimination of ROC (BEDROC) [21]). The framework simulates LBVS on pooled targets from three data sets representing 88 targets in total. The three data sets include Database of Useful Decoys (DUD) [22], ChEMBL [23] and Maximum Unbiased Validation (MUV) [24]. For each target a set of known actives and inactives (decoys) is available. As the framework aims to high reproducibility of experiments it also contains a predefined random selection of actives and decoys. Thanks to that, the simulation of LBVS is deterministic and can be easily reproduced by any researcher.

However, one of the drawbacks of the framework is that it is designed to use the same method with the same parameterization for all the data sets. There is no learning phase per dataset. Such phase could be useful for benchmarking of methods including a learning phase [25]. The absence of learning phase influences performance of our method in a negative way as our method needs a proper parameterization that differs based on the task (dataset) at hand. Still, we decided to not modify the benchmarking platform and to use a single parameterization over all data sets as the determination of the right parameterization is not the goal of this article. The problem of correct parameterization and feature selection is a separate topic.

Riniker et al. [18] recommend to use at least two different benchmarking methods, for example AUC and BEDROC as the AUC alone is considered to be insufficiency sensitive. On the other hand, the advantage of the AUC in comparison to some other methods is that it is non-parametric. Thus, it can be easily used to give a basic idea about the performance of tested method especially in a large scale evaluation. From this reason, we decided to show only AUC values in the following experimental evaluation.

4.1. Comparison to existing methods

In this section, we presents the comparison of *VectorFp* with other fingerprints from selected benchmarking framework. We used *VectorFP* with the best found parameterization (aggregated over all targets). However, we emphasize that the *VectorFp* performance strongly depends on the selected parameterization (see section IV-B) and since the parameterization optimization is a hard (and separate) problem, there is still room for improvement. Moreover, in this comparison we use a single parameterization for all targets which is not the optimal and intended use of *VectorFp*, but we find it useful in order to get a rough comparison with the other existing methods. To denote the other fingerprints we use abbreviations from the original article [18] containing also the details about the remaining fingerprints.

The best parameterization we obtained in our experiments in terms of average *auc* (average of *auc* over all data sets) was *nHBDon_Lipinski*,*nN*. This parameterization utilizes two descriptors –

nHBDon_Lipinski and *nN*, where each descriptor occupies 16 bits in the final representation. We denote this parameterization further in the text as *vectorFp*.

As already stated, the VectorFp is designed as a generic representation that should be rather used with parameterization based on the given task. In order to demonstrate the potential of *VectorFp*, we defined virtual VectorFp (*vVectorFp*). To get the results for *vVectorFp*, we select the best tested parameterization for every dataset. Thus, *vVectorFp* can be understood as *VectorFp* with an oraculum that gives us the best encountered parameterization for given target.

As for the source of descriptors for labelling the extracted fragments, we used the PaDEL [26] tool. PaDEL is capable of generating about 770 2D descriptors that can be easily utilized in *VectorFp*. To convert the descriptor values into the bitstring in the inner arrays of *VectorFp* we use the binning and unary coding.

As the results show (Table 1.), *vectorFp* (with the *nHBDon-Lipinski*,*nN* parametrization) is, in terms of *auc*, the best fingerprint for 8 out of the 88 data sets and it ends up on position 9.966 on average. The best obtained average position is 8.092 reached by the *TT* fingerprint.Thus, although the single parameterization is used for multiple data sets, it is clearly comparable with the best existing approaches. On some data sets, our method is superior to all the other methods. The performance differs throughout all the data sets (Table 2.).

Table 1. Aggregated performance statistics of *vectorFp* and *vVectorFp* with respect to other fingerprints

name	average AUC	number of best results	average position
tt	0.8034	12	8.09
hashap	0.7701	11	14.20
rdk6	0.7821	10	12.55
vectorFp	0.7890	8	9.97
laval	0.7798	7	12.91
avalon	0.7755	7	14.03
rdk7	0.7407	6	17.62
ap	0.7914	5	10.25
hashtt	0.7973	4	9.31
rdk5	0.7827	3	12.25
lfcfp6	0.7631	3	14.71
fcfp2	0.7457	3	18.17
ecfc6	0.7795	3	11.79
fcfp4	0.7643	2	14.71
fcfc6	0.7625	2	15.10
lfcfp4	0.7620	2	15.23
lecfp4	0.7606	2	15.03
lecfp6	0.7581	2	16.03
fcfp6	0.7657	1	14.59
ecfp2	0.7522	1	17.68
fcfc2	0.7435	1	19.55
maccs	0.7333	1	20.08
ecfc4	0.7798	0	11.61
ecfc2	0.7739	0	13.68
fcfc4	0.7603	0	15.77
ecfp4	0.7582	0	16.03
ecfp6	0.7573	0	16.68

ecfc0	0.7340	0	20.43
ecfp0	0.6463	0	27.91
vVectorFp	0.8174	31	4.37

As can be seen most of the tested fingerprints perform reasonably well, in comparison to the others, on at least one dataset. Out of the three data sets, the MUV dataset shows up to be the hardest for *vVectorFp* as in 3 cases it performs strongly under average. However, the MUV dataset is the most difficult for every tested fingerprint. The goal of the MUV design is to generate sets with a spatially well distributed active and decoy molecules in a simple descriptor space. Moreover, another goal is to evenly distribute actives among the decoys which makes the MUV dataset difficult for virtual screening. The best tested parameterization for MUV shows up to be *naAromAtom16*,*ETA_BetaP_s16*,*minHsNH2* with the average *auc* on MUV being 0.6258 compared to *vectorFp* having the average *auc* of 0.6214.

As the VectorFp in fact utilizes one of the extended connectivity fingerprints (*ecfp*) as the underlying fingerprint, we were interested how it compares to the performance of other fingerprints from the same family. From this perspective our method performs well and outperforms most fingerprints from this family.

Our method was in term of average *auc* over all the data sets outperformed by *tt*, *hashtt* and *ap* fingerprints. All those fingerprints are based on different fragments than used in current version of VectorFp. *tt* and *hashtt* use paths of length four while *ap* use atom pairs. This suggest that the change of fragment extraction process (underlying fingerprint) may improve the performance of *VectorFp*.

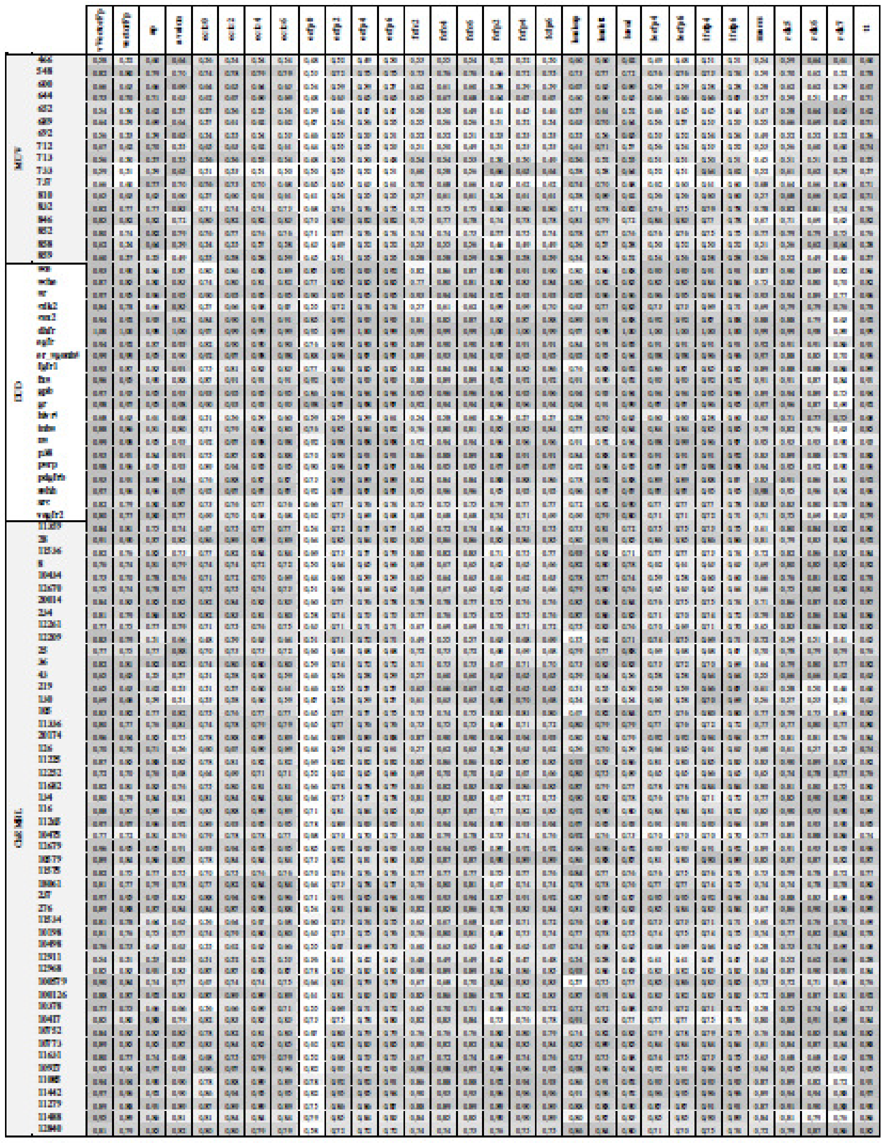
Notice, that *vVectorFp* is also included in the comparison. As it is not based on a single parameterization, the values presented in Table 1. were computed without the *vVectorFp*, and at the end the *vVectorFp* was added. Thus *vVectorFp* results did not influence the positions of other approaches. The vVectorFp outperforms all other methods in all the presented evaluation criteria (average *auc*, number of best results). We believe that this demonstrates the potential of *VectorFp* if parameterized properly. We emphasize again that there may be a better parameterization as we tested only a very limited subset of all possible parameterizations.

4.2. Parameterization

As a part of our experiments we systematically tested hundreds of different parameterizations focusing on various descriptors provided by PaDEL (see above). Although there are more ways of how to parameterize VectorFp, here we focused on descriptor selection only being the most result influencing part of the parameterization.

To test how the amount of used descriptors per fragment influences the discriminative power of the molecular representation, we started with just one descriptor per fragment and then added more. In the preparation phase, we extracted all fragments for all chemical compounds in every data sets of the benchmarking platform. For each fragment we computed all descriptors available in PaDEL. These descriptors were the subject of a basic descriptor analysis before running the experiments themselves. The goal of the analysis was to remove descriptors which clearly did not have enough discriminative power to be used for screening.

Table 2. Comparison of vVectorFp and vectorFp with other fingerprints. The colours show the relative performance of given fingerprint to others on given target (dataset). The grey cell represents the best result on given target while white represents the worst result.



As the first step of the analysis we dropped all the descriptors that were constant which resulted in the elimination of 258 descriptors. In the next step we utilized variance to decide which descriptors have the potential being a useful discriminator. A descriptor taking only two values has a low chance to well discriminate thousands of compounds. As a prestep to variance analysis we had performed normalization on every descriptor. First, we had removed outliers from every descriptor (values outside the second and third quantile), then we normalized the data into the [0, 1] interval using the min-max normalization. After the normalization we computed variance (var_{norm}) for each descriptor. Many descriptors ended up with $var_{norm} = 0$. For example in case of *nAcid* descriptor, about 96.7% of fragments have zero value. This does not leave much space for other values, and basically divides all fragments into just few categories (3 in case of *nAcid*). If we consider the second and third quantile only we get zero variance. This step eliminated 326 descriptors. Since we used these descriptors later in the experiments, we formed group from them called *constVarQ* (constant variance on quantiles). The remaining 185 descriptors were split into 4 groups of almost the same size based on the value of variance. The groups were called *var_00_25*, *var_25_50*, *var_50_75* and *var_75_100*.

4.2.1. Single descriptor

In the first step we evaluated the performance for selected descriptors from groups *var_00_25*, *var_25_50*, *var_50_75* and *var_75_100* and *constVarQ*. The descriptors in *constVarQ* group performed worst of all, as expected. This was caused by the fact that in many data sets the descriptors were constant and so had no discriminative power. However, despite the overall bad performance few exceptions emerged. For example, using the descriptor *nAcid* (number of acidic groups) for target 20174 resulted in *auc* 0.909. The *tt*, *hashtt* and *ap* scored 0.841, 0.8430 and 0.8450 respectively. This demonstrates that good performance can be reached even with a single simple descriptor. As for the target 20174, the best performance (0.9560) was obtained by *vectorFp*.

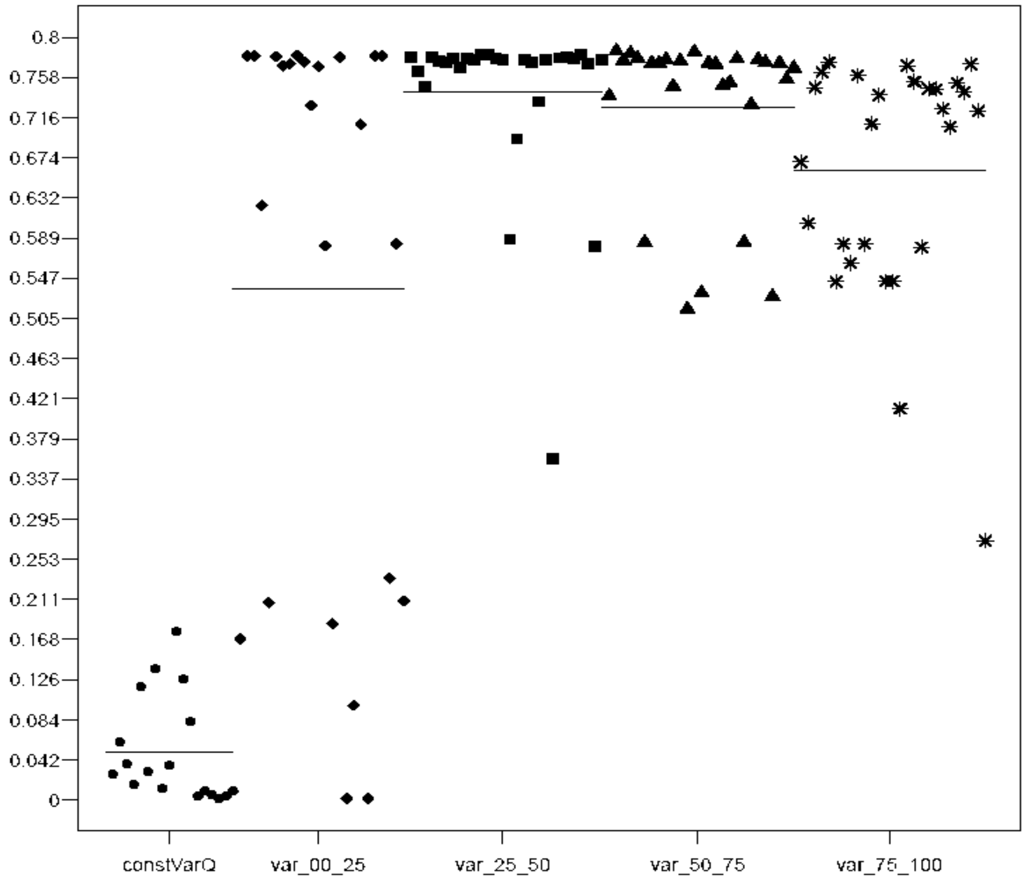


Figure 2. *auc* performance for single descriptor parameterization among the variability groups. The horizontal lines represent the average *auc* for given group.

The performance of all the descriptors shows Figure 2. where descriptors' data points in the same group share same shape. The X-axis corresponds to the individual descriptors while the Y-axis shows the average AUC over all the targets for each of the descriptors. The horizontal line then represents the average of the descriptors performance for each of the group. We can clearly see that the descriptors in the *constVarQ* show worse performance then descriptors in the other groups. In all the var groups we can identify several well performing descriptors. However, the group *var_00_25* contains many descriptors showing very poor performance. It follows that a descriptor with low variability is likely to perform poorly. On the other hand, the descriptors with high variability (group *var_75_100*) also lead to worse performance than the descriptors with moderate variability (groups *var_25_50* and *var_50_75*).

4.2.2. Multiple descriptors

In the next step, we first created pairs and then triplets of descriptors and used them to label the fragments. Thus, in the previous step each fragment was labelled by exactly one descriptor but in the second step pairs and triplets of descriptors were utilized. Our hope was that the performance would increase when using tuples in contrast to using single descriptors alone.

Since we did not have sufficient computational resources to test every possible pair and triplet of descriptor we implemented a filter. The purpose of the filter is to filter out such tuples which are unlikely to lead to best results. The filter utilizes AUC (*auc*) of single descriptors and correlation (*cor*) between pairs of descriptors.

Let n denote the number of descriptors that should be used in the parameterization (in our case n is 2 or 3). Let auc_i denote the average AUC for *i*-th descriptor (out of n) and $cor_{i,j}$ the correlation between AUC values of *i*-th and *j*-th descriptor over data sets. Thus if two descriptors show similar AUCs over all data sets they have high correlation. In order for the tuple of descriptors to pass the filter, the following two conditions need to be satisfied:

$$\sum_{i=0}^{n} auc_i > auc_{Level}$$

$$cor_{LevelMin} < \max_{0 \le i,j \le n, i \ne j} cor_{i,j} < cor_{LevelMax}$$

The filter is parameterized by the values auc_{Level}, $cor_{LevelMin}$ and $cor_{LevelMax}$. Using auc_{Level} simply prefers tuples consisting of descriptors which behave well when used alone. The idea behind restricting the correlation is that bringing together correlated descriptors would not result in new information and thus probably would not increase the discriminative power of the resulting molecular representation. We tried several parameterizations of the filter (see Table 3.) to get a reasonable number of pairs/triplets for our experiments.

The descriptors in 2B pairs are required to have *cor* between 0.47 and 0.6. The lower bound for *cor* secures that the pairs in 2A and 2B are different. As the trade of, the required *auc* needs to be slightly higher. As the 2B group was selected with less stress on *cor* it was expected that the paired descriptors would have more similar results over the data sets. The goal of different parameterizations of the filter was to test which combination of correlation and quality parameters leads to better results. The same holds for the groups of triplets.

Table 3. Specification of tested filters

group name	auc_{Level}	$cor_{LevelMin}$	$cor_{LevelMax}$	group size
2A	1.48	0.00	0.47	102
2B	1.487	0.47	0.60	40
3A	2.08	0.00	0.50	41
3B	2.20	0.50	0.60	48

How different number of descriptors used to label the fragments influence the *auc* show Table. 3. and Figure 4. The Table 3. average *auc* for parameterizations using single descriptors (the *var* groups), pairs of descriptors (the 2A and 2B group) or triplets of descriptors (the 3A and 3B group) to label the fragments. As the results show the 2A-filtered pairs of descriptors perform on average significantly better those based on the 2B filter.

The difference between 2A and 2B is much higher than in case of 3A and 3B. As Table 4. shows, the performance of triplets of descriptors is somewhere between the 2A and 2B based pairs. From the Figure 3 it seems that the performance of triplets of descriptors is more variable than in case of pairs. We believe that it is the consequence of the fact that there are more triplets of descriptors than there are pairs. Therefore, it is more difficult to identify the correct triplets. Thus the variance in the results of triplets of descriptor is simply due to the imperfection of the descriptor selection procedure. In case of both pairs and triplets of descriptors, the group with more restricted *cor* seems to provide better results, especially in terms of worst case performance.

Table 4. Average reached *auc* for different numbers of descriptors per fragment

group name	average auc
var_00_25	0.535
var_25_50	0.741
var_50_75	0.725
var_75_100	0.659
2A	0.783
2B	0.780
3A	0.782
3B	0.782

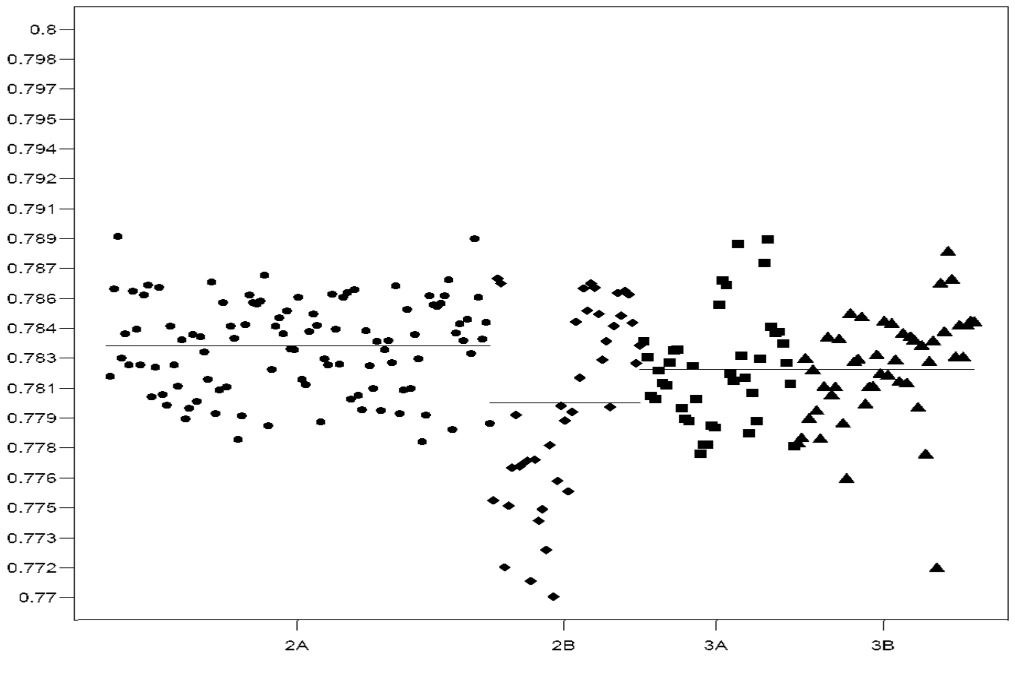


Figure 3. *auc* performance for two and three descriptors parameterizations among groups. The horizontal lines represent average of *auc* for given group.

4. CONCLUSION

In this work we presented a generic molecular representation called VectorFp. The representation was tested using the recently published benchmarking platform for LBVS. Therefore, the results should be easily reproducible and results were easily comparable with other existing commonly used molecular representations. The main motivation for our work was to provide a molecular representation which could be parameterizable with specific descriptors suitable for given biological target. Even though we operated within the boundaries of the benchmarking framework by forcing us to fix the parameterization our method it still outperformed most of the existing methods.

We also showed the potential of our method by creating a virtual representation *vVectorFp* where the best encountered parameterization for given target was used. This representation clearly outperformed all the existing approaches showing that potential strength of the method with correct parameterization. As a virtual representation demonstrates the potential of VectorFp if the right parameterization is used, it follows that the research on the parameterization will be the main direction of our future work on VectorFp. Moreover, we tested only up to three descriptors per parameterization while there are, beside the computer memory, virtually no restrictions of how many descriptors can be used. Finally, we also plan to investigate the possibility of stressing the importance of a single descriptor by its multiple application.

ACKNOWLEDGEMENTS

This work was supported by Grant Agency of of Charles University [project Nr. 154613], by SVV-2014-260100 and by the Czech Science Foundation grant 14-29032P.

REFERENCES

[1] B. Battersby and M. Trau, (2002) “Novel miniaturized systems in high throughput screening,” Trends in Biotechnologi, vol. 20, pp. 167–173

[2] J. Besnard, G. F. Ruda, V. Setola, K. Abecassis, R. M. Rodriguiz, X. P. Huang, S. Norval, M. F. Sassano, A. I. Shin, L. A. Webster, F. R. Simeons, L. Stojanovski, A. Prat, N. G. Seidah, D. B. Constam, G. R. Bickerton, K. D. Read, W. C. Wetsel, I. H. Gilbert, B. L. Roth, and A. L. Hopkins, (Dec 2012) “Automated design of ligands to polypharmacological profiles,” Nature, vol. 492, no. 7428, pp. 215–220

[3] S. Ekins, J. Mestres, and B. Testa, (2007) “In silico pharmacology for drug discovery: methods for virtual ligand screening and profiling,” British Journal of Pharmacology, vol. 152, pp. 9–

[4] R. S. Ferreira, A. Simeonov, A. Jadhav, O. Eidam, B. T. Mott, M. J. Keiser, J. H. McKerrow, D. J. Maloney, J. J. Irwin, and B. K. Shoichet, (Jul 2010) “Complementarity between a docking and a high-throughput screen in discovering new cruzain inhibitors,” J. Med. Chem., vol. 53, no. 13, pp. 4891–4905

[5] L. R. Vidler, P. Filippakopoulos, O. Fedorov, S. Picaud, S. Martin, M. Tomsett, H. Woodward, N. Brown, S. Knapp, and S. Hoelder, (Oct 2013)“Discovery of novel small-molecule inhibitors of BRD4 using structurebased virtual screening,” J. Med. Chem., vol. 56, no. 20, pp. 8073–

[6] K. Heikamp and J. Bajorath, (Jan 2013) “The future of virtual compound screening,” Chem Biol Drug Des, vol. 81, no. 1, pp. 33–40

[7] O. Taboureau, J. B. Baell, J. Fernandez-Recio, and B. O. Villoutreix, (Jan 2012) “Established and emerging trends in computational drug discovery in the structural genomics era,” Chem. Biol., vol. 19, no. 1, pp. 29–41,

[8] P. Ripphausen, B. Nisius, L. Peltason, and J. Bajorath, (Dec 2013) “Quo vadis, virtual screening? a comprehensive survey of prospective applications,” Journal of Medicinal Chemistry, vol. 53, no. 24, pp. 8461–8467

[9] P. Ripphausen, D. Stumpfe, and J. Bajorath, (Apr 2012) “Analysis of structure based virtual screening studies and characterization of identified active compounds,” Future Med Chem, vol. 4, no. 5, pp. 603–613

[10] W. L. Jorgensen, (Nov 1991) “Rusting of the lock and key model for protein-ligand binding,” Science, vol. 254, no. 5034, pp. 954–955

[11] M. A. Johnson and G. M. Maggiora, (1990) Concepts and Applications of Molecular Similarity. Wiley-Interscience

[12] L. Xue and J. Bajorath, (Oct 2000) “Molecular descriptors in chemoinformatics, computational combinatorial chemistry, and virtual screening,” Comb. Chem. High Throughput Screen., vol. 3, no. 5, pp. 363–372

[13] J.-L. Faulon and A. Bender, (Apr 2010) Handbook of Chemoinformatics Algorithms (Chapman & Hall/CRC Mathematical & Computational Biology), 1st ed. Chapman and Hall/CRC

[14] R. Nilakantan, N. Bauman, J. S. Dixon, and R. Venkataraghavan, (1987) “Topological torsion: a new molecular descriptor for sar applications. comparison with other descriptors,” Journal of Chemical Information and Computer Sciences, vol. 27, no. 2, pp. 82–85,

[15] J. Hert, P. Willett, D. J. Wilton, P. Acklin, K. Azzaoui, E. Jacoby, and A. Schuffenhauer, (2004) “Comparison of topological descriptors for similarity-based virtual screening using multiple bioactive reference structures,” Organic and Biomolecular Chemistry, vol. 2, pp. 3256–3266

[16] H. L. Morgan, “The generation of a unique machine description for chemical structures-a technique developed at chemical abstracts service.” Journal of Chemical Documentation, vol. 5, no. 2, pp. 107–113

[17] (2014)“Rdkit: Cheminformatics and machine learning softwares,” [Online]. Available: http://www.rdkit.org

[18] S. Riniker and G. Landrum, (2013) “Open-source platform to benchmark fingerprints for ligand-based virtual screening,” Journal of Cheminformatics, vol. 5, no. 1, p. 26,

[19] (2014) “Python,” [Online]. Available: https://www.python.org/

[20] R. P. Sheridan, S. B. Singh, E. M. Fluder, and S. K. Kearsley, (2001) “Protocols for bridging the peptide to nonpeptide gap in topological similarity searches,” Journal of Chemical Information and Computer Sciences, vol. 41, no. 5, pp. 1395–1406

[21] J.-F. Truchon and C. I. Bayly, (2007) “Evaluating virtual screening methods: Good and bad metrics for the early recognition problem,” Journal of Chemical Information and Modeling, vol. 47, no. 2, pp. 488–508

[22] N. Huang, B. K. Shoichet, and J. J. Irwin, (2006) “Benchmarking sets for molecular docking,” Journal of Medicinal Chemistry, vol. 49, no. 23, pp. 6789–6801,

[23] K. Heikamp and J. Bajorath, (2003) “Large-scale similarity search profiling of chembl compound data sets,” Journal of Chemical Information and Modeling, vol. 51, no. 8, pp. 1831–1839

[24] S. G. Rohrer and K. Baumann, (2009) “Maximum unbiased validation (muv) data sets for virtual screening based on pubchem bioactivity data,” Journal of Chemical Information and Modeling, vol. 49, no. 2, pp. 169–184,

[25] D. Hoksza and P. Škoda, (2014) “2d pharmacophore query generation,” in Bioinformatics Research and Applications, ser. Lecture Notes in Computer Science, M. Basu, Y. Pan, and J. Wang, Eds. Springer International Publishing, vol. 8492, pp. 289–300

[26] C. W. Yap (2011) “Padel-descriptor: An open source software to calculate molecular descriptors and fingerprints,” Journal of Computational Chemistry, vol. 32, p. 1466-1477

AUTHORS

Petr Skoda received his master degree in 2014 from Faculty of Mathematics and Physics at the Charles University in Prague. Since 2014 he is Ph.D. studentunder the supervision of David Hoksza. His main research interest is chemoinformatics.

David Hoksza received his Ph.D. in 2010 from the Dept. of Software Engineering, Charles University inPrague. Since 2011 he is an associated professor of software engineering in the department of Software Engineering at the CharlesUniversity in Prague. His research interests include structural bioinformatics, chemoinformatics, data engineering and similarity searching.