Pharmacophore Alignment Search Tool: Influence of Canonical Atom Labeling on Similarity Searching

VOLKER HÄHNKE,[1,2] MATTHIAS RUPP,[3] MIREILLE KRIER,[4] FRIEDRICH RIPPMANN,[4] GISBERT SCHNEIDER[2]
[1]*Chair for Chem- and Bioinformatics, Johann Wolfgang-Goethe University, Frankfurt am Main 60323, Germany*
[2]*Eidgenössische Technische Hochschule (ETH), Institute of Pharmaceutical Sciences, Zürich 8093, Switzerland*
[3]*Helmholtz Zentrum München, German Research Center for Environmental Health, Neuherberg 85764, Germany*
[4]*Merck KGaA, Merck Serono Research, Bio- and Chemoinformatics, Darmstadt 64293, Germany*

Received 10 February 2010; Revised 7 April 2010; Accepted 7 April 2010
DOI 10.1002/jcc.21574
Published online 1 June 2010 in Wiley Online Library (wileyonlinelibrary.com).

Abstract: Previously, (Hähnke et al., J Comput Chem 2009, 30, 761) we presented the Pharmacophore Alignment Search Tool (PhAST), a ligand-based virtual screening technique representing molecules as strings coding pharmacophoric features and comparing them by global pairwise sequence alignment. To guarantee unambiguity during the reduction of two-dimensional molecular graphs to one-dimensional strings, PhAST employs a graph canonization step. Here, we present the results of the comparison of 11 different algorithms for graph canonization with respect to their impact on virtual screening. Retrospective screenings of a drug-like data set were evaluated using the BEDROC metric, which yielded averaged values between 0.4 and 0.14 for the best-performing and worst-performing canonization technique. We compared five scoring schemes for the alignments and found preferred combinations of canonization algorithms and scoring functions. Finally, we introduce a performance index that helps prioritize canonization approaches without the need for extensive retrospective evaluation.

 J Comput Chem 31: 2810–2826, 2010

Key words: global alignment; line notation; molecular graph; similarity; virtual screening

Introduction

The Pharmacophore Alignment Search Tool (PhAST) is a string-based approach to virtual screening.[1] It reduces each molecule to an unambiguous linear representation describing its pharmacophore—called "PhAST-sequence"—in three steps: (i) each nonhydrogen atom in the structure graph is replaced by a potential pharmacophoric point symbol, hydrogen atoms are removed; (ii) vertices of this pharmacophore graph are canonically labeled by the algorithm of Weininger et al.[2]; and (iii) vertex symbols are concatenated into a string in increasing order of their canonical labels. For virtual screening, both the screening compound collection ("library") and the query molecules are converted, and the resulting PhAST-sequences are compared using pairwise global sequence alignment.[3] As a result, molecular similarity values are computed from the alignment, which can be used for the retrieval of pharmacophorically similar molecules from a compound database.

Here, we present some modifications to the original method. To speed up the alignment process, we exchanged the Needleman Wunsch algorithm[3] with an algorithm proposed by Durbin et al.[4] that has the same asymptotic runtime complexity but a lower constant, i.e., it runs faster in practice. We compared the retrospective virtual screening performance of both algorithms using BEDROC scores[5] with Pearson's ρ[6] and complete ranked result lists with Kendalls's τ as rank-correlation coefficient.[7] We further investigated alternatives for the evaluation of sequence alignments, namely the alignment score and the normalized alignment score. This was motivated by a previous comparison of these methods for determination of homology of protein sequences.[8] There, sequence identity was inferior to the (normalized) alignment score, and both performed worse than significance-based evaluation methods like the *E*-value measure.[9]

The focus of the present work lies in comparison with different canonical labeling algorithms in step (ii) (*vide supra*) of

Additional Supporting Information may be found in the online version of this article.

Correspondence to: G. Schneider; e-mail: gisbert.schneider@pharma.ethz.ch

PhAST. We demonstrate the necessity of this canonization step, which improves performance over random symbol ordering. In addition to the algorithms proposed by Weininger et al.[2] and Jochum and Gasteiger,[10] we implemented a third canonization algorithm for molecular graphs suggested by Prabhakar and Balasubramanian.[11] All three algorithms were tested in their original version and in a modified version that excludes some of the original vertex prioritization rules. In addition, we used several dimensionality reduction methods, namely linear principal component analysis[12] (PCA) and the nonlinear methods Laplacian Eigenmaps,[13] Isomap,[14] and minimum volume embedding[15] (MVE) to reduce two-dimensional graphs to one-dimensional representations.

We compared the different canonization methods by retrospective virtual screening of a collection of drugs and lead compounds (collection of bioactive reference analogues (COBRA)).[16] For statistical evaluation, we used BEDROC[5] scores (with $\alpha = 20$ as suggested as default value for evaluation[5]), the permutation test proposed by Zhao et al.,[17] and the Kolmogorov–Smirnov test.[18] Finally, we investigated properties of the canonization algorithms related to their impact on virtual screening performance. To this end, we quantified the extent to which neighborhoods of graph vertices are preserved by the algorithms. Because the small structural modifications of molecules should result in similar PhAST-sequences, we investigated the effect of adding small fragments to the original molecules.

Methods

Definitions of pharmacophoric points used in step (i) of PhAST are shown in Tables 1 and 2. We use the original PhAST version[1] as baseline with the only modification being a gap open penalty of three instead of five. The reason for this was a change in preprocessing of compounds ["washing" with MOE (Molecular Operating Environment, version 2010.06, Chemical Computing Group, Montreal, Canada) instead of using fully protonated structures], resulting in best retrospective performance for the new gap penalty. All retrospective screens were performed using the COBRA library[16] (version 6.1, 8311 bioactive compounds; see Table 3 for a list of the selected targets).

Sequence alignment is used in bioinformatics to decide how related two sequences (deoxyribonucleic acid (DNA), ribonucleic acid (RNA), and amino acid sequences) are. To create the alignment of two sequences $X = x_1x_2...x_n$ and $Y = y_1y_2...y_m$, their symbols are matched. The symbol order is retained, and gaps may be inserted to improve the matching (insertion of paired gaps is forbidden). Three cases exist: (i) x_i is aligned to y_j and $x_i = y_j$ (match), (ii) x_i is aligned to y_j and $x_i \neq y_j$ (mismatch), (iii) x_i is aligned to a gap in Y, or y_j is aligned to a gap in X. In protein sequence alignment, matches represent conserved residues, mismatches may arise from mutations, and gaps from insertions or deletions in an assumed evolutionary process of the sequences. Consequently, matches are rewarded with a positive score, mismatches are either rewarded with a positive score or penalized with a negative score (depending on the particular scoring scheme), and gaps are penalized with a negative score. The optimal alignment is the one with the highest overall score (summed over the whole alignment). It can be computed using dynamic programing.[3]

Table 1. Potential Pharmacophoric Points Used in PhAST.

Possible interactions	PPP symbol
Hydrogen bond acceptor	A
Charge positive	P
Charge negative	N
Lipophilic	L
Aromatic	R
Hydrogen bond acceptor, hydrogen bond donor	E
Hydrogen bond acceptor, charge positive	Q
Hydrogen bond acceptor, hydrogen bond donor, charge positive	U
No possible interactions	O

Table 2. Pharmacophoric Point Definitions in Terms of Molecular Query Language (MQL)[19] Queries.

MQL query	PPP symbols
C	R
N	R
*[charge < 0]	N
*[charge > 0]	P
C(=O)−O−H	O;N;E
P(=O)−O−H	O;N;E
S(=O)−O−H	O;N;E
N[allHydrogens=0&totalConnections=3]	Q
N[allHydrogens=1&totalConnections=3](−C′)−C′	U
N[allHydrogens=2&totalConnections=3]−C′	U
N[allHydrogens=1&totalConnections=2]=C′	E
N[allHydrogens=0&totalConnections=2](=C′)−C′	A
O−H	E
C=O	O;A
C[!bound(∼ N)&!bound(∼O)]-*′[Cl|F|Cl|Br|I|S]	L
Cl	L
Br	L
I	L
S[!bound(∼N)bound(∼O)]∼*′[Cl|H]	L

Symbols are assigned to atoms in the query from left to bottom; queries are used from top to bottom.

Table 3. Targets, Taken From the COBRA Library (Version 6.1, $n = 8{,}311$).

Target	Abbreviation	No. of actives
Angiotensine-converting enzyme	ACE	34
Cyclooxygenase 2	COX2	136
Dihydrofolate-reductase	DHFR	64
Factor Xa	FXA	228
Peroxisome-proliferator-activated receptor type γ	PPARγ	44
Thrombin	THR	183
Total		689

Previously, we used an implementation of the Needleman Wunsch algorithm[3] for sequence alignment by PhAST.[1] The algorithm was adapted to the affine gap penalty model with a fixed gap open and gap extension penalty.[4] This version runs in $O(nm)$ instead of the original $O(nm(n+m))$ for any gap penalty model. To further speed up virtual screenings, we implemented the global pairwise sequence alignment algorithm conceived by Durbin et al.,[4] hereafter referred to as finite-state-machine ("FSM") algorithm. It has the same $O(nm)$ runtime but runs noticeably faster in practice. In some cases, the two algorithms alignments are not identical, because the FSM algorithm prohibits the insertion of a gap in Y directly following a gap in X. This simplification reduces computational cost and causes the speedup, but it does not change the asymptotic runtime.

To assess whether this influences results, we conducted the same retrospective virtual screenings twice, once for each algorithm. For each target (Table 3), each active was used as query, resulting in 689 ranked lists for each of the two alignment algorithms. For each list, the BEDROC score[5] was calculated (with $\alpha = 20$). The correlation between the two sets of BEDROC scores was determined using Pearson's ρ.[6] Statistical significance of the observed correlation was estimated from the p-value of a t test under the null hypothesis that the correlation equals zero.

The BEDROC score is based on ranks and thus invariant under permutations of the actives' ranks. To investigate differences in the complete ranked lists produced by both algorithms, we compared the two ranked lists of each query with Kendall's τ[7] as rank-correlation coefficient. Because ties can occur, we used τ_b, which corrects for this scenario. The significance of the observed rank correlation was calculated as the p-value of a z test under the null hypothesis that the correlation equals zero.

The focus of this work is on the influence of the canonization step on PhAST performance. We compared canonization methods as follows: With each method and for each target, each active was used as query in a retrospective virtual screening, resulting in 689 ranked lists. For each ranked list, the BEDROC score was calculated ($\alpha = 20$). The mean BEDROC score was used as overall performance index. For each canonization method and target, gap open and extension penalties were optimized using a grid search (starting from gap open penalty = 2 and gap extension penalty = 1, each combination with gap extension penalty lower than gap open penalty was tested, resulting in 190 penalty combinations), as it is hard to choose them by intuition.[20] Gap penalties greater than 20 seem unreasonable as they exceed the highest mismatch penalties.

To prove that the canonization step is essential, we compared baseline PhAST (Weininger canonization) against PhAST with random labeling in step (ii) of the algorithm. To avoid bias (default gap penalties are optimized for Weininger algorithm), we used the same simple scoring scheme for both labeling methods: Matches are rewarded with +1, mismatches are penalized with −1, and both gap penalties are 1. For random labeling, we generated 100 pairs of PhAST-sequences for each pair of molecules and used the average score as final similarity value.

To assess whether two different versions of PhAST have significantly different performance, we used the permutation test proposed by Zhao et al.[17] It has the null hypothesis that virtual screening method P performs significantly better than method Q. Assuming p and q are rank lists of actives resulting from the virtual screening methods, the null hypothesis requires that BEDROC(p) > BEDROC(q). As each active has two ranks, one in p and one in q, new rank lists p^* and q^* can be created by swapping its rank in p with its rank in q for each active with probability 1/2. This is repeated 10,000 times and the frequency of the event that BEDROC(p)–BEDROC(q) is less than BEDROC(p^*)–BEDROC(q^*) is recorded. The frequency of this event is the type I error rate for the null hypothesis. In addition, we used a Kolmogorov–Smirnov test[18] for the same purpose. Both methods were used to assess the significance of the difference between using the Weininger algorithm for graph canonization and using random concatenation of symbols and to assess the improvement from baseline PhAST to the best combination of algorithms identified in this work. In both cases, calculations were based on the 689 BEDROC scores resulting from each version of PhAST.

Canonization Algorithms

The atom-typing step in PhAST yields a graph of potential pharmacophoric points that has the same topology as the molecular graph with suppressed hydrogen atoms. Each vertex is colored with a symbol (Table 1) that represents a potential pharmacophoric point. Edges represent covalent bonds. Canonization is the labeling of the vertices with the natural numbers 1,2,3... In a previous study,[1] we compared the canonization algorithms of Weininger et al.[2] and Jochum and Gasteiger[10]; we reevaluate them, here, because of changes in molecule preprocessing. In contrast to these two algorithms, the one by Prabhakar and Balasubramanian[11] is based on paths, a property thought to be beneficial for PhAST. We modified all algorithms by using pharmacophoric points as prioritization criterion instead of the element number.

Jochum–Gasteiger Method

The canonical labels created by the Jochum and Gasteiger algorithm[10] are in most cases identical to those obtained by the Morgan[21] algorithm.[10] The first step is the separation of all vertices into two sets—terminal vertices (vertices with exactly one single bond) and core vertices (all others). All core vertices with the same buriedness are members of the same equivalence class. The algorithm divides the vertices of each class further using a set of prioritization criteria, until only one atom remains that gets the next label, starting from the vertices in the most buried class. Prioritization criteria are (i) priority of the potential pharmacophoric point (atom number in the original application) and (ii) number of free electrons. In both cases, the vertex with the highest value has priority. The next criteria involve the environment of the vertices organized in spheres around each vertex. The vertex with the highest of these values in his neighborhood gets priority: (iii) number of vertices, (iv) priority of potential pharmacophoric points, (v) number of free electrons, (vi) number of bonds in the next sphere, (vii) bond order of these bonds, (viii) neighborhood to an already labeled vertex, and (ix) bond

order to the vertex in (viii). If more than one vertex remains after (ix), all of them are marked as indistinguishable and the remaining vertices have priority over them. After all distinguishable vertices are labeled, (viii) is used to label the undistinguishable vertices. After all core vertices are prioritized, terminal vertices are prioritized by criteria (i) and (viii).

Weininger Method

The canonization algorithm by Weininger et al. was proposed as part of canonical simplified molecular input line entry system (SMILES) generation.[2] Its idea is to assign vertices to topological symmetry classes. It first assigns a property vector to each vertex that consists of different atomic invariants mainly based on the original molecular graph: (i) number of connected vertices, (ii) number of connected nonhydrogen atoms, (iii) priority of the pharmacophoric point (atom number in the original version), (iv) sign of charge, (v) absolute charge, and (vi) number of connected hydrogen atoms. Vertices with identical vectors form an equivalence class, and all vertices are sorted ascending by this vector. For each vertex, its extended connectivity is calculated as follows: Beginning with the equivalence class with the lowest index, the vertices in each class are assigned the same prime number, starting with 2. For each vertex in the graph, the product of the primes of its neighbors is calculated. These product values define new equivalence classes on the vertices. Each equivalence class, in order of product values, is assigned an index, starting from 1. This process is repeated until the number of equivalence classes does not change in a step. If, after extended connectivity calculation, an equivalence class contains two or more vertices, these ties are broken by an additional step: the index of each equivalence class is doubled, and one vertex from the equivalence class with the lowest index is randomly chosen to form an own equivalence class with the index of its original equivalence class lowered by 1. After that, all equivalence classes are renumbered starting from 1. These two steps (computing the extended connectivity and breaking ties) are alternated until the number of equivalence classes equals the number of vertices in the graph.

Prabhakar–Balasubramanian Method

The canonization algorithm by Prabhakar and Balasubramanian[11] uses more graph-based prioritization rules than the other two algorithms and progresses along paths through the graph. First, the number of incident bonds with respect to bond order is determined for each vertex (c_n). As with the Jochum and Gasteiger algorithm, vertices are divided into two sets, terminal vertices ($c_n = 1$) and core vertices ($c_n > 1$). Labeling starts with the core atoms. Using the following prioritization rules, they are divided into smaller subsets, until only one atom (which will get the next canonical label) remains: (i) number of incident bonds, (ii) number of incident bonds with respect to bond order, and (iii) pharmacophoric point priority (atom number in the original version of the algorithm). In these cases, the vertex with the highest value has priority. If more than one atom with highest priority remains, copies of the original graph are created, called "fragments." If there are n vertices left for prioritization, $n-1$ copies of the original graph are created for each vertex v. In each copy, the first bond in the shortest path between v and one of the other competing vertices is deleted. Only the part of the copy that includes v is retained. The remaining prioritization rules are applied to these fragments; a vertex has the highest priority, if one of his fragments has a higher priority than the fragments created for all other vertices: (iv) the length of the path starting in the competing vertex and following the highest precomputed c_n values, until it reaches a vertex already visited or labeled, (v) number of loops, (vi) length of the longest path in the fragment, (vii) number of pharmacophoric points not lipophilic, aromatic or no interaction, (viii) summed symbol priorities of vertices in the fragment, (ix) averaged distances between all vertices not lipophilic, aromatic, or no interaction in the fragment. In all cases except the last one, the fragment with the highest value has priority. If there remains more than one vertex and there is no already labeled vertex, one of them is chosen arbitrarily and has priority over all other vertices. If there is already at least one labeled vertex, (x) the label of the connected vertex is used. These rules are used in a depth-first search. All neighbors of the last labeled vertex are the potential candidates for the next label. If this search reaches an end point, all vertices adjacent to an already labeled vertex are candidates for the next label. After all core vertices are labeled, terminal vertices are labeled according to criteria (ii), (iii), and the label of the neighboring core atom.

Irrespective of the canonization method used, the PhAST-sequences created from two identical graphs of pharmacophoric points are identical. If a pharmacophoric point is changed, but the topology remains the same, the relative order of symbols in the PhAST-sequence should remain unchanged as well. Yet all three algorithms use pharmacophoric point priority as a prioritization criterion. Consequently, the changes in a PhAST-sequence because of exchange of a single vertex symbol can be more severe than intended. To attenuate this, each canonization algorithm was tested in a modified version:

- In the Jochum and Gasteiger algorithm, clipping of terminal atoms was omitted, and the criteria symbol priority and number of free electrons were eliminated.
- For the Weininger algorithm, the creation of the initial prioritization vector was changed: the priority of the pharmacophoric point, the total number of neighbors, and the number of neighboring hydrogen vertices were omitted and both charge criteria.
- For the Prabhakar algorithm, the initial clipping of terminal vertices was omitted. The priority of the pharmacophoric point, the number of pharmacophoric points not lipophilic, aromatic or no interaction in a fragment and averaged distances between all vertices not lipophilic, aromatic or no interaction in a fragment were removed.

Canonization by Dimensionality Reduction

An alternative approach to canonization that to our knowledge has not been used before for canonization and does not suffer from the mentioned drawbacks is the use of dimensionality reduction algorithms. We implemented four such methods.

Principal Component Analysis

Principal component analysis[12] is a linear dimensionality reduction method often used to visualize high-dimensional data. We used PCA to calculate one-dimensional coordinates from two-dimensional graph layouts generated by the 2D depiction algorithm of MOE (version 2010.06, Chemical Computing Group, Montreal, Canada). Therefore, the coordinates of the vertices in each graph were mean-centered, and the covariance matrix between the position vectors of all vertices calculated. The computation of the eigenvectors and eigenvalues of the covariance matrix gives the loading vectors that are used for the computation of the new coordinates of the vertices. To get the one-dimensional coordinate for each vertex, the dot product between its original position vector and the loading vector with the highest absolute eigenvalue was calculated. Beginning with the vertex with lowest one-dimensional coordinate, we assigned labels in ascending order. For identical one-dimensional coordinates, we used their coordinate in the second dimension of the principal component space as prioritization criterion. In all cases, the vertex with the lowest coordinate had the highest priority.

PCA finds a low-dimensional embedding of data points that best preserves their variance. However, PCA fails when a data set contains nonlinear structures. Nonlinear approaches that overcome this problem start with the assignment of vertex neighborhoods by using a connectivity algorithm like k-nearest neighbors,[22] b-matching[23] (each vertex gets assigned exactly b neighbors), or ε-balls (a vertex is connected to all vertices within distance ε), resulting in a neighborhood graph with edges between neighboring vertices. We directly used the topology of the molecular graph instead of connectivity algorithms. In the embedding, these methods aim at preserving the pairwise distances between neighbors.

Laplacian Eigenmaps

Laplacian Eigenmaps[13] start by calculating three matrices from the neighborhood graph: the weight matrix W with [eq. (1)]

$$W_{ij} = \begin{cases} 1 \text{ if } i \text{ and } j \text{ are connected} \\ 0 \text{ else} \end{cases}, \quad (1)$$

the degree weight matrix D with the column sums of W as entries [eq. (2)],

$$D_{ii} = \sum_j W_{ij}, \quad (2)$$

and the positive semidefinite Laplacian matrix L [eq. (3)] with

$$L = D - W. \quad (3)$$

Then, the eigenvalues and eigenvectors of the generalized eigenvector problem [eq. (4)] are calculated.

$$Lf = \lambda Df \quad (4)$$

Eigenvectors (f) are sorted according to their eigenvalues (λ) in ascending order. The first eigenvector with $\lambda = 0$ is omitted. The next d eigenvectors are used for embedding. In our case, the second eigenvector contains the coordinates for the one-dimensional embedding.

Isomap

The Isomap algorithm by Tenenbaum et al.[14] uses the neighborhood graph to estimate geodesic distances between the vertices. A matrix D of shortest distances between all vertices is computed, e.g., using the Floyd–Warshall algorithm.[24,25] Using D, the matrix $\tau(D)$ is calculated as [eq. (5)]:

$$\tau(D) = -\tfrac{1}{2} * HSH \quad (5)$$

where S is the matrix of squared distances [eq. (6)]

$$S_{ij} = D_{ij}^2 \quad (6)$$

and H the centering matrix [eq. (7)]:

$$H_{ij} = \delta_{ij} - \frac{1}{n} \quad (7)$$

with δ_{ij} the Kronecker delta and n the number of vertices. The eigenvectors and eigenvalues of $\tau(D)$ are computed. To embed in d dimensions, the first d eigenvectors sorted according to their eigenvalues in decreasing order are used. If λ_p is the pth eigenvalue of $\tau(D)$ and $v^i{}_p$ is the ith component of the pth eigenvector, then the pth component of the d-dimensional coordinate vector of a vertex is equal to $\sqrt{\lambda_p} v^i_p$.

Minimum Volume Embedding

The two previous methods lose all information contained in the eigenvectors that are not used for the embedding. None of them aims at minimizing the amount of information lost this way. MVE[15] preserves as much information as possible in the d dimensions used for embedding. This is achieved by an iterative process based on semi-definite programing (SDP). First, an affinity matrix A is calculated for the vertices using a kernel function k. A is positive semidefinite and must be centered. This matrix is used in the neighborhood definition process (instead of given vertex coordinates) to obtain a binary connectivity matrix C. A third matrix K is set equal to A. The following procedure is repeated until convergence: (i) Calculate the eigenvectors f_i and eigenvalues λ_i of K, sort the f_i descending to their corresponding λ_i. (ii) Calculate the matrix B using eq. (8),

$$B = -\sum_{i=1}^{d} f_i f_i^T + \sum_{i=d+1}^{N} f_i f_i^T \quad (8)$$

(iii) use SDP to *s*olve eq. (9)

$$K = \arg\min_{K \in \mathrm{K}} \mathrm{tr}(KB) \quad (9)$$

under constraints K defined by Shaw and Jebara,[15] tr denotes the matrix trace (sum of the diagonal elements).

After convergence, kernel PCA[26] is performed with K to get the d eigenvectors used for embedding. MVE works with any positive semidefinite kernel k. We used MVE with two different kernel functions. The first one is a diffusion kernel.[27] For each pair of vertices v_i and v_j, it returns the probability that a random walk starting in v_i will be in v_j after an infinite number of steps, with only a low probability of leaving the current vertex in each step. The kernel matrix can be calculated according to eq. (10).[28]

$$K = e^{(-\beta L)} \tag{10}$$

where L is the Laplacian matrix introduced in (3) and β is the diffusion parameter. If β equals 0, no diffusion is allowed and K equals the unit matrix. K is computed by matrix exponentiation, which is different from componentwise exponentiation. We used 12 values for β to determine its influence on PhAST: 0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, and 10. The second kernel function calculates inner products from the Euclidean coordinates of the vertices. It is defined as given in eq. (11).

$$k(x,y) = \frac{1}{2} * (\|x\|^2 + \|y\|^2 - \|x-y\|^2). \tag{11}$$

To obtain Euclidean coordinates for each vertex, as for PCA, we used the 2D depiction algorithm of MOE. This particular kernel is parameter free.

When using dimensionality reduction methods, slight modifications of a molecule can switch the direction of the axis of the one-dimensional coordinate system. We addressed this issue by repeating every sequence comparison during the virtual screening with one of the sequences inverted, if dimensionality reduction was involved. We used the higher of the two resulting values as similarity measure.

Evaluation of the Alignment

In addition to canonization methods, we analyzed the effect of different alignment evaluation methods. The alignment of nucleic or amino acid sequences is a traditional field in bioinformatics and well analyzed. It was shown that there are methods to determine peptide sequence homology from pairwise alignments that yield better results than the sequence identity used in PhAST.[8] These methods are alignment scores and significance estimations of alignments. We evaluated two variants of sequence identity and three variants of alignment scores. There is no significant overhead compared with sequence identity calculation.

Sequence Identity

The original PhAST used the percent identity (PID_1) between two sequences X and Y [eq. (12)],

$$\mathrm{PID}_1 = \frac{M(A(X,Y))}{L(A(X,Y))} \tag{12}$$

with $A(X,Y)$ the alignment of X and Y, $M(A(X,Y))$ the number of matches in the alignment, and $L(A(X,Y))$ the length of the alignment (including all gaps). Comparing two sequences of molecules active on the same target but of different size might result in a low PID_1, because the global alignment has to extend the shorter sequence to the length of the longer one with gaps. To counteract this effect, we correct for the size of the sequences: We first calculate the maximum reachable PID_1 of two sequences by inserting the maximum number of matches (length of the shorter sequence) and the minimum length of the alignment (length of the longer sequence) into eq. (12). We then normalize PID_1 according to eq. (13):

$$\mathrm{PID}_2 = \frac{\text{actual PID}_1}{\max \text{PID}_1} \tag{13}$$

Alignment Score

Besides sequence identity, we investigated alignment scores as evaluation measures. The raw alignment score S_1 is the sum of matches, mismatches, and gap penalties. Alignments of long sequences tend toward higher scores, so S_1 depends on sequence similarity and length. We normalize S_1 by dividing through the length of the alignment, yielding S_2. The resulting score measures the average contribution of each event in the alignment (match, mismatch, or gap). S_1 can also be normalized by dividing through the length of the shortest original sequence, yielding S_3, a measure of the maximum averaged contribution of each symbol in a sequence. All alignment score methods are measures of the similarity between aligned sequences, but are no longer bounded by 0 and 1.

Alignment evaluation methods involving alignment length or the number of occurrences of an event suffer from the drawback that there can be more than one optimal alignment of two sequences, which one is found depends on implementation details. The alignment of the sequence pair (X,Y) can, therefore, differ from that of (Y,X) in length, number of matches, number of mismatches, number of gaps, and gap length. To ensure the symmetry of our method, i.e., identical scores for $A(X,Y)$ and $A(Y,X)$, we modified the affected evaluation methods. In case of PID_1, we compute $A(X,Y)$ and $A(Y,X)$, and use the average PID_1 as final evaluation measure. This correction is used in the calculation of actual PID_1 for PID_2 as well. In case of S_2, we align both sequence pairs and use the averaged alignment length for normalization.

In total, we compared 11 graph canonization methods combined with five alignment evaluation methods and 190 gap penalty combinations by conducting 689 virtual screenings for each combination and averaging the resulting BEDROC scores (α = 20). To assess whether our modifications lead to significant improvements of PhAST, we used the permutation test proposed by Zhao et al.[17] and a Kolmogorov–Smirnov test.[18]

Canonization Analysis

To further quantify the differences between canonization methods, we determined how well the neighborhood relations in the original pharmacophoric point graph are retained and represented in the resulting PhAST sequences. We did this by counting how often the vertices, which are neighbors in the graph, are

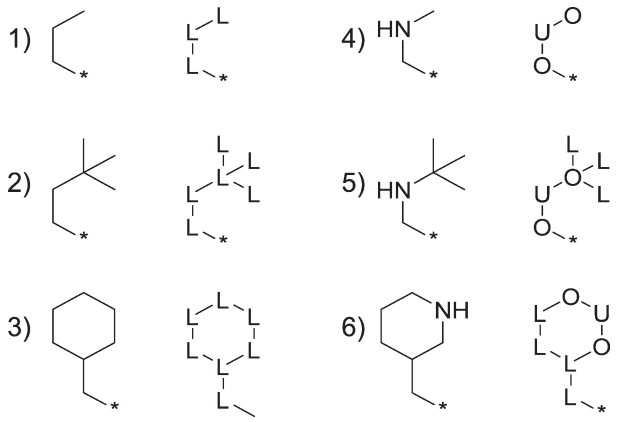

Figure 1. Fragments for the comparison of modified molecules. * Denotes the attachment point. Shown are molecular fragments (left) and corresponding pharmacophoric point graphs (right).

neighbors in the PhAST-sequence. This was done for every molecule in the COBRA library of reference compounds.[16] For all graph neighbors that are not neighbored in the one-dimensional representation, we counted how many vertices were inserted between them, resulting in a histogram of these distances. As each pair of neighboring vertices was viewed twice, once from each vertex as origin, the resulting counts were divided by 2.

For use with PhAST, canonization algorithms should be robust against small changes in molecular structure, i.e., similar molecules should have similar PhAST-sequence. This in turn means that neighbors in the PhAST-sequences of a molecule should remain neighbors even if the molecule is slightly modified. To test this, the compounds in the COBRA library were modified by attaching small fragments. After conversion to PhAST-sequence, we counted (i) neighboring vertices in the original PhAST-sequence that are neighbors in the modified PhAST-sequence, (ia) with their relative orientation as in the original PhAST-sequence, (ib) with their relative orientation changed; (ii) neighboring vertices in the original PhAST-sequence that are not neighbors in the modified PhAST-sequence, (iia) with their relative orientation as in the original PhAST-sequence, (iib) with their relative orientation changed. If two vertices of the same type that are neighbors in the original PhAST-sequence are still neighbors in the modified PhAST-sequence, but changed their relative orientation, this event is counted as (ia) because the change of positions has no effect on the PhAST-sequence due to identical types. Cases (ib) and (iib) present a problem for global sequence alignment as a string comparison method. As the relative position of symbols cannot be changed, the only two operations that can be reconstructed in the alignment process are mutations (a symbol changed to another one) and insertion/deletions (compensated by gaps). A transversion, i.e., the swapping of two different symbols, cannot be properly modeled by only one event.

Figure 1 presents the six used fragments. Three fragments consist only of carbon atoms that are typed as L in the atom typing step. The other three are topologically identical to the first three fragments, with one carbon changed to nitrogen typed as U. This way, the algorithms are confronted with topological modifications and changes in vertex priorities. Fragment attachment points should not change the atom typing. We used carbon atoms that were typed as L, R, or O in the original molecule and were connected to at least two hydrogen atoms. One of the hydrogen atoms was replaced by the first atom of the fragment. Molecules (1612/8311) from COBRA were omitted, because they had less than 10 possible attachment points. Each single fragment attachment was repeated five times at random positions. In addition, each possible combination between fragments (resulting in 21 unique pairs) was used five times as well, again at random positions. In total, each molecule was compared to 135 variants of itself.

All modifications were undertaken in a single preprocessing of the COBRA compound collection to ensure that all algorithms are compared with the same modified molecules. The resulting molecular graphs were depicted using MOE, because one variant of MVE depends on Euclidean distances. In the case of dimensionality reduction methods, again both possible canonization results were used in the analysis, and we used the one with a higher preservation of neighborhood relationships. This is justified, because for these methods, both orientations of the PhAST-sequence are used during a virtual screen.

All programing was done using the Java Programing Language (version 6). Eigen decompositions were done with the java linear algebra package (JLAPACK) library.[29] SDP problems were solved with the c semi-definite programming library (CSDP) solver.[30] Productive runs and calculations were performed on a Linux cluster with 40 advanced micro devices (AMD) Opteron 8214 processors and 320 gigabyte (GB) random access memory (RAM).

Results and Discussion

Choice of Alignment Algorithm

To determine whether the exchange of the alignment algorithm of Needleman and Wunsch by the faster FSM algorithm affected the performance of PhAST, we determined the correlation of BEDROC scores obtained from virtual screening with each active as reference ($n = 689$). The Pearson-correlation coefficient was 0.9996 with a p-value of below 10^{-1051}. To quantify the differences within the complete ranked lists, Kendall's τ was computed (see Fig. 2). The minimum correlation observed was 0.945, the maximum correlation observed was 0.998, and the average correlation observed was 0.984. The p-value for each τ was below $0.5 * erfc\left(\frac{82,779,141,588*\sqrt{2}}{109,242,709}\right)$. The FSM algorithm used only 40% of the runtime of the Needleman Wunsch algorithm. On the basis of the gain in computation speed and the high correlations, we decided to employ the FSM algorithm for all experiments in this study.

Necessity for Canonization

To verify the importance of the canonization step, we compared BEDROC values ($\alpha = 20$) of baseline PhAST and PhAST with random labeling (average of 100 random labeling procedures

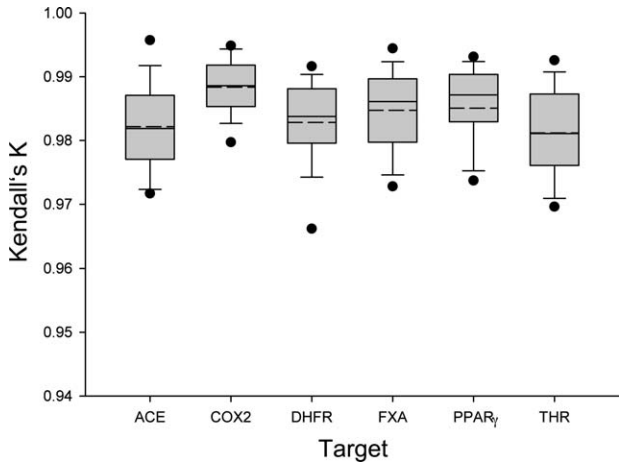

Figure 2. Box-whisker plots of rank correlation coefficients between the ranked lists obtained using the Needleman–Wunsch and the FSM algorithm for sequence alignment per target (n = 34, 136, 64, 228, 44, and 183). Shown are 5th/95th (points), 10th/90th (whiskers), 25th/75th (box borders) percentiles, median (solid line), and mean (dashed line).

per comparison; matches = +1, mismatches = −1, gap penalties = 1).

Figure 3 presents the distribution of BEDROC scores per target. There is almost no overlap, with the exception of cyclooxygenase 2 (COX2). The latter can be explained by the distribution of pharmacophoric points: As given in Table 4, COX2 ligands have 56% pharmacophoric points of type R. Random concatenations of pharmacophoric points of the same type do not change the resulting PhAST sequence (more symbols of the same type

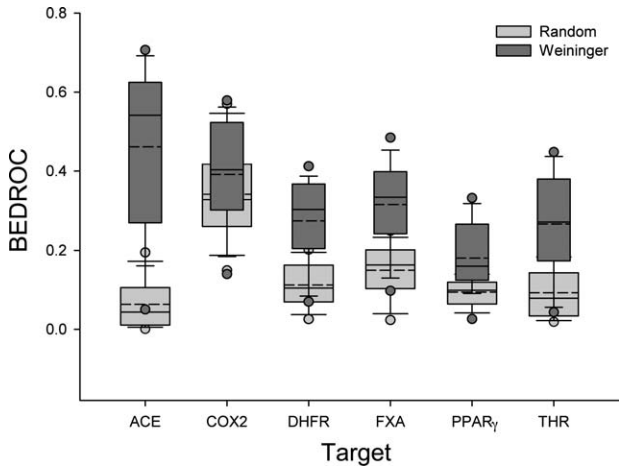

Figure 3. BEDROC (α = 20) scores of Weininger versus random canonization. A simple alignment scoring system with +1 (−1) for matches (mismatches) and gap open and extension penalty of 1. For random canonization, the mean similarity of 100 random sequences was used. Shown are 5th/95th (points), 10th/90th (whiskers), 25th/75th (box borders) percentiles, median (solid line), and mean (dashed line).

Table 4. Symbol Frequencies in the COBRA Library.

	ACE	COX2	FXA	PPARγ	DHFR	THR
Ø Symbols	27.12	24.61	34.34	28.57	27.34	35.95
σ	7.43	3.17	5.28	7.03	5.71	7.90
A	8.89	2.27	4.87	5.73	4.91	6.99
E	0	0.24	0.28	0.08	0.29	1.08
L	23.64	17.99	9.45	17.10	13.83	21.01
N	6.40	0.69	0.46	3.10	2.57	00.43
O	23.64	22.80	25.57	19.97	17.49	29.47
P	3.25	0.09	2.85	0.56	0.97	3.02
Q	1.74	0.18	2.92	0.88	0.74	2.60
R	31.45	55.48	48.43	51.31	49.94	29.21
U	0.98	0.27	5.17	1.27	9.26	6.19

"Ø Symbols" and "σ" indicate the average number of pharmacophoric points per molecule and the standard deviation.

only result in fewer possible sequences). The global sequence alignment matches regions of identical symbols between sequences, resulting in a high score. Subsequences of different lengths are compensated by gaps, lowering the score. PhAST-sequences of COX2 ligands have shortest average length and standard deviation, i.e., they are least affected by this effect because they are of comparable length. In agreement with this reasoning, the target with second largest overlap, peroxisome-proliferator-activated receptor type$_\gamma$ (PPARγ), is also second in type R symbols (51%) and of comparatively small size.

Table 5 presents the results of the Zhao permutation test. PhAST with Weininger canonization performed significantly better than random labeling in 91% of all screenings at a significance level of 0.05. In 5% of the screens, random labeling performed better. The latter cases are dominated by screenings on COX2. A Kolmogorov–Smirnov test showed that the difference between these two methods is significant with a p-value of 1.0835×10^{-78}. We conclude that the canonization step is necessary for PhAST.

Comparison of Canonization Methods

We compared 11 canonization algorithms, five alignment evaluation methods, and 190 gap penalty combinations for their effect on PhAST in a set of virtual screening experiments employing

Table 5. Permutation Test Results for Weininger Canonization Versus Random Canonization.

	No. of queries	Weininger	Random
ACE	34	100 (100)	0 (0)
COX2	136	70 (68)	22 (21)
DHFR	64	95 (95)	2 (2)
FXA	228	98 (98)	1 (1)
PPARγ	44	84 (84)	7 (5)
THR	183	97 (97)	0 (0)
Total	689	91 (91)	5 (5)

Shown are the percentages of cases where one contestant performs significantly better than the other at a significance level of 0.05 (0.01).

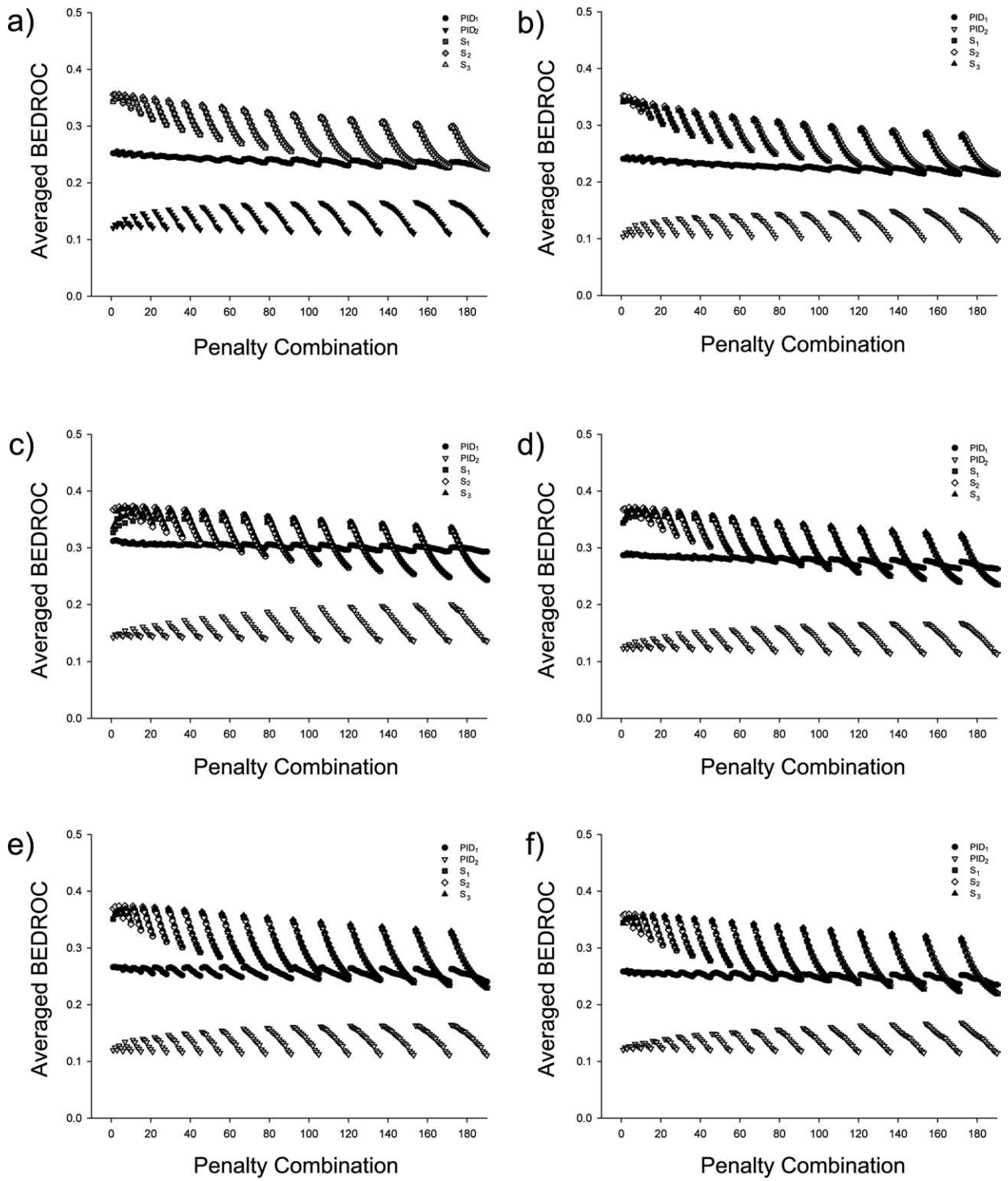

Figure 4. BEDROC ($\alpha = 20$) scores for combinations of alignment evaluation methods and gap penalties. (a) Jochum and Gasteiger algorithm, (b) modified Jochum and Gasteiger algorithm, (c) Weininger algorithm, (d) modified Weininger algorithm, (e) Prabhakar algorithm, (f) modified Prabhakar algorithm, (g) MVE with diffusion kernel and diffusion parameter 0.4, (h) MVE with Euclidean distance kernel, (i) Laplacian Eigenmaps, (j) Isomap, and (k) PCA.

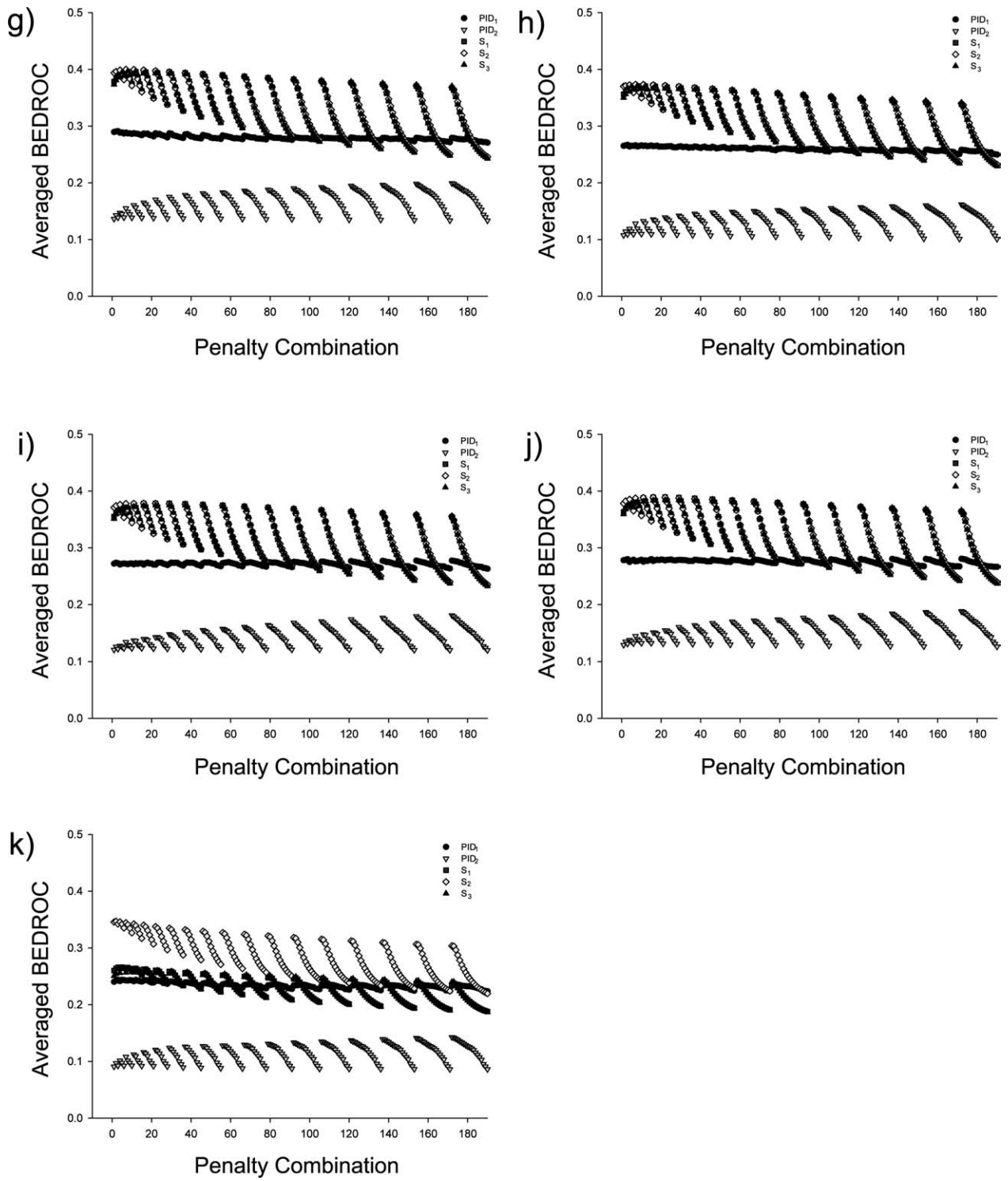

Figure 4. (*Continued*)

six targets (Table 3) from the COBRA library of bioactive compounds. For each combination of canonization, alignment, and gap penalties, each active molecule of each target was used as query in a virtual screening. Screening success was assessed by average (n = 689) BEDROC (α = 20) scores. For MVE with diffusion kernel, we compared 12 values of the diffusion parameter β. Only the best performing version with β = 0.4 that reaches the highest averaged BEDROC score is included in the

Table 6. BEDROC Scores ($\alpha = 20$).

	Scoring scheme				
	PID_1	PID_2	S_1	S_2	S_3
Jochum Gasteiger	0.26 (3, 2)	0.17 (20, 1)	0.35 (5, 1)	0.36 (4, 1)	0.35 (4, 2)
Jochum Gasteiger modified	0.24 (3, 2)	0.15 (20, 1)	0.35 (4, 1)	0.35 (2, 1)	0.35 (3, 2)
Isomap	0.28 (20, 1)	0.19 (20, 1)	0.39 (8, 1)	0.39 (7, 1)	0.38 (8, 1)
Laplacian Eigenmaps	0.28 (20, 1)	0.18 (20, 1)	0.38 (8, 1)	0.38 (7, 1)	0.38 (9, 1)
MVE diffusion kernel 0.4	0.29 (3, 2)	0.20 (20, 1)	0.39 (7, 1)	0.40 (5, 1)	0.39 (6, 2)
MVE Euclidean kernel	0.27 (3, 2)	0.16 (20, 1)	0.37 (7, 1)	0.37 (5, 1)	0.37 (5, 2)
PCA	0.24 (3, 2)	0.14 (20, 1)	0.27 (4, 2)	0.35 (3, 1)	0.26 (4, 3)
Prabhakar	0.27 (13, 2)	0.16 (20, 1)	0.37 (6, 1)	0.37 (5, 1)	0.37 (7, 1)
Prabhakar modified	0.26 (3, 2)	0.17 (20, 1)	0.26 (6, 1)	0.36 (4, 1)	0.36 (6, 1)
Weininger	0.31 (3, 1)	0.20 (20, 1)	0.36 (7, 2)	0.37 (6, 1)	0.37 (7, 2)
Weininger modified	0.29 (3, 2)	0.17 (29, 1)	0.36 (5, 2)	0.37 (4, 1)	0.37 (5, 2)

The best gap open/gap extension penalties are shown in parentheses.
Higher scores indicate better virtual screening performance.

following analysis. The results for the remaining settings of β are presented in Figure S1 (cf. Supporting Information). Figure 4 presents the outcome of the comparison.

For each canonization algorithm, alignment evaluation methods and gap penalty combinations have similar effects. For the algorithms of Weininger, Prabhakar, and Jochum–Gasteiger, the respective modified versions display a retrospective performance that is comparable to the original versions of the algorithms. Our modifications, therefore, neither worsened nor improved their performance. For MVE, the effect of different kernel functions is minor. The best parameterization of the diffusion kernel was slightly superior to the Euclidean distance kernel. Isomap and Laplacian Eigenmaps performed comparably but with slightly lower BEDROC scores. Among the dimensionality reduction methods, PCA performed worst. Considering the comparison of baseline PhAST with random labeling, we conclude that having a canonization method at all is more important than the particular method used.

PID_1 is the original alignment evaluation method, PID_2 penalizes differences in sequence lengths to a lesser extent than PID_1, and performs worse than PID_1 manifesting in lower averaged BEDROC scores (see Fig. 4). We introduced PID_2 to compensate for the difference in sequence lengths of actives of the same target. Although PID_2 yielded greater similarity values than PID_1, it also did so for sequences from different targets. This effect generates false positives, thereby, diminishing screening performance.

The evaluation methods S_1, S_2, and S_3 are all based on the alignment score with different normalization techniques. They perform similar and all of them appear to be superior to alignment evaluation methods based on sequence identity. We explain this observation by the improved weighting of matches and mismatches. Sequence identity is influenced only by the number of (mis)matches. The alignment score, however, is influenced by the exact type of match or mismatch depending on the symbols involved. For similar numbers of matches and mismatches, this enables a more differentiated evaluation of the alignment.

All combinations of canonization and alignment evaluation strongly depend on the gap penalties used. Retrospective results for one particular combination of canonization algorithm and alignment evaluation method show strong variation with different penalty combinations. For each gap open penalty, retrospective performance decreases with increasing gap extension penalty. This can be explained by the alignment process itself. The optimal alignment is the combination of positive scores for matches and negative scores for mismatches and gaps that is highest in sum. If gap penalties exceed mismatch scores, gaps will decrease the alignment score more than mismatches, thus increasing the number of mismatches. This results in alignments dominated by mismatches due to this effect and not because of the exchange of functional groups in the molecular graph. The resulting alignments do not reflect molecular similarity anymore and decrease virtual screening accuracy.

The averaged BEDROC score of the best performing gap penalty combination is given in Table 6 for each canonization algorithm and alignment evaluation method. The performance of baseline PhAST (Weininger canonization, gap open penalty = 3, gap extension penalty = 1, alignment evaluation PID_1) is 0.29 (bottom left in the table). The best performance was 0.40 [MVE canonization using the diffusion kernel ($\beta = 0.4$), gap open penalty = 5, gap extension penalty = 1, alignment evaluation S_2]. To see whether this improvement is significant, we performed the permutation test proposed by Zhao. Table 7 presents the results per target. The lowest fraction of significantly improved screenings is for angiotensine-converting enzyme (ACE) with ~24%. On average, the performance was significantly increased in 71% of all screening experiments. Baseline PhAST performs better only in 21%. In combination with the increased average BEDROC scores, we conclude that the improvement of our method is significant. This is supported by a Kolmogorov–Smirnov test producing a p-value of 1.37×10^{-21}.

Average BEDROC performance for the globally optimal gap penalty combination and gap penalties optimized for each target

Table 7. Permutation Test Results for MVE ($\beta = 0.4$) Canonization with S_2 Versus Weininger Canonization With PID_1.

	No. of queries	MVE 0.4 S_2 (5, 1)	Weininger PID_1 (3, 1)
ACE	34	24 (24)	59 (56)
COX2	136	56 (56)	40 (40)
DHFR	64	95 (95)	3 (3)
FXA	228	67 (64)	22 (21)
PPARγ	44	64 (61)	16 (14)
THR	183	90 (90)	6 (5)
Total	689	71 (70)	21 (20)

Shown are the percentages of cases where one contestant performs significantly better than the other at a significance level of 0.05 (0.01).

are very close (Table 8), with a maximum difference of 0.024 for COX2. We conclude that the global optimum (gap open penalty = 5, gap extension penalty = 1) provides reasonable default values for practical applications.

We varied two major components of PhAST, the canonization algorithm and the alignment evaluation method. Changing both improved baseline PhAST significantly. But which of these two variables is more important? The lowest score achieved with PID_1 (baseline PhAST) is 0.24 (modified Jochum–Gasteiger canonization), the highest score is 0.31 (Weininger canonization). With S_2 (best performance), even the lowest score of 0.35 (PCA canonization) lies above the highest score of PID_1. The highest score with S_2 for alignment evaluation is 0.40 (MVE canonization with diffusion kernel, $\beta = 0.4$). In 32 of 33 cases, the alignment evaluation methods based on the alignment score yield higher values with the same canonization algorithm than PID_1. For PID_2, this is true for all 33 cases. In Table 6, the mean coefficient of variation $(\sigma/\mu)^6$ of the columns is 0.09, whereas that of the rows is 0.27. Varying the alignment evaluation method, therefore, influences the performance three times stronger than varying the canonization method. We conclude that the choice of the alignment evaluation method is more important than the choice of the canonization algorithm.

Table 8. BEDROC Scores ($\alpha = 20$) Per Target for Global and Target-Optimal Gap Penalties.

	Average BEDROC with global optimum gap penalties (5, 1)	Average BEDROC with optimum gap penalties per target	
ACE	0.4034	0.4081	(2, 1)
COX2	0.4011	0.4251	(11, 2)
DHFR	0.5654	0.5704	(9, 1)
FXA	0.3563	0.3676	(2, 1)
PPARγ	0.2612	0.2612	(5, 1)
THR	0.4130	0.4165	(2, 1)

Global optimum gap penalties are gap open penalty 5 and gap extension penalty 1. The best performing gap penalties per target are shown in parentheses. Higher scores indicate better virtual screenings performance.

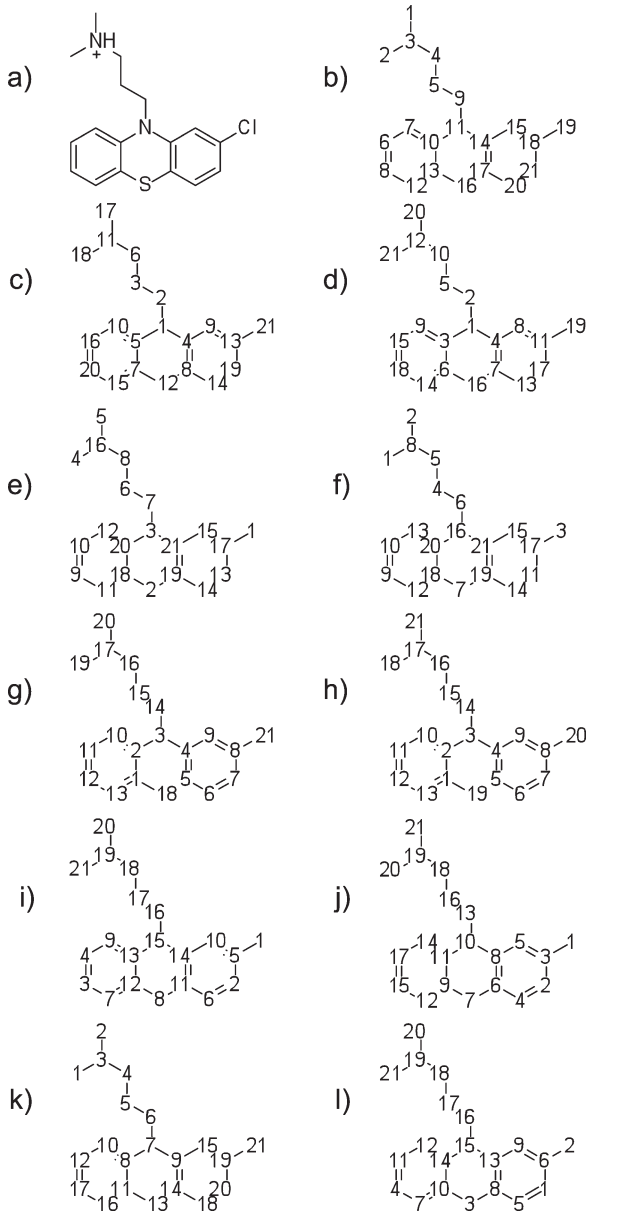

Figure 5. Canonical labels for (a) chlorpromazine with (b) PCA, (c) Jochum and Gasteiger algorithm, (d) modified Jochum and Gasteiger algorithm, (e) Weininger algorithm, (f) modified Weininger algorithm, (g) Prabhakar algorithm, (h) modified Prabhakar algorithm, (i) MVE with diffusion kernel and diffusion parameter 0.4, (j) MVE with Euclidean distance kernel, (k) Laplacian Eigenmaps, and (l) Isomap.

Neighborhood Preservation

As an example, Figure 5 presents the canonical labels generated for chlorpromazine with each of the 11 compared canonization algorithms. Both versions of the Prabhakar algorithm have a tendency to label consecutive paths in the graph. For both versions of the Gasteiger algorithm, the ground concept of number-

Table 9. Preserved Neighborhood Relations.

Algorithm	No. of preserved neighborhoods	% Preservation
Jochum Gasteiger	20,587	8
Jochum Gasteiger modified	22,801	9
Laplacian Eigenmaps	129,082	49
Isomap	105,152	40
MVE diffusion kernel 0.4	79,289	30
MVE Euclidean kernel	82,363	31
PCA	89,465	34
Prabhakar	163,691	62
Prabhakar modified	174,861	66
Weininger	24,734	9
Weininger modified	25,866	10

ing the vertices in spheres around the most buried atom is recognizable. The other methods tend to spread the labels in a nonintuitive manner. To further assess the differences between canonization algorithms, we analyzed to which extent each method is capable of preserving neighborhood relations during the reduction of the two-dimensional graph of potential pharmacophoric points to its one-dimensional form, the PhAST-sequence. For each method, 264,220 neighborhood relations were checked. The results are summarized in Table 9.

The large number of preserved neighborhoods with both variants of the Prabhakar algorithm (original: 62%, modified: 66%) is no surprise as both perform a depth-first search with a complex set of rules to decide which vertex is visited next. The creation of paths of consecutive canonical labels is inherent to this approach. The modified version preserves even more neighborhoods as the original algorithm, because it lacks the removal of terminal atoms as initial steps. As a consequence, paths through the molecule can include more atoms, and the fragments with consecutive canonical labels may be elongated.

Both variants of the Jochum–Gasteiger algorithm preserve least neighborhoods (original: 8%, modified: 9%). This is no surprise as well, because originating from the most buried vertices in the graph, they work in spheres around this centre. In this approach, it cannot be anticipated that the resulting canonical labels reflect neighborhoods to a high extent. If a vertex v_i in a sphere with n_i vertices receives canonical label x, all adjacent vertices from the next sphere of size n_j will get assigned canonical labels that are bound between $x + 1$ and $x + (n_i - 1) + n_j$. But the algorithm does not guarantee that these canonical labels will be directly subsequent to x or to each other. As with the Prabhakar algorithm, the modified version preserves more neighborhoods because of the fact that the terminal atoms were not treated separately.

As the number of connected vertices is the main criterion used for prioritization in the Weininger algorithm, it tends to work its way from the outside to the inside of a molecule. The initial equivalence classes are created based on further properties regarding all atoms at once, not limited to a certain subset as the vertices connected to the last labeled vertex as in the Prabhakar algorithm or atoms with the same buriedness as the Jochum–Gasteiger method. So, it does not group atoms by their affiliation to a certain region of the molecule but by the similarity of their overall properties. Further on, the algorithm divides vertices from the same equivalence class into unique subsets comprising only one vertex, so this initial partition cannot be reversed. This behavior is reflected in the low preservation of neighborhoods with 9% for the original and 10% for the modified version.

All methods for dimensionality reduction only moderately preserve the neighborhoods. For MVE (diffusion kernel: 30%, Euclidean distance kernel: 31%), Isomap (40%), and Laplacian Eigenmaps (49%), this was expected, as they were developed to preserve local distances between neighboring points in datasets as good as possible. They do not work in a greedy approach like the depth-first search used by the Prabhakar algorithm. By preserving the distances between neighboring pairs, distances between nonadjacent vertices may be changed. So, it was anticipated that the degree of preservation is lower than for the Prabhakar algorithm. For PCA (34%), however, this result is surprising. During PCA projection neighborhoods of vertices are not regarded explicitly, and different parts of a molecule may collapse in the same region of the PhAST-sequence, merging vertices from different parts of a molecule in the process.

MVE with the diffusion kernel that performed best in the retrospective comparison does not perform best in neighborhood preservation. The Prabhakar algorithm, which in the modified version preserves the most neighborhoods, does not perform best in the retrospective comparison. Pearson's correlation coefficient between the retrospective results using the best performing alignment evaluation method S_2, and the percentage of preserved neighborhoods during the reduction of the two-dimensional graph to a PhAST-sequence is 0.46. So, despite the fact that different approaches for graph canonization yield notably different results in this analysis; the percentage of preserved neighborhoods seems unsuited to explain why MVE performed best in the retrospective comparison.

For each vertex pair in the two-dimensional graph of pharmacophoric points, we checked whether their corresponding symbols are adjacent to each other in the PhAST-sequence. If not,

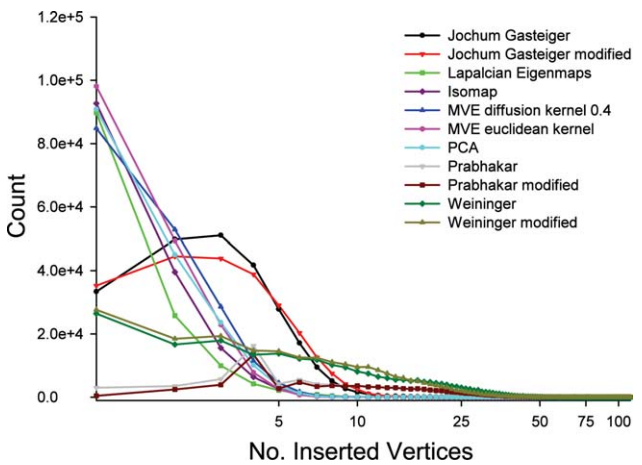

Figure 6. Number of inserted vertices between vertex pairs that are neighbors in the molecule graph, but not the PhAST-sequence. X-axis logarithmic to emphasize the interval in which the behavior of the canonization algorithms diverges most.

Table 10. Percentages of Neighborhood Relations Preserved and Changed Between PhAST-Sequences When Modifying a Molecule by Attaching Fragments.

	% Preserved	% Preserved original orientation	% Preserved transversed orientation	% Changed	% Changed original orientation	% Changed transversed orientation
Jochum Gasteiger	43.41	42.76	0.65	56.59	44.64	11.95
Jochum Gasteiger modified	40.65	39.47	1.17	59.35	46.43	12.93
Laplacian Eigenmaps	69.86	53.61	16.24	30.14	23.99	23.99
Isomap	64.63	43.25	21.38	35.37	24.55	10.82
MVE diffusion kernel 0.4	65.52	42.70	22.82	34.48	22.84	11.64
MVE Euclidean kernel	53.62	36.54	17.07	46.38	29.47	16.91
PCA	38.42	29.46	8.96	61.58	44.61	16.97
Prabhakar	75.60	72.76	2.84	24.40	19.25	5.15
Prabhakar modified	81.54	77.81	3.74	18.46	14.24	4.22
Weininger	62.03	62.01	0.01	37.97	36.75	1.22
Weininger modified	56.18	55.21	0.97	43.82	40.31	3.51

For both cases, the percentage of neighborhood relations in original orientation and in transversed orientation is shown in addition.

we counted by how many symbols they are separated. The result is presented in Figure 6, exact values are available from Table S1 in the Supporting Information. Overall, all methods for dimensionality reduction behave similarly. They separate neighboring vertices in the graph more often by only one vertex in the PhAST-sequence compared to the other canonization algorithms. For separations consisting of more than five vertices, they have the lowest count of occurrence among all methods. As both variants of the Prabhakar algorithm have the highest neighborhood preservation, their overall count of insertions between neighboring vertices is the lowest. They perform a depth-first search, and only reaching a dead end in this search can cause an event counted in this experiment. In case of these two algorithms, the number of vertices inserted between neighboring vertices gives the path length from a dead end in the depth-first search and the next unlabeled vertex.

The Jochum–Gasteiger algorithm works in spheres around the most buried vertex. Vertices from the same sphere receive consecutive labels. This explains why both variants of the algorithm have a high number of insertions with less than 10 inserted vertices. Most molecules analyzed here do not possess enough vertices to create spheres of more than 10 members, so the count of insertions of this size is very low.

Both variants of the Weininger algorithm lead to a medium number of insertions of sizes 1–7, but from thereon, they have the highest count for insertions. MVE performs best in retrospective studies for our method. But as well as the percent of preserved neighborhoods, the number of vertices inserted between originally neighboring vertices in the graph during the generation of the PhAST-sequence seems not to be suitable to explain this good performance.

We have demonstrated that different approaches for graph canonization show different behavior regarding the number of vertices they insert between vertices in the PhAST-sequences that were connected in the two-dimensional graph of pharmacophoric points. MVE with diffusion kernel turned out to be the best performing method in the retrospective comparison.

Robustness Against Structural Modification

We tested the robustness of the compared canonization methods by comparing the PhAST-sequence generated from a molecule with that generated from a molecule similar to the original but with a slight structural modification. For each pair of neighboring vertices in the original sequence, we checked whether they remain adjacent in the modified PhAST-sequence and whether they changed their relative orientation. We used six fragments for chemical structure modification (see Fig. 1) that were attached individually and in pairs. Each original PhAST-sequence was compared with 135 variants, 6699 molecules from the COBRA molecule library were investigated that way. The results are presented in Table 10.

Both variants of the Jochum–Gasteiger algorithm preserve around 40% of the neighborhoods from the original PhAST-sequence in the modified variants. Nearly all of these preserved relationships are kept in the original orientation, only a small amount of transversions is generated (original: 0.65%, modified: 1.17%). Nonpreserved neighborhoods are kept mostly in the original orientation as well (original: ~45%, modified: ~46%), enabling the global sequence alignment to compensate these changes by inserting gaps. Only around 12% of all neighborhood relations are not kept and transversed at the same time.

All methods for nonlinear dimensionality reduction keep more neighborhood relations but introduce transversions at the same time to a higher extent, foremost MVE in combination with the diffusion kernel with over 22% transversions. In case of disrupted neighborhoods, the fraction of created transversions is as high as for the Jochum–Gasteiger algorithm or even higher.

The aim of these algorithms for nonlinear dimensionality reduction is to keep pairwise distances between neighboring points while embedding a set of data points in a lower-dimensional space. So, they only consider relationships between pairs of points. Distances between two points are kept even if these points switch coordinates. So, these methods introduce a high amount of transversions, because this is a valid operation in their functioning. PCA preserves the least neighborhoods from the

Table 11. Time (Seconds) to Canonize the COBRA Library (Version 6.1, $n = 8{,}311$) on a Single CPU.

	Total	Mean	Max	Min	σ
Jochum Gasteiger	95.61	0.01150	2.93918	0.00007	0.07857
Jochum Gasteiger modified	116.65	0.01404	2.85762	0.00009	0.08386
Laplacian Eigenmaps	21.48	0.00258	0.08001	0.00019	0.00327
Isomap	20.32	0.00245	0.07376	0.00017	0.00320
MVE diffusion kernel 0.4	9639.19	1.15981	96.29566	0.08336	2.72437
MVE Euclidean kernel	10045.93	1.20875	101.30420	0.09399	2.86952
PCA	2.02	0.00024	0.03797	0.00013	0.00073
Prabhakar	35904.97	4.32017	3159.95603	0.00005	66.66446
Prabhakar modified	135843.44	16.34502	23524.26899	0.00005	402.00449
Weininger	4.46	0.00054	0.01046	0.00006	0.00048
Weininger modified	4.51	0.00054	0.07926	0.00006	0.00098

original to the modified PhAST-sequence, but most of them in original orientation, not as transversions. It introduces transversions in the disrupted neighborhoods to an extent comparable to the nonlinear methods for dimensionality reduction.

Both variants of the Weininger algorithm introduce the lowest fraction of transversions (original version: 1%, modified version: 4%). Neighborhood preservation is well above the Jochum–Gasteiger method.

Preservation of neighborhoods between related PhAST-sequences is highest for both variants of the Prabhakar algorithm (original version 76%, modified version 82%). This indicates that paths in the original and altered molecules are very similar. This is in perfect agreement with the low-transversion rate in preserved (3% and 5%) and nonpreserved neighborhoods (4% and 4%).

The correlation between retrospective results using the best performing alignment evaluation method S_2 and (i) the percentage of preserved neighborhoods is 0.57, (ii) the percentage of kept but transversed neighborhoods is 0.65, and (iii) the percentage of neighborhoods that are disrupted and transversed is 0.09. This indicates that transversions do not affect performance as drastically as we expected. The sequence alignment uses only mutations and insertions/deletions; a transversion can, thus, be treated by two mismatches or a combination of gap and mismatch. This explanation is backed up by (iii). None of the correlations is sufficiently strong to qualify the corresponding property as necessary for "good" retrospective results. However, when both variants of the Prabhakar algorithm are omitted as outliers, the correlation (i) increases to 0.93. We interpret this observation as an indication that the Prabhakar algorithm differs from the other canonization approaches (omitting other algorithms does not increase correlation as much: without the Jochum–Gasteiger algorithms: 0.39, without the Weininger algorithms: 0.57, and without the methods for nonlinear dimensionality reduction: 0.65). Indeed, there is such a difference: The subset of vertices from which the next vertex is chosen is the smallest of all approaches because of the depth-first like canonization process of the Prabhakar algorithm. The number of candidates is four or even less in most cases because of the distribution of vertex degrees in molecular graphs.[31] The Jochum–Gasteiger algorithm limits the number of candidates by the size of the current sphere, which is potentially larger than four levels. For the Weininger algorithm, the limit is given by the number of vertices with the same properties, which typically exceeds four as well. For the dimensionality reduction methods, the next vertex can potentially be chosen from all remaining vertices.

Until this point, we have combined the results for single and pairwise modifications of molecules. Treating both cases separately does not dramatically change the picture (Tables S3 and S4 of the Supporting Information). Both the Jochum–Gasteiger and the Weininger algorithms introduce the fewest transversions. Both versions of the Prabhakar algorithm preserve neighborhoods best, followed closely by the methods for nonlinear dimensionality reduction that preserve most neighborhood relations but not in the original orientation. PCA preserves neighborhoods least but does not introduce as many transversions as the nonlinear methods. Calculating Pearson's correlation coefficient between the retrospective evaluation scores and the percentage of neighborhoods kept in original orientation, kept in transversed orientation, and neighborhoods changed and transversed at the same time reveals in both cases a slightly different relationship as the combined evaluation. Using only the results of single (double) modifications, the correlation coefficient between retrospective performance and percentage of neighborhoods preserved is 0.57 (0.57). In case of transversed neighborhoods, the correlation coefficient is 0.64 (0.66). The major difference to the combined case is the relationship between retrospective performance and percentage neighborhoods changed and transversed, which correlates with -0.24 (-0.27).

In summary, transversions in the PhAST sequences of similar molecules do not affect the performance to a great extent. None of the investigated properties correlate strongly with retrospective virtual screening performance.

Canonization Time

Computational efficiency is important for rapid virtual screening. We compared our implementation of the canonization algorithms with respect to the time needed to process the COBRA library on a single central processing unit (CPU) of our cluster, atom typing excluded (Table 11). With only $\sim$2 s for all 8311 molecules, PCA was fastest. Both variants of the Weininger algorithm are fast with a time requirement of about 4 s. The Jochum–Gasteiger algorithm and all methods for nonlinear

dimensionality reduction take more time but are still feasible. For a prospective application, the Prabhakar algorithm might be too slow. For a medium-sized screening library with 0.5×10^6 compounds, canonization with this algorithm would take 25 days for the original and 95 days for the modified version (not including atom-typing). MVE with the diffusion kernel (best retrospective performance) would need ~7 days.

Conclusions and Outline

The canonization algorithm influences the performance of PhAST in virtual screening, although to a minor extent. None of the investigated properties of canonization algorithms seems useful as an *a priori* indicator of retrospective virtual screening performance by PhAST. The best retrospective performance was achieved using MVE with the diffusion kernel ($\beta = 0.4$), gap open penalty = 5, and gap extension penalty = 1. Different kernel functions vary MVE performance only slightly. Future work could investigate alternative canonization approaches based on the molecular graph[32] or MVE using other kernels like p-step random walks[33] to further improve performance. Kernels operating on the graph topology (as opposed to spatial vertex coordinates) have the advantage of being independent from the used layout/conformation algorithm. We used covalent bonds to define atom neighborhoods. In the original applications of these algorithms, neighborhoods were defined by connectivity algorithms like k-nearest neighbors,[22] b-matching,[23] or ε-balls. Using these instead of covalent bonds would render PhAST-sequences independent from the original connectivity, possibly increasing the chance for scaffold-hopping. The recently developed structure preserving embedding method by Shaw and Jebara[34] preserves global connectivity and might be a promising candidate for further investigation. This technique already showed good results in embedding 3D structures into two dimensions.[34]

MVE inserts many transversions into PhAST-sequences of similar molecules. Global sequence alignment can treat these only by mutations and insertions/deletions, and thus might not be the best metric in this situation. Other string metrics such as the Damerau–Levenshtein-distance[35] that are capable of using transversions as well as mutations and insertions/deletions might be promising alternatives, not only for PhAST but also for other string representations like SMILES.

Our study demonstrated that the alignment evaluation method influences performance more than the canonization algorithm. For all canonization methods, the alignment evaluation by alignment score yielded better results than the sequence identity calculated from the alignment. This has also been observed by studies on the alignment of protein sequences[8]; there, significance-based methods perform even better than the actual alignment score. Some techniques originally developed for local alignments seem promising for global alignments also.[36–38] A first simple step in this direction could be the use of Z-scores.[1,39] Until recently, one had to create a population of alignment scores by shuffling and realigning the originally compared sequences to estimate mean and standard deviation of alignment scores from alignments of random sequences. Booth et al. showed that it is possible (for the ungapped case) to calculate mean and standard deviation efficiently, avoiding the time-consuming realignment step.[40]

An important parameter of PhAST that was not changed in our study is the score matrix used to score matches and mismatches in the alignments. It directly influences the alignments, and thus the similarity score as well. The systematic development of a new score matrix that no longer depends on chemical intuition alone will be the subject of our future studies. Krier and Hutter[41] recently proposed a process for building a scoring scheme based on aligning SMILES of molecular fragments. Their score matrix reflects the frequencies of chemical replacements in pharmaceutical substances. For PhAST, a similar approach might be possible based on pharmacophoric points, resulting in a score matrix close to the original concept of Dayhoff et al.[42] Modification of the pharmacophoric points is another option, one should address at the same time.

With the alignment score as a measure for the evaluation of global alignments (instead of percent identity) the weighting of the influence of certain pharmacophoric points seems reasonable. These points could represent interactions that are necessary for binding. By upweighting the match and mismatch scores of important pharmacophoric points, one could force isofunctional points to be matched. If no such points exist, key interactions are missing resulting in a low score. Incorporating domain knowledge in this way could further improve the performance of PhAST.

Acknowledgments

V.H. is grateful for a Ph.D. scholarship granted by Merck KGaA.

References

1. Hähnke, V.; Hofmann, B.; Grgat, T.; Proschak, E.; Steinhilber, D.; Schneider, G. J Comput Chem 2009, 30, 761.
2. Weininger, D.; Weininger, A.; Weininger, J. L. J Chem Inf Comput Sci 1989, 29, 97.
3. Needleman, S. B.; Wunsch, C. D. J Mol Biol 1970, 48, 443.
4. Durbin, R.; Eddy, S. R.; Krogh, A.; Mitchison, G. Biological Sequence Analysis; Cambridge University Press: Cambridge, 1998.
5. Truchon, J. F.; Bayly, C. I. J Chem Inf Model 2007, 47, 488.
6. Pearson, K. Phil Trans R Soc 1896, 187, 253.
7. Kendall, M. Biometrika 1938, 30, 81.
8. Brenner, S. E.; Chothia, C.; Hubbard, T. J. P. Proc Natl Acad Sci USA 1998, 95, 6073.
9. Karlin, S.; Altschul, S. F. Proc Natl Acad Sci USA 1990, 87, 2264.
10. Jochum, C.; Gasteiger, J. J Chem Inf Comput Sci 1977, 17, 113.
11. Prabhakar, Y. S.; Balasubramanian, K. J Chem Inf Model 2006, 46, 52.
12. Pearson, K. Philos Mag 1901, 2, 559.
13. Belkin, M.; Niyogi, P. Neural Comput 2003, 15, 1373.
14. Tenenbaum, J. B.; de Silva, V.; Langford, J. C. Science 2000, 290, 2319.
15. Shaw, B.; Jebara, T. Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics; Omnipress: Madison, 2007.
16. Schneider, P.; Schneider, G. QSAR Comb Sci 2003, 22, 713.
17. Zhao, W.; Hevener, K. E.; White, S. W.; Lee, R. E.; Boyett, J. M. BMC Bioinformatics 2009, 10, 225.

18. Massey, F. J. J Am Stat Ass 1951, 46, 68.
19. Proschak, E.; Wegner, J. K.; Schüller, A.; Schneider, G.; Fechner, U. J Chem Inf Model 2007, 47, 295.
20. Vingron, M.; Waterman, M. S. J Mol Biol 1994, 235, 1.
21. Morgan, H. L. J Chem Doc 1965, 5, 107.
22. Cover, T.; Hart, P. IEEE Trans Info Theory 1967, 13, 21.
23. Müller-Hannemann, M.; Schwartz, A. J Exp Algor 2000, 5.
24. Floyd, R. W. Comm ACM 1962, 5, 345.
25. Warshall, S. J ACM 1962, 9, 11.
26. Schölkopf, B.; Smola, A.; Müller, K. Neural Comput 1998, 10, 1299.
27. Kondor, R. I.; Lafferty, J. D. Proceedings of the Nineteenth International Conference on Machine Learning; Morgan Kaufmann Publishers Inc.: San Francisco, 2002.
28. Tsuda, K.; Noble, W. S. Bioinformatics 2004, 20, 326.
29. Doolin, D.; Dongarra, J.; Seymour, K. Sci Program 1999, 7, 111.
30. Borchers, B. Opt Meth Soft 1999, 11, 613.
31. Rupp, M. Kernel Methods for Virtual Screening; Johann Wolfgang Goethe-University: Frankfurt am Main, 2009.
32. Faulon, J. L.; Collins, M. J.; Carr, R. D. J Chem Inf Comput Sci 2004, 44, 427.
33. Smola, A. J.; Kondor, R. I. Proceedings of the 16th Annual Conference on Computational Learning Theory and 7th Kernel Workshop; Springer: Berlin and Heidelberg, 2003.
34. Shaw, B.; Jebara, T. Proceedings of the 26th International Conference on Machine Learning; Omnipress: Madison, 2009.
35. Damerau, F. J. Commun ACM 1964, 7, 171.
36. Newberg, L. A. J Comput Biol 2008, 15, 1187.
37. Chia, N.; Bundschuh, R. J Comput Biol 2006, 13, 429.
38. Hartmann, A. K. Phys Rev E 2002, 65, 056102.
39. Lipman, D. J.; Pearson, W. R. Science 1985, 227, 1435.
40. Booth, H. S.; Maindonald, J. H.; Wilson, S. R.; Gready, J. E. J Comput Biol 2004, 11, 616.
41. Krier, M.; Hutter, M. C. J Chem Inf Model 2009, 49, 1280.
42. Dayhoff, M. O.; Schwartz, R. M.; Orcutt, B. C. Atlas of Protein Sequence and Structure; National Biomedical Research Foundation: Washington, DC, 1978.