Soft Computing manuscript No.
(will be inserted by the editor)

An Empirical Approach For Probing the Definiteness of Kernels*

Martin Zaefferer · Thomas Bartz-Beielstein · Günter Rudolph

Received: date / Accepted: date

Abstract Models like support vector machines or Gaussian process regression often require positive semi-definite kernels. These kernels may be based on distance functions. While definiteness is proven for common distances and kernels, a proof for a new kernel may require too much time and effort for users who simply aim at practical usage. Furthermore, designing definite distances or kernels may be equally intricate. Finally, models can be enabled to use indefinite kernels. This may deteriorate the accuracy or computational cost of the model. Hence, an efficient method to determine definiteness is required. We propose an empirical approach. We show that sampling as well as optimization with an evolutionary algorithm may be employed to determine definiteness. We provide a proof-of-concept with 16 different distance measures for permutations. Our approach allows to disprove definiteness if a respective counter-example is found. It can also provide an estimate of how likely it is to obtain indefinite kernel matrices. This provides a simple, efficient tool to decide whether additional effort should be spent on designing/selecting a more suitable kernel or algorithm.

Keywords Definiteness · Kernel · Distance · Sampling · Optimization · Evolutionary Algorithm

1 Introduction

The definiteness of kernels and distances is an important issue in statistics and machine learning (Feller 1971; Vapnik 1998; Schölkopf 2001). One application that recently gained interest is the field of surrogate model-based combinatorial optimization (Moraglio and Kattan 2011; Zaefferer et al. 2014b; Bartz-Beielstein and Zaefferer 2017). Continuous distance measures are replaced by distance measures that are adequate for the respective search space (e.g., permutation distances or string distances). Such a measure will have an effect on the definiteness of the employed kernel function. For arbitrary problems, practitioners may come up with any kind of suitable distance measure or kernel.

While it is easy to determine definiteness of matrices, determining the definiteness of a function is not as simple. Proving definiteness by theoretical means may be infeasible in practice (Murphy 2012, p. 482; Ong et al. 2004). It may be equally difficult to design a function to be definite. Finally, algorithms may be adapted to handle indefinite kernels. These adaptations usually have a detrimental impact on the computational effort or accuracy of the derived model. Hence, this study tries to answer the following two research questions:

Q$_1$ **Discovery:** Is there an efficient, empirical approach to determine the definiteness of kernel functions based on arbitrary distance measures?

*preprint submitted to Soft Computing journal, Springer

Martin Zaefferer (corresponding author)
Faculty of Computer Science and Engineering Science,
TH Köln - University of Applied Sciences,
Steinmüllerallee 1, 51643 Gummersbach, Germany
Telephone: +49 2261 8196 6327
E-mail: Martin.Zaefferer@th-koeln.de

Thomas Bartz-Beielstein
Faculty of Computer Science and Engineering Science,
TH Köln - University of Applied Sciences,
Steinmüllerallee 1, 51643 Gummersbach, Germany

Günter Rudolph
Department of Computer Science, TU Dortmund University
Otto-Hahn-Str. 14, 44227 Dortmund, Germany

Q$_2$ **Measurability:** If Q$_1$ can be answered affirmatively, can we quantify to what extent a lack of definiteness is problematic?

Q$_1$ tries to find an answer to the general question of whether or not a function is definite. Measurability (Q$_2$) is important, as this may allow determining the impact that indefiniteness has in practice. For instance, if a kernel rarely produces indefinite matrices, an optimization or learning process that explores only a small subset of the search space may not be negatively affected. Therefore, this article proposes two approaches.

A$_1$ **Sampling:** Random sampling is used to determine the proportion of solution sets with indefinite distance or kernel matrices for a given setting.

A$_2$ **Optimization:** Maximizing the largest eigenvalue related to a certain solution set with an Evolutionary Algorithm (EA), hence finding indefinite cases even when they are rare.

If either approach detects an indefinite matrix, the respective kernel function is demonstrated to be indefinite. On the other hand, if no indefinite matrix is detected, definiteness of the function is not proven. Still, this indeterminate result may indicate that indefiniteness is at least unlikely to occur. Hence, the function may be treated as unproblematic in practical use.

Section 2 provides the background of the methods presented in Section 3. The experimental setup for a proof-of-concept, based on permutation distance measures, is described in Section 4. Results of the experiments are analyzed and discussed in Section 5. Finally, a summary of this work as well as an outlook on future research directions is given in Section 6.

2 Background: Distances, Kernels and Definiteness

2.1 Distance Measures

Distance measures compute the dissimilarity of two objects $x, x' \in \mathcal{X}$, where we do not assume anything about the nonempty set $\mathcal{X}$. Such objects can be, e.g., permutations, trees, strings, or vectors of real values. Thus, a distance measure $d: \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$ expresses a scalar, numerical value $d(x, x')$ that should become larger the more distinct the objects x and x' are. For a set of $n \in \mathbb{N}$ objects, the distance matrix D of dimension $n \times n$ collects all pairwise distances $D_{ij} = d(x_i, x_j)$ with $i = 1, \ldots, n$ and $j = 1, \ldots, n$. Intuitively, distances can be expected to satisfy certain conditions, e.g., they should be zero when comparing identical objects.

A more formal definition is implied by the term *distance metric*. A distance metric $d(x, x')$ is symmetric $d(x, x') = d(x', x)$, non-negative $d(x, x') \geq 0$, preserves identity $d(x, x') = 0 \iff x = x'$, and satisfies the triangle inequality $d(x, x'') \leq d(x, x') + d(x', x'')$. Distance measures that do not preserve identity are often called pseudo-metrics.

An important class of distance measures are edit distance measures. Edit distance measures can be defined to count the minimal number of edit operations required to transform one object into another. An edit distance measure may concern one specific edit operation (e.g., only swaps) or a set of different operations (e.g., Levenshtein distance with substitutions, deletions, insertions). Edit distances usually satisfy the metric axioms.

2.2 Kernels

In the following, a kernel (also: kernel function, similarity measure or correlation function) is defined as a real valued function $k(x, x')$ with

$$\begin{aligned} k : \mathcal{X} \times \mathcal{X} &\to \mathbb{R} \\ (x, x') &\mapsto k(x, x') \end{aligned} \tag{1}$$

that will usually be symmetric $k(x, x') = k(x', x)$ and non-negative $k(x, x') \geq 0$ (Murphy 2012, p. 479). Kernels can be based on distance measures, i.e., $k(d(x, x'))$.

2.3 Definiteness of Matrices

One important property of kernels and distance measures is their definiteness. We refer to the literature for more in-detail descriptions and proofs, which are the basis for the following paragraphs (Berg et al. 1984; Schölkopf 2001; Camastra and Vinciarelli 2008).

First, we introduce the concept of matrix definiteness. A symmetric, square matrix A of dimension $n \times n$ ($n \in \mathbb{N}$) is positive definite (PD) if and only if

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j A_{ij} > 0,$$

for all $c \in \mathbb{R}^n \setminus \{0\}$. This is equivalent to all eigenvalues $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$ of the matrix A being positive. Due to symmetry, the eigenvalues are $\lambda \in \mathbb{R}^n$. Respectively, a matrix is negative definite (ND) if all eigenvalues are negative. If eigenvalues are non-negative (i.e., some are zero) or non-positive, the matrix is respectively called Positive or Negative Semi-Definite (PSD, NSD). If mixed signs are present in the eigenvalues, the matrix may be called indefinite. Kernel (or correlation, covariance) matrices are examples of matrices that have to be PSD.

A broader class of matrices are Conditionally PSD or NSD (CPSD, CNSD). Here, the coefficients satisfy

$$\sum_{i=1}^{n} c_i = 0, \tag{2}$$

with $n > 1$. All PSD (NSD) matrices are CPSD (CNSD). To check conditional definiteness, let the $n \times n$ matrix P be

$$P = \begin{pmatrix} I_{(n-1)} - \mathbf{e}\mathbf{e}^T/n & \mathbf{e}/n \\ (0, \dots, 0) & 1 \end{pmatrix}$$

with $\mathbf{e} = (1, \dots, 1)^T$, and $B = PAP^T$. Then, A is CNSD if and only if

$$\hat{A} = B_{n-1} \tag{3}$$

is NSD (Ikramov and Savel'eva 2000, Algorithm 1). Here, B_{n-1} is the leading principal submatrix of B, that is, the last column and row of B are removed.

2.4 Definiteness of Kernel Functions

In a similar way to the definiteness of matrices, definiteness can also be determined for kernel functions. The upcoming description roughly follows the definitions and notations by Berg et al. (1984) and Schölkopf (2001). For the nonempty set $\mathcal{X}$, a symmetric kernel k is called PSD if and only if

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j k(x_i, x_j) \geq 0,$$

for all $n \in \mathbb{N}$, $x \in \mathcal{X}$ and $\mathbf{c} \in \mathbb{R}^n$. A PSD kernel will always yield PSD kernel matrices.

An important special case are conditionally definite functions. Analogous to the matrix case, they observe the respective condition in equation (2). One example of a CNSD function is the Euclidean distance. The importance of CNSD functions is also due to the fact that the distance measure $d(x, x')$ is CNSD if and only if the kernel $k(x, x') = \exp(-\theta d(x, x'))$ is PSD $\forall\ \theta > 0$ (Schölkopf 2001, Proposition 2.28).

It is often stated that kernels must be PSD (Rasmussen and Williams 2006; Curriero 2006). Still, even an indefinite kernel function may yield a PSD kernel matrix. This depends on the specific data set used to train the model (Burges 1998); (Li and Jiang 2004, Theorem 2) as well as the parameters of the kernel function. Some frequently used kernels are known to be indefinite. Examples are the sigmoid kernel (Smola et al. 2000; Camps-Valls et al. 2004) or time-warp kernels for time series (Marteau and Gibet 2014).

To handle the issue of a kernel's definiteness, different (not mutually exclusive) approaches can be found in the literature.

- **Proving:** Definiteness of a specific function can be *proven* (or disproven) by theoretical considerations in some cases (cf. (Berg et al. 1984)). For complex cases, or practitioners this may be an infeasible approach (Ong et al. 2004).
- **Designing:** Functions can be *designed* to be definite (Haussler 1999; Gärtner et al. 2003; Marteau and Gibet 2014). Especially noteworthy are the so called convolution kernels (Haussler 1999), as they provide a method to construct PSD kernels for structured data. For a similar purpose, Gärtner et al. (2004) show how to design a syntax based PSD kernel for structured data. However, convolution kernels may be hard to design (Gärtner et al. 2004). Also, kernels and distance measures may be predetermined for a certain application.
- **Adapting:** Algorithms or kernel functions may be *adapted* to be usable despite a lack of definiteness. This, however, may affect the computational effort or accuracy of the derived model. Some approaches alter matrices (or rather, their eigenspectrum), hence enforcing PSDness (Wu et al. 2005; Chen et al. 2009; Zaefferer and Bartz-Beielstein 2016). For the case of SVMs, Loosli et al. (2015) provide a nice comparison of various approaches of this type and propose an interesting new solution to the issue of indefinite kernels based on learning in Krein spaces. A recent survey is given by Schleif and Tino (2015).

Between these three approaches, there is a lack of straightforward empirical procedures, without resorting to complex theoretical reasoning. How can the definiteness of a function be determined? And what impact does a lack of definiteness have on a model?

Hence, we propose the two related empirical approaches A_1 and A_2 introduced in Sec. 1 to fill the gap. An empirical approach may help to overcome the difficulty of theoretical considerations or designed kernels. Empirical results may also be a starting point for a more formal approach. Furthermore, it may give a quick answer on whether or not the algorithm will have to be adapted for non-PSD matrices (which, if applied by default, would require additional computational effort and may limit the accuracy of derived models).

3 Methods for Estimating Definiteness

We propose an experimental approach to determine and analyze definiteness (A_1, A_2). As a test case, we determine the definiteness of a distance-based exponential kernel, $k(x, x') = exp(-\theta d(x, x'))$. The kernel is definite if the underlying distance function is CNSD. For a given distance matrix, CNSDness is determined by the largest eigenvalue λ_n of $\hat{D}$, based on equation (3). D is not CNSD if $\lambda_n > 0$. We could also probe the kernel matrix K, but in this case the kernel parameter θ would have to be dealt with.

Of course, we cannot simply check definiteness of a single matrix D, since this would be only one possible outcome of the respective kernel or distance function. Hence, a large number of solution sets with respective distance matrices has to be generated to determine whether any of the matrices are CNSD (research question Q_1: discovery) and to what extent this may affect a model (Q_2: measurability). For smaller, finite spaces a brute force approach may be viable. All potential matrices D can be enumerated and checked for CNSDness. Since this quickly becomes computational infeasible, we propose to use sampling or optimization instead.

3.1 Estimating Definiteness with Random Sampling

To estimate the definiteness of a distance or kernel function we propose a simple random sampling approach (A_1). This approach randomly generates $t \in \mathbb{N}$ sets $X_1, \dots, X_t$. Each set has size n, that is, it contains $n \in \mathbb{N}$ candidate samples $X = \{x_1, \dots, x_n\}$.

For each set, the distance matrix D is computed, containing distances between all candidates in the set. Based on this, $\hat{D}$ is derived from equation (3). Then, the largest eigenvalue λ_n of $\hat{D}$ is computed. This eigenvalue determines whether $\hat{D}$ is NSD, and hence whether D is CNSD and K PSD. This is repeated for all t sets. The number of times that the largest eigenvalue is positive ($\lambda_n > 0$) is retained as $n_{\lambda+}$. Accordingly, the proportion of non-CNSD matrices is determined with $p = \frac{n_{\lambda+}}{t}$.

Obviously, all distance measures that yield $p > 0$ are proven to be non-CNSD. Hence, an exponential kernel based on these measures is also proven to be indefinite. If $p = 0$, CNSDness is not proven or disproven.

In general, the proposed method can be categorized as a randomized algorithm of the complexity class RP (Motwani and Raghavan 1995, p. 21f.). That is, it stops after polynomially many steps, and if the output is “no” then the distance measure is non-CNSD with probability 1, and if the output is “yes” then the distance measure is CNSD with some probability strictly bounded from zero.

The parameter p is an estimator of how likely a non-CNSD matrix is to occur, for the specified set size n. To determine definiteness, the calculation of λ_n of $\hat{D}$ is not mandatory, but it may be useful to see how close to zero λ_n is, to distinguish between a pathological case and cases where the matrix is just barely non-CNSD. We will show in Section 5.3 that λ_n of $\hat{D}$ can be linked to model quality.

Note, that inaccuracies of the numerical algorithm used to compute the eigenvalues might lead to an erroneous sign of the largest eigenvalue. To deal with that, one could try to use exact or symbolic methods or else use a tolerance when checking whether the largest eigenvalue is larger than zero. In the latter case, a matrix $\hat{D}$ is assumed to be non-NSD if $\lambda_n > \epsilon$, where ϵ is a small positive number.

3.2 Estimating Definiteness with Directed Search

If very few sets X yield indefinite matrices, A_1 may fail to find indefinite matrices by pure chance. In such cases, it may be more efficient to replace random sampling with a directed search (A_2). In detail, a set X can itself be viewed as a candidate solution of an optimization problem. The largest eigenvalue λ_n of the transformed distance matrix $\hat{D}$ is the objective to be maximized. By maximizing the largest eigenvalue, a positive λ_n may be found more quickly (and more reliably).

This optimization problem is strongly dependent on the kind of solution representation used. Evolutionary Algorithms (EAs) are a good choice to solve this problem, because they are applicable to a wide range of solution representations (e.g., real values, strings, permutations, graphs, and mixed search spaces). EAs use principles derived from natural evolution for the purpose of optimization. That is, EAs are optimization algorithms based on a cyclic repetition of parent selection, recombination, mutation, and fitness selection (see e.g., Eiben and Smith (2003)). These operations can be adapted to a large variety of representations, mainly depending on suitable mutation and recombination operators. The EA in this study will operate as follows:

- **Individual:** X**.** A set $X = \{x_1, \dots, x_n\}$ with set size n is considered as an individual. Set elements x are samples in the actual search or input space, as used throughout Sec. 2.
- **Search space:** S^n**.** All possible sets X of size n, i.e., $X \in S^n$.
- **Population:** Z**.** A population of size r, containing $X_k \in Z$ with $k \in \{1, \dots, r\}$ and $Z \subseteq S^n$.

- **Objective Function:**

$$f:\ S^n \to \mathbb{R} \qquad X \mapsto \lambda_n \tag{4}$$

  where λ_n is the largest eigenvalue of the transformed distance matrix $\hat{D}$ based on equation (3). The objective function f is maximized.
- **Mutation: Alteration of an individual.** $X_{\text{new}} = mutation(X) = \{x_1, \dots, x_{j-1}, submutation(x_j), x_{j+1}, \dots, x_n, \}$, with $j \in \{1, \dots, n\}$. For the submutation function, any edit operation that works for a sample x can be chosen: $x_{new} = submutation(x) = edit(x)$.
  For example, in case of permutations, one permutation $x_j \in X$ is chosen and mutated with typical permutation edit-operations (swap, interchange, reversal). The specific edit-operation is called submutation operator, to distinguish between mutation of the individual set X and the submutation of a single permutation $x_j \in X$.
- **Recombination: Combining two sets.** For recombination, two sets are randomly split and the parts of both sets are joined to form a new set of the same size.
- **Repair: Duplicate removal.** Mutation and recombination may create duplicates ($x_i = x_j$ with $i \neq j$). In practice, duplicates are not desirable and are irrelevant to the question of definiteness. Hence, duplicates are replaced by randomly generated, unique samples $x^* \notin X$.
- **Stopping criterion: Indefiniteness proven or budget exhausted.** The optimization can stop when some solution set X is found which yields $\lambda_n > \epsilon$, where ϵ is a small positive number. Alternatively, the process stops if a budget of objective function evaluations is exhausted.

4 Experimental Validation

The proposed approaches can be useful in any case where definiteness of kernels is of interest. The experiments provide a proof of concept of the proposed approaches. Hence, we chose to pick a recent application as a motivation for our experiments: surrogate-model based optimization in permutation spaces (Moraglio and Kattan 2011; Zaefferer et al. 2014a).

In many real-world optimization problems, objective function evaluations are expensive. Sequential modeling and optimization techniques are state-of-the-art in these settings (Bartz-Beielstein and Zaefferer 2017). Typically, an initial model is built at the first stage of the process. The model will be subsequently refined by adding further data until the budget is exhausted. At each stage of this sequential process, available information from the model is used to determine promising new candidate solutions. This motivates the rather small data set sizes used throughout this study.

4.1 Test Case: Permutations

As a proof-of-concept, we selected 16 distance measures for permutations; see Table 1 for a complete list. The implementation of these distance measures is taken from the R-package `CEGO`[1].

Similarly to Schiavinotto and Stützle (2007) we define Π^m as the set of all permutations of the numbers $\{1, 2, \dots, m\}$. A permutation has exactly m elements. We denote a single permutation with $\pi \in \Pi^m$ and $\pi = \{\pi_1, \pi_2, \dots, \pi_m\}$ where π_i is a specific element of the permutation at position i. For example, a permutation in this notation is $\pi = \{3, 2, 1, 4, 5\} \in \Pi^5$. Explanations and formulas (where applicable) for the distance measures are given in appendix A.

Table 1 Distance measures for permutations. Second column lists runtime complexity where m is the number of elements of the permutation. Metric refers to permutation space; these measures may be non-metric in other spaces. LC is short for Longest Common.

name	complexity	metric	abbreviation
Levenshtein	$O(m^2)$	yes	Lev
Swap	$O(m^2)$	yes	Swa
Interchange	$O(m^2)$	yes	Int
Insert	$O(mlog(m))$	yes	Ins
LC Substring	$O(m^2)$	yes	LCStr
R	$O(m^2)$	yes	R
Adjacency	$O(m^2)$	pseudo	Adj
Position	$O(m^2)$	yes	Pos
Position^2	$O(m^2)$	no	Posq
Hamming	$O(m)$	yes	Ham
Euclidean	$O(m)$	yes	Euc
Manhattan	$O(m)$	yes	Man
Chebyshev	$O(m)$	yes	Che
Lee	$O(m)$	yes	Lee
Cosine	$O(m)$	no	Cos
Lexicographic	$O(mlog(m))$	yes	Lex

4.2 Random Sampling

In a single experiment, $t = 10,000$ sets of permutations are randomly generated. Each set contains n permutations and each permutation has m elements. For each

[1] The package `CEGO` is available on CRAN at `http://cran.r-project.org/package=CEGO`.

set, the largest eigenvalue λ_n of $\hat{D}$ is computed based on equation (3). To summarize all t sets, the largest λ_n as well as the ratio $p = \frac{n_{\lambda+}}{t}$ are recorded. The tolerance value used to check whether the largest eigenvalue is positive is ϵ=1e-10. This process is repeated 10 times, to achieve a reliable estimate of the recorded values.

Two batches of experiments are performed. In the first, all 16 distance measures are examined, with $n = \{4, \dots, 20\}$ and $m = \{4, \dots, 15\}$. In the second batch, larger sizes $n = \{21, \dots, 40, 45, 50, 60, 70, 80, 90, 100\}$ are tested, but the permutations are restricted to $m = \{5, \dots, 15\}$ and the distance measures are only LCStr, Insert, Chebyshev, Levenshtein and Interchange.

4.3 Directed Search

To be comparable to the random sampling approach, the budget for each EA run is $10,000$ fitness function evaluations. A run will stop if the budget is exhausted or if $\lambda_n > \epsilon = 10^{-10}$. The population size of the EA is set to 100. The recombination rate is 0.5, the mutation rate is $1/m$ and truncation selection is used.

To identify bias introduced by the choice of the submutation operator (which may have a strong interaction with the respective distance measures), each EA run is performed repeatedly with three different submutation operators:

- **Swap mutation**: Transposing two adjacent elements of the permutation.

$$\pi = \pi_1, \dots, \pi_a, \pi_b, \dots, \pi_m$$
$$\pi^* = \pi_1, \dots, \pi_b, \pi_a, \dots, \pi_m,$$

  with $1 \leq a < (m-1)$ and $b = a + 1$.
- **Interchange mutation**: Transposing two arbitrary elements of the permutation.

$$\pi = \pi_1, \dots, \pi_{a-1}, \pi_a, \pi_{a+1}, \dots, \pi_{b-1}, \pi_b, \pi_{b+1}, \dots, \pi_m$$
$$\pi^* = \pi_1, \dots, \pi_{a-1}, \pi_b, \pi_{a+1}, \dots, \pi_{b-1}, \pi_a, \pi_{b+1}, \dots, \pi_m,$$

  with $1 \leq a \leq m$ and $1 \leq b \leq m$.
- **Reversal mutation**: Reversing a substring of the permutation.

$$\pi = \pi_1, \dots, \pi_a, \pi_{a+1}, \dots, \pi_{b-1}, \pi_b, \dots, \pi_m$$
$$\pi^* = \pi_1, \dots, \pi_b, \pi_{b-1}, \dots, \pi_{a+1}, \pi_a, \dots, \pi_m,$$

  with $1 \leq a < b \leq m$.

All 16 distance measures are tested, with $n = \{4, \dots, 20\}$ and $m = \{4, \dots, 15\}$. With ten repeats, and the three different submutation operators, this results into $97,920$ EA runs, each with $10,000$ fitness function evaluations. The employed EA implementation is part of the R-package `CEGO`.

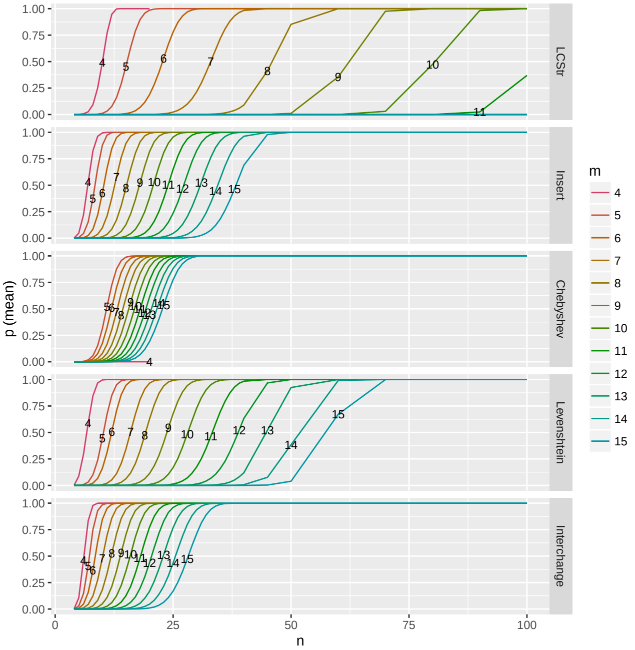

Fig. 1 Mean proportion of sets yielding matrices $\hat{D}$ with positive λ_n found with random sampling. Only distance measures that achieved positive λ_n are shown. The numeric labels and the color indicate the respective value of m.

4.4 Tests For Other Search Domains

To show that the proposed approach is not limited to the presented permutation distance example, we also briefly explore other search domains and their respective distances. However, these are not analyzed in further detail. Instead, we provide a list of minimal examples in Appendix B: the smallest (w.r.t. dimension) indefinite distance matrix for each tested distance measure. The examples include distance measures for permutations, signed permutations, trees, and strings.

5 Observations and Discussion

5.1 Sampling Results

The proportions of sets with positive eigenvalues (p) are summarized in Fig. 1. The largest eigenvalues are depicted in Fig. 2. Only the five indefinite distance measures, which achieved positive eigenvalues are shown: Longest Common Substring, Insert, Chebyshev, Levenshtein, Interchange.

No counter-examples are found for the remaining eleven measures. That does not prove that they are CNSD (although some are CNSD, e.g., Euclidean distance, Swap distance (Jiao and Vert 2015), Hamming distance (Hutter et al. 2011)), but indicates that it may be unproblematic to use them in practice. Some

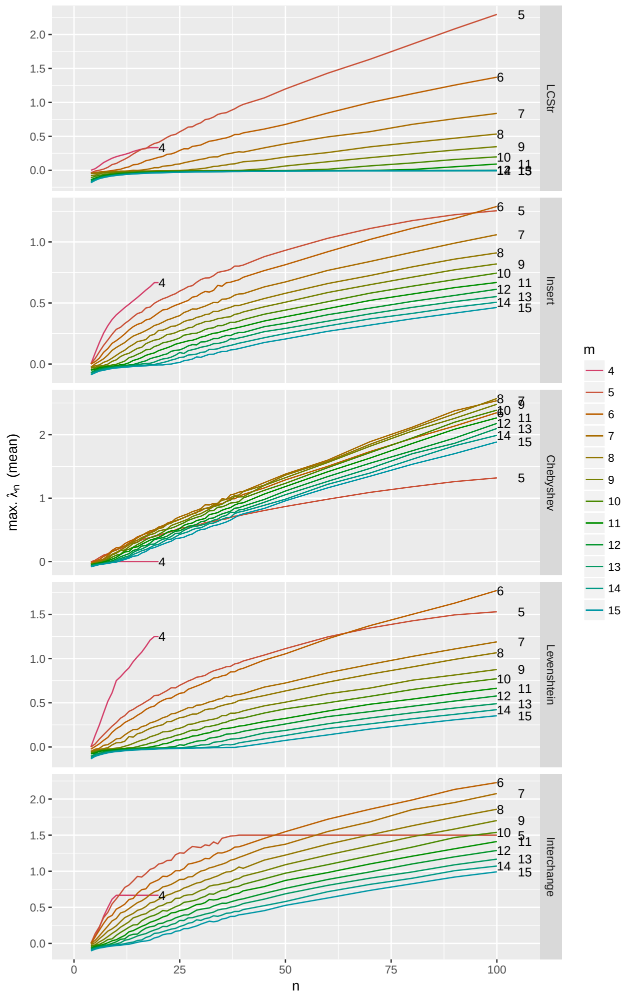

Fig. 2 Mean of maximum λ_n of $\hat{D}$ found with random sampling. The maximum is determined over all sets in a single experiment. The mean is determined over ten repeats of the sampling. Only distance measures that achieved positive λ_n are shown. The numbers inside the plot and the color indicate the respective value of m.

of the five non-CNSD distance measures were reported to work well in a surrogate-model optimization framework, e.g., Levenshtein distance seemed to work well for modeling of scheduling problems (Zaefferer et al. 2014a). This is mainly due to the fact, that even a non-CNSD distance *may* yield PSD kernel matrix K, depending on the specific data set and kernel parameters used. We do not suggest that non-CNSD distance measures should be avoided, but that their application should be handled with care.

Regarding values of p (Fig. 1), some trends can be observed. For indefinite distance measures, increasing the set size (n) will in general lead to larger values of p. Obviously, a larger set is more likely to contain combinations of samples that yield negative eigenvalues. In addition, the lower bound for eigenvalues decreases with increasing matrix size (Constantine 1985).

In contrast to the set size, increasing the number of permutation elements (m) decreases the proportion of positive eigenvalues p in all five cases. This can be attributed to a larger and hence more difficult search space. Overall, none of the distance measures shows exactly the same behavior. LCStr distance has the least problematic behavior. Only very few sets (of comparatively large size) yielded positive eigenvalues with LCStr distance. Interchange, Levenshtein and Insert distance all have relatively large p for small set sizes n. Chebyshev on the other hand, starts to have non-zero p for relatively large n. However, the number of permutation elements has only a weak influence in case of Chebyshev distance. Hence, curves for different m are much closer to each other, compared to the other distance measures. Somewhat analogous to p, the largest eigenvalues of $\hat{D}$ plotted in Fig. 2 are generally increasing for larger n, and decreasing with larger m.

Our findings are confirmed by some results from literature. Cortes et al. (2004) have shown that an exponential kernel function based on the Levenshtein distances is indefinite for strings of more than one symbol. Our experiments show that this result can be easily rediscovered empirically, for the case of permutations. At the same time, these findings also confirm (and provide reasons for) problems observed with these kinds of kernel functions in a previous study (Zaefferer et al. 2014a).

As a consistency check, Table 2 compares the sampling results to a brute force approach for $n = \{4, \ldots, 8\}$ and $m = 4$. It shows that the sampling indeed approximates the number of non-CNSD matrices quite well. In this small set, the sampling identified all combinations of distance measure and n that may yield non-CNSD matrices.

5.2 Optimization Results

The average number of fitness function evaluations (i.e., number of times λ_n of $\hat{D}$ is computed) required to find a matrix $\hat{D}$ with positive λ_n is depicted in Fig. 3. The optimization results show similar behavior with respect to n and m as the sampling approach. Increasing m leads to an increased number of fitness function evaluations. That means, finding positive eigenvalues becomes more difficult with increasing values of m. Increasing n reduces the number of required fitness function evaluations. That is, finding positive eigenvalues becomes easier. In some cases, this effect disappears for large values of m, e.g., for LCStr distance, where the average is more or less constant over n, if m is large enough.

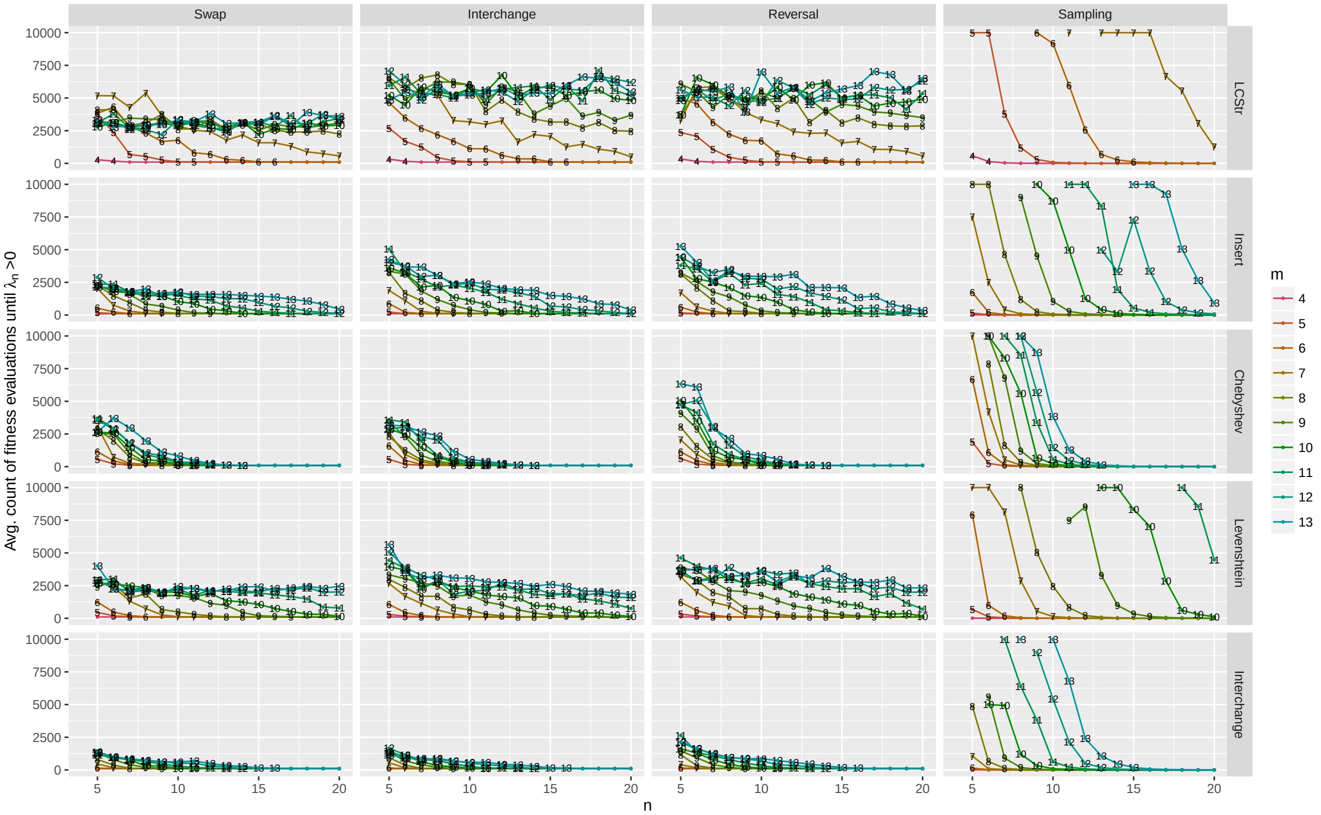

Fig. 3 Average number of fitness function evaluations until a positive eigenvalue was found. Missing nodes indicate that no positive eigenvalues are found, within the given budget. Averages are computed after removal of cases where no positive values are found. Columns show the results of the EA with each submutation function and the earlier described sampling approach. The rows indicate different distances measures. Only distance measures which achieved positive eigenvalues are shown.

Importantly, the comparison to the sampling results clearly shows that the EA has some success in optimizing the largest eigenvalue of the transformed distance matrix $\hat{D}$. In several cases, positive eigenvalues are found by the EA while sampling with the same budget failed to find any. Hence, it can be assumed that the fitness landscape based on λ_n of $\hat{D}$ is sufficiently smooth to allow for optimization. The eigenvalue λ_n seems to be a good indicator of how close a solution set is to yielding an indefinite kernel matrix.

Furthermore, the expected bias of the used submutation operator becomes visible. For Insert and LCStr distance, the EA with swap mutation works considerably better than the other two variants. Hence, comparisons of these values across different distance measures should be handled with caution. Clearly, other aspects of the optimization algorithm (e.g., selection criteria or recombination operators) might have similar effects. While this bias is troubling, results may still offer interesting insights. The eigenvalue optimization could be interpreted as a worst-case scenario that occurs if an iterative learning process strongly correlates with the eigenvalues of the employed distance matrix.

5.3 Verification: Impact on Model Quality

Earlier, we discussed two values that may express the effect of the lack of definiteness in practice, i.e., p and λ_n of $\hat{D}$. But what do these values imply?

The value p can be seen as a probability of generating indefinite matrices. If we assume that a model is unable to deal with indefinite data, the fraction p is an estimate of how likely a modeling failure is. For other cases, it is very hard to link it to a model performance measure such as accuracy without making too many assumptions. We still argue that p provides useful information, especially when the kernel is designed and probed before sampling any data (e.g., when planning an experiment). We suggest to use p to support an initial decision (e.g., whether to spend additional time on fixing or otherwise dealing with the indefinite kernel). One advantage is that it is rather easy to interpret.

In contrast, the parameter λ_n of $\hat{D}$ is more difficult to interpret. But it has the advantage that it may be estimated for a single matrix as well as its average for a set of matrices. Hence, we want to determine whether the magnitude of this eigenvalue affects model performance. We expect an influence that depends on

Table 2 Comparing the true proportion of non-CNSD matrices p determined by brute force (true) with the mean of p estimated by sampling (estimate). The table only presents results for permutations with $m = 4$ elements and set sizes n. The number of sets evaluated by brute force is $n_s = n_p! \ / \ (n! \ (n_p - n)!)$ with $n_p = m!$.

n	distance	n_s	true	estimate
5	Insert	42,504	0.047	0.048
6	Insert	134,596	0.221	0.219
7	Insert	346,104	0.530	0.529
8	Insert	735,471	0.827	0.825
5	Interchange	42,504	0.106	0.105
6	Interchange	134,596	0.465	0.463
7	Interchange	346,104	0.833	0.834
8	Interchange	735,471	0.978	0.978
5	Levenshtein	42,504	0.087	0.087
6	Levenshtein	134,596	0.293	0.293
7	Levenshtein	346,104	0.591	0.589
8	Levenshtein	735,471	0.847	0.846
5	LCStr	42,504	0.002	0.002
6	LCStr	134,596	0.007	0.007
7	LCStr	346,104	0.026	0.026
8	LCStr	735,471	0.093	0.092

the choice of model. Consider, e.g., a Gaussian process regression model, as e.g., described by Forrester et al. (2008). The model may be able to mitigate the problematic eigenvalue by assigning larger θ values to the kernel $k(x, x') = \exp(-\theta d(x, x'))$. For very large λ_n and thus very large θ, this will lead to kernel matrices that approximate the unit matrix, which is positive definite. A model with a unit kernel matrix would be able to reproduce the training data, but would predict the process mean for most other data points. Hence, we examine Gaussian process regression models, since they provide a transparent and interpretable test case (but similar experiments could easily be made with support vector regression).

An experimental test has to consider the potential bias of the used data set. We need to be able to reasonably assume that differences in performance are actually due to the properties of employed distance or kernel (i.e., a kernel performs poorly because the corresponding λ_n of $\hat{D}$ is high) rather than properties of the data set (i.e., a kernel performs poorly because it does not fit well to the ground truth of the data set). To that end, we suggest that observations in a test data set are derived from the same distances that are used in the model.

Hence, we randomly created data sets X of size n with permutations of dimension m, similarly to the random sampling performed earlier. Then, we created training observations by evaluating the distance of each permutation in X to a reference permutation $x_{ref} = 1, ..., m$, i.e., $y = d(x, x_{ref})$. A Gaussian process model was then trained with this data, largely following the descriptions in Forrester et al. (2008). The model was trained with the kernel $k(x, x') = \exp(-\theta d(x, x'))$, and the model parameters (e.g., θ) were determined by maximum likelihood estimation, via the locally biased version of the DIviding RECTangles (DIRECT) algorithm (Gablonsky and Kelley 2001) with 1,000 likelihood evaluations. For each test, the distance chosen to produce the observations y and the distance chosen in the kernel were identical.

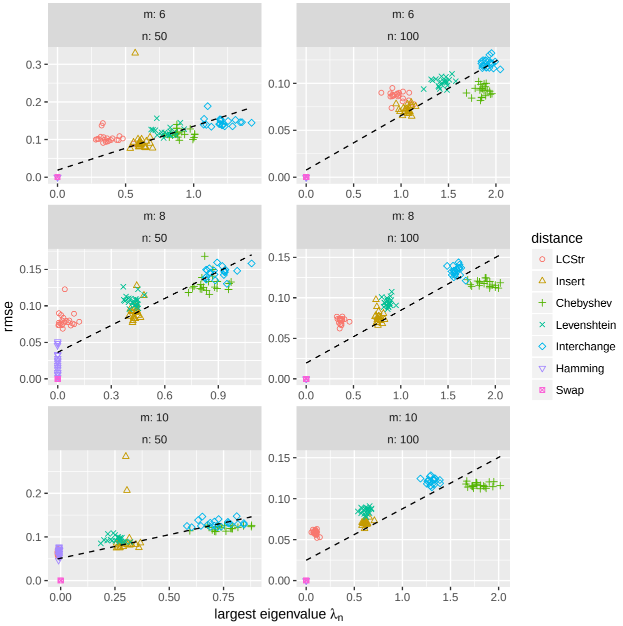

Fig. 4 RMSEs of a Gaussian process regression model plotted against the eigenvalue λ_n of $\hat{D}$ that is critical to definiteness of the underlying distance matrices. The denoted distance measures are used in the underlying data set as well as the model itself. The dashed line depicts a linear trend.

The Root Mean Square Error (RMSE) of the model was evaluated on 1000 randomly chosen permutations. The resulting RMSE values for each training set X, as well as the corresponding eigenvalue λ_n of $\hat{D}$ are shown in Fig. 4. The graphic shows a trend that confirms our expectation. It seems that distances associated to larger λ_n tend to produce larger errors.

6 Summary and Outlook

The focus of this study was the definiteness of kernel and distance measures. Definiteness is a main requirement for modeling techniques like Gaussian processes or SVMs. Distance-based kernels with unknown definiteness may be promising choices for certain parameter spaces. Yet, their definiteness may be very hard to prove or enforce.

It was found, that empirical approaches (based on sampling or optimization) may help to assess definiteness of the respective function. In detail, two research questions were investigated:

Q_1 **Discovery:** Is there an efficient, empirical approach to determine the definiteness of kernel functions based on arbitrary distance measures?

Q_2 **Measurability:** If Q_1 can be answered affirmatively, can we quantify to what extent a lack of definiteness is problematic?

Two empirical approaches were suggested towards that end. The first approach (A_1) samples from the space of solution sets, and determines whether a set is found which leads to an indefinite distance or kernel matrix. If indefinite matrices are rare, a directed search with an EA is more successful (A_2). The EA maximizes the largest eigenvalue of a transformed distance matrix $\hat{D}$, respectively minimizes the smallest eigenvalue of a kernel matrix. Hence, the EA searches for sets that yield indefinite matrices.

As a proof-of-concept, the approaches were applied to distance measures for permutations. It was shown that five problematic distance measures could be identified: Longest Common Substring (LCStr), Insert, Chebyshev, Levenshtein, and Interchange distance. Information known from literature (regarding indefiniteness of the respective kernel function) could be rediscovered by the empirical approaches.

The optimization approach was successful, as it was able to outperform the sampling approach in discovering sets with indefinite kernel matrices. Still, the results also indicated that the choice of variation operators in the optimization algorithm does introduce bias. Hence, the respective results do not allow a conclusion about the impact of a lack of definiteness of the respective sets/matrices. Still, the success of the EA indicates that the fitness landscape posed by the largest eigenvalue is not excessively rugged and has an exploitable structure. This suggests that the largest eigenvalue is a good indicator of how far a certain solution set (and the respective distance or kernel matrix) is from becoming indefinite. In an additional set of experiments, we further verified that increasing the largest eigenvalue can in fact be linked to a decrease in model quality. This results into the following responses to the posed research questions:

R1 **Discovery:** Sampling from the space of potential candidate solution sets allows identifying problems with definiteness, by identifying solution sets that lead to non-CNSD distance matrices (A_1). Where such situations are rare (hence more likely to be missed by the sampling), an optimization approach may be more successful (A_2). While neither approach A_1 nor A_2 are able to prove definiteness, both are able to disprove it. If no negative results are found it is reasonable to assume that using the respective distance/kernel function is feasible.

R2 **Measurability:** The sampling approach (A_1) yields a proportion of potentially non-CNSD matrices, which in turn yields an estimate of how problematic a distance measure is. In a similar way, yet potentially biased by the choice of optimization algorithm, the number of evaluations required by the optimization approach gives a similar estimate. In addition, the success of the optimization approach (A_2) suggests that the respective largest eigenvalue is an indicator of how close certain sets (and respective distance or kernel matrices) are to becoming indefinite. Additional experiments showed how this eigenvalue could be linked to model performance.

For future research, it may be of interest to allow the EA to change the set size. Clearly, one issue would be that enlarging the sets may quickly lead to a trivial solution, since larger sets naturally lead to larger λ_n of $\hat{D}$. Hence, there is a trade-off between the largest eigenvalue and the set size. A multi-objective EA (e.g. NSGA-II (Deb et al. 2002) or SMS-EMOA (Beume et al. 2007)) may be used to handle this issue by simultaneously maximizing λ_n and minimizing the set size n.

Finally, the herein described kernels and distances are not the full story. For other kernels, the relation between distance measure and kernel function may not be as straightforward. Parameters of the distance measure or the kernel function could complicate the situation. It may be necessary to adapt the proposed method to, e.g., include parameters in the sampling and optimization procedures.

Compliance with ethical standards

Conflicts of Interest The authors declare that they have no conflict of interest.

References

Bader, D. A., Moret, B. M., Warnow, T., Wyman, S. K., Yan, M., Tang, J., Siepel, A. C., & Caprara, A. (2004). Genome rearrangements analysis under parsimony and other phylogenetic algorithms (grappa) 2.0. Online: https://www.cs.unm.edu/ moret/GRAPPA/. accessed: 16.11.2016.

Bartz-Beielstein, T. & Zaefferer, M. (2017). Model-based methods for continuous and discrete global optimization. *Applied Soft Computing*, 55:154 – 167.

Berg, C., Christensen, J. P. R., & Ressel, P. (1984). *Harmonic Analysis on Semigroups*, volume 100 of *Graduate Texts in Mathematics*. Springer New York.

Beume, N., Naujoks, B., & Emmerich, M. (2007). SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669.

Boytsov, L. (2011). Indexing methods for approximate dictionary searching: Comparative analysis. *J. Exp. Algorithmics*, 16:1–91.

Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167.

Camastra, F. & Vinciarelli, A. (2008). *Machine Learning for Audio, Image and Video Analysis: Theory and Applications*. Advanced information and knowledge processing. Springer, London.

Campos, V., Laguna, M., & Martí, R. (2005). Context-independent scatter and tabu search for permutation problems. *INFORMS Journal on Computing*, 17(1):111–122.

Camps-Valls, G., Martín-Guerrero, J. D., Rojo-Álvarez, J. L., & Soria-Olivas, E. (2004). Fuzzy sigmoid kernel for support vector classifiers. *Neurocomputing*, 62:501–506.

Chen, Y., Gupta, M. R., & Recht, B. (2009). Learning kernels from indefinite similarities. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, (pp. 145–152), New York, NY, USA. ACM.

Constantine, G. (1985). Lower bounds on the spectra of symmetric matrices with nonnegative entries. *Linear Algebra and its Applications*, 65:171–178.

Cortes, C., Haffner, P., & Mohri, M. (2004). Rational kernels: Theory and algorithms. *J. Mach. Learn. Res.*, 5:1035–1062.

Curriero, F. (2006). On the use of non-euclidean distance measures in geostatistics. *Mathematical Geology*, 38(8):907–926.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

Deza, M. & Huang, T. (1998). Metrics on permutations, a survey. *Journal of Combinatorics, Information and System Sciences*, 23(1-4):173–185.

Eiben, A. E. & Smith, J. E. (2003). *Introduction to Evolutionary Computing*. Springer Berlin Heidelberg.

Feller, W. (1971). *An Introduction to Probability Theory and Its Applications, Volume 2*. JOHN WILEY & SONS INC.

Forrester, A., Sobester, A., & Keane, A. (2008). *Engineering Design via Surrogate Modelling*. Wiley.

Gablonsky, J. & Kelley, C. (2001). A locally-biased form of the direct algorithm. *Journal of Global Optimization*, 21(1):27–37.

Gärtner, T., Lloyd, J., & Flach, P. (2003). Kernels for structured data. In Matwin, S. & Sammut, C., editors, *Inductive Logic Programming*, volume 2583 of *Lecture Notes in Computer Science*, pages 66–83. Springer Berlin Heidelberg.

Gärtner, T., Lloyd, J., & Flach, P. (2004). Kernels and distances for structured data. *Machine Learning*, 57(3):205–232.

Haussler, D. (1999). Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10, Department of Computer Science, University of California at Santa Cruz.

Hirschberg, D. S. (1975). A linear space algorithm for computing maximal common subsequences. *Commun. ACM*, 18(6):341–343.

Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. In *Proc. of LION-5*, (pp. 507–523).

Ikramov, K. & Savel'eva, N. (2000). Conditionally definite matrices. *Journal of Mathematical Sciences*, 98(1):1–50.

Jiao, Y. & Vert, J.-P. (2015). The Kendall and Mallows kernels for permutations. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, (pp. 1935–1944).

Kendall, M. & Gibbons, J. (1990). *Rank Correlation Methods*. Oxford University Press.

Lee, C. (1958). Some properties of nonbinary error-correcting codes. *IRE Transactions on Information Theory*, 4(2):77–82.

Li, H. & Jiang, T. (2004). A class of edit kernels for svms to predict translation initiation sites in eukaryotic mrnas. In *Proceedings of the Eighth Annual International Conference on Resaerch in Computational Molecular Biology*, RECOMB '04, (pp. 262–271), New York, NY, USA. ACM.

Loosli, G., Canu, S., & Ong, C. (2015). Learning SVM in Krein spaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(6):1204–1216.

Marteau, P.-F. & Gibet, S. (2014). On recursive edit distance kernels with application to time series classification. *IEEE Transactions on Neural Networks and Learning Systems*, PP(99):1–1.

Moraglio, A. & Kattan, A. (2011). Geometric generalisation of surrogate model based optimisation to combinatorial spaces. In *Proceedings of the 11th European Conference on Evolutionary Computation in Combinatorial Optimization*, EvoCOP'11, (pp. 142–154), Berlin, Heidelberg, Germany. Springer.

Motwani, R. & Raghavan, P. (1995). *Randomized Algorithms*. Cambridge University Press.

Murphy, K. P. (2012). *Machine Learning*. MIT Press Ltd.

Ong, C. S., Mary, X., Canu, S., & Smola, A. J. (2004). Learning with non-positive kernels. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, (pp. 81–88), New York, NY, USA. ACM.

Pawlik, M. & Augsten, N. (2015). Efficient computation of the tree edit distance. *ACM Transactions on Database Systems*, 40(1):1–40.

Pawlik, M. & Augsten, N. (2016). Tree edit distance: Robust and memory-efficient. *Information Systems*, 56:157–173.

Rasmussen, C. E. & Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. The MIT Press.

Reeves, C. R. (1999). Landscapes, operators and heuristic search. *Annals of Operations Research*, 86(0):473–490.

Schiavinotto, T. & Stützle, T. (2007). A review of metrics on permutations for search landscape analysis. *Computers & Operations Research*, 34(10):3143–3153.

Schleif, F.-M. & Tino, P. (2015). Indefinite proximity learning: A review. *Neural Computation*, 27(10):2039–2096.

Schölkopf, B. (2001). The kernel trick for distances. In Leen, T. K., Dietterich, T. G., & Tresp, V., editors, *Advances in Neural Information Processing Systems 13*, pages 301–307. MIT Press.

Sevaux, M. & Sörensen, K. (2005). Permutation distance measures for memetic algorithms with population management. In *Proceedings of 6th Metaheuristics International Conference (MIC'05)*, (pp. 832–838). University of

Vienna.

Singhal, A. (2001). Modern information retrieval: a brief overview. *IEEE Bulletin on Data Engineering*, 24(4):35–43.

Smola, A. J., Ovári, Z. L., & Williamson, R. C. (2000). Regularization with dot-product kernels. In *Advances in Neural Information Processing Systems 13, Proceedings*, (pp. 308–314). MIT Press.

van der Loo, M. P. (2014). The stringdist package for approximate string matching. *The R Journal*, 6(1):111–122.

Vapnik, V. N. (1998). *Statistical Learning Theory*, volume 1. Wiley New York.

Voutchkov, I., Keane, A., Bhaskar, A., & Olsen, T. M. (2005). Weld sequence optimization: The use of surrogate models for solving sequential combinatorial problems. *Computer Methods in Applied Mechanics and Engineering*, 194(30-33):3535–3551.

Wagner, R. A. & Fischer, M. J. (1974). The string-to-string correction problem. *J. ACM*, 21(1):168–173.

Wu, G., Chang, E. Y., & Zhang, Z. (2005). An analysis of transformation on non-positive semidefinite similarity matrix for kernel machines. In *Proceedings of the 22nd International Conference on Machine Learning*.

Zaefferer, M. & Bartz-Beielstein, T. (2016). Efficient global optimization with indefinite kernels. In *Parallel Problem Solving from Nature–PPSN XIV*, (pp. 69–79). Springer.

Zaefferer, M., Stork, J., & Bartz-Beielstein, T. (2014a). Distance measures for permutations in combinatorial efficient global optimization. In Bartz-Beielstein, T., Branke, J., Filipič, B., & Smith, J., editors, *Parallel Problem Solving from Nature–PPSN XIII*, (pp. 373–383), Cham, Switzerland. Springer.

Zaefferer, M., Stork, J., Friese, M., Fischbach, A., Naujoks, B., & Bartz-Beielstein, T. (2014b). Efficient global optimization for combinatorial problems. In *Proceedings of the 2014 Conference on Genetic and Evolutionary Computation*, GECCO '14, (pp. 871–878), New York, NY, USA. ACM.

Appendix A: Distance Measures for Permutations

In the following, we describe the distance measures employed in the experiments.

– The Levenshtein distance is an edit distance measure:

$$d_{Lev}(\pi, \pi') = edits_{\pi \to \pi'}$$

Here, $edits_{\pi \to \pi'}$ is the minimal number of deletions, insertions, or substitutions required to transform one string (or here: permutation) π into another string π'. The implementation is based on Wagner and Fischer (1974).

– Swaps are transpositions of two adjacent elements. The Swap distance (also: Kendall's Tau (Kendall and Gibbons 1990; Sevaux and Sörensen 2005) or Precedence distance (Schiavinotto and Stützle 2007)) counts the minimum number of swaps required to transform one permutation into another. For permutations, it is (Sevaux and Sörensen 2005):

$$d_{Swa}(\pi, \pi') = \sum_{i=1}^{m} \sum_{j=1}^{m} z_{ij} \quad \text{with}$$

$$z_{ij} = \begin{cases} 1 & \text{if } \pi_i < \pi_j \text{ and } \pi'_i > \pi'_j, \\ 0 & \text{otherwise.} \end{cases}$$

– An interchange operation is the transposition of two arbitrary elements. Respectively, the Interchange (also: Cayley) distance counts the minimum number of interchanges ($interchanges_{\pi \to \pi'}$) required to transform one permutation into another (Schiavinotto and Stützle 2007):

$$d_{Int}(\pi, \pi') = interchanges_{\pi \to \pi'}$$

– The Insert distance is based on the longest common subsequence $LCSeq(\pi, \pi')$. The longest common subsequence is the largest number of elements that follow each other in both permutations, with interruptions. The corresponding distance is

$$d_{Ins}(\pi, \pi') = m - LCSeq(\pi, \pi').$$

We use the algorithm described by Hirschberg (1975). The name is due to its interpretation as an edit distance measure. The corresponding edit operation is a combination of insertion and deletion. A single element is moved from one position (delete) to a new position (insert). It is also called Ulam's distance (Schiavinotto and Stützle 2007).

– The Longest Common Substring distance is based on the largest number of elements that follow each other in both permutations, without interruption. Unlike the longest common subsequence all elements have to be adjacent.

If $LCStr(\pi, \pi')$ is the length of the longest common string, the distance is

$$d_{LCStr}(\pi, \pi') = m - LCStr(\pi, \pi').$$

– The R-distance (Campos et al. 2005; Sevaux and Sörensen 2005) counts the number of times that one element follows another in one permutation, but not in the other. It is identical with the uni-directional adjacency distance (Reeves 1999). It is computed by

$$d_R(\pi, \pi') = \sum_{i=1}^{m-1} y_i \quad \text{with}$$

$$y_i = \begin{cases} 0 & \text{if } \exists j : \pi_i = \pi'_j \text{ and } \pi_{i+1} = \pi'_{j+1}, \\ 1 & \text{otherwise.} \end{cases}$$

– The (bi-directional) Adjacency distance (Reeves 1999; Schiavinotto and Stützle 2007) counts the number of times two elements are neighbors in one, but not in the other permutation. Unlike R-distance (uni-directional), the order of the two elements does not matter. It is computed by

$$d_{Adj}(\pi, \pi') = \sum_{i=1}^{m-1} y_i \quad \text{with}$$

$$y_i = \begin{cases} 0 & \text{if } \exists j : \pi_i = \pi'_j \text{ and } \pi_{i+1} \in \{\pi'_{j+1}, \pi'_{j-1}\}, \\ 1 & \text{otherwise.} \end{cases}$$

– The Position distance (Schiavinotto and Stützle 2007) is identical with the Deviation distance or Spearman's footrule (Sevaux and Sörensen 2005),

$$d_{\mathrm{Pos}}(\pi, \pi') = \textstyle\sum_{k=1}^{m} |i - j| \quad \text{where} \quad \pi_i = \pi'_j = k \ .$$

– The non-metric Squared Position distance is Spearman's rank correlation coefficient (Sevaux and Sörensen 2005). In contrast to the Position distance, the term $|i - j|$ is replaced by $(i - j)^2$.

– The Hamming distance or Exact Match distance simply counts the number of unequal elements in two permutations, i.e.,

$$d_{Ham}(\pi, \pi') = \textstyle\sum_{i=1}^{m} a_i, \quad \text{where} \quad a_i = \begin{cases} 0 & \text{if } \pi_i = \pi'_i, \\ 1 & \text{otherwise.} \end{cases}$$

- The Euclidean distance is
$$d_{Euc}(\pi, \pi') = \sqrt{\sum_{i=1}^{m}(\pi_i - \pi'_i)^2} \; .$$
- The Manhattan distance (A-Distance, cf. (Sevaux and Sörensen 2005; Campos et al. 2005)) is
$$d_{Man}(\pi, \pi') = \sum_{i=1}^{m} |\pi_i - \pi'_i| \; .$$
- The Chebyshev distance is
$$d_{Che}(\pi, \pi') = \max_{1 \leq i \leq m} (|\pi_i - \pi'_i|) \; .$$
- For permutations, the Lee distance (Lee 1958; Deza and Huang 1998) is
$$d_{Lee}(\pi, \pi') = \sum_{i=1}^{m} \min(|\pi_i - \pi'_i|, m - |\pi_i - \pi'_i|) \; .$$
- The non-metric Cosine distance is based on the dot product of two permutations. It is derived from the cosine similarity (Singhal 2001) of two vectors:
$$d_{Cos}(\pi, \pi') = 1 - \frac{\pi \cdot \pi'}{||\pi|| \; ||\pi'||}.$$
- The Lexicographic distance regards the lexicographic ordering of permutations. If the position of a permutation π in the lexicographic ordering of all permutations with fixed m is $L(\pi)$, then the Lexicographic distance metric is
$$d_{Lex}(\pi, \pi') = |L(\pi) - L(\pi')|.$$

Appendix B: Minimal Examples for Indefinite Sets

To showcase the usefulness of the proposed methods, this section lists small example data sets and the respective indefinite distance matrices. Besides the standard permutation distances we also tested:

- **Signed permutations, reversal distance:** Permutations where each element has a sign are referred to as signed permutations. An application example for signed permutations is, e.g., weld path optimization (Voutchkov et al. 2005). The reversal distance counts the number of reversals required to transform one permutation into another. We used the non-cyclic reversal distance provided in the GRAPPA library version 2.0 (Bader et al. 2004).
- **Labeled trees, tree edit distance:** Trees in general are widely applied as solution representation, e.g., in Genetic Programming. In this study, we considered labeled trees. The tree edit distance counts the number node insertions, deletions or relabels. We used the efficient implementation in the APTED 0.1.1 library (Pawlik and Augsten 2015, 2016). The labeled trees will be denoted with the bracket notation: curly brackets indicate the tree structure, letters indicate labels (internal and terminal nodes).
- **Strings, Optimal String Alignment distance (OSA):** The OSA is an non-metric edit distance that counts insertions, deletions, substitutions and transpositions of characters. Each substring can be edited no more than once. It is also called the restricted Damerau-Levenshtein distance (Boytsov 2011). We used the implementation in the `stringdist` R package (van der Loo 2014).
- **Strings, Jaro-Winkler distance:** The Jaro Winkler distance is based on the number of matching characters in two strings as well as the number of transpositions required to bring all matches in the same order. We used the implementation in the `stringdist` R package (van der Loo 2014).

The respective results are listed in Table 3. All of the listed distance measures are shown to be non-CNSD.

Table 3 Minimal examples for indefinite distance matrices. The matrix in the table is the actual distance matrix, while the eigenvalue refers to the transformed matrix $\hat{D}$ derived from equation (3). The lower triangular matrix is omitted due to symmetry.

Permutations, Insert, $n = 5$, $m = 4$, $\lambda_n \approx 0.090$						
i	x_i	$d_{i,1}$	$d_{i,2}$	$d_{i,3}$	$d_{i,4}$	$d_{i,5}$
1	$\{1, 2, 3, 4\}$	0	1/3	1/3	2/3	1/3
2	$\{1, 3, 4, 2\}$		0	2/3	1/3	2/3
3	$\{2, 3, 4, 1\}$			0	1/3	2/3
4	$\{3, 4, 1, 2\}$				0	1/3
5	$\{4, 1, 2, 3\}$					0
Permutations, Interchange, $n = 5$, $m = 4$, $\lambda_n \approx 0.090$						
1	$\{1, 2, 3, 4\}$	0	1/3	1/3	2/3	1/3
2	$\{1, 2, 4, 3\}$		0	2/3	1/3	2/3
3	$\{1, 3, 2, 4\}$			0	1/3	2/3
4	$\{1, 3, 4, 2\}$				0	1/3
5	$\{1, 4, 3, 2\}$					0
Permutations, Levenshtein, $n = 5$, $m = 4$, $\lambda_n \approx 0.135$						
1	$\{1, 2, 4, 3\}$	0	1	1/2	1/2	1
2	$\{2, 3, 1, 4\}$		0	1/2	1/2	1
3	$\{2, 4, 3, 1\}$			0	1	1/2
4	$\{3, 1, 2, 4\}$				0	1/2
5	$\{3, 4, 2, 1\}$					0
Permutations, LCStr, $n = 5$, $m = 4$, $\lambda_n \approx 0.023$						
1	$\{1, 3, 2, 4\}$	0	2/3	1/3	1/3	2/3
2	$\{2, 4, 1, 3\}$		0	1/3	1/3	2/3
3	$\{3, 2, 4, 1\}$			0	2/3	1
4	$\{4, 1, 3, 2\}$				0	2/3
5	$\{4, 2, 1, 3\}$					0
Permutations, Chebyshev, $n = 5$, $m = 5$, $\lambda_n \approx 0.034$						
1	$\{1, 5, 3, 4, 2\}$	0	1/4	3/4	3/4	1
2	$\{2, 5, 3, 4, 1\}$		0	1	1	3/4
3	$\{4, 2, 3, 1, 5\}$			0	2/4	1/4
4	$\{4, 3, 1, 2, 5\}$				0	1/4
5	$\{5, 3, 2, 1, 4\}$					0
Sign. Permutations, Reversal, $n = 5$, $m = 5$, $\lambda_n \approx 0.016$						
1	{ 4, 5, -1, -2, -3}	0	4/6	5/6	3/6	2/6
2	{ 2, 1, 3, -4, -5}		0	2/6	3/6	5/6
3	{-2, 1, 3, 5, 4}			0	5/6	3/6
4	{ 4, -2, 3, 1, -5}				0	2/6
5	{ 4, -2, 1, -5, -3}					0
Labeled Trees, Edit dist., $n = 5$, $\lambda_n \approx 0.026$						
1	$\{b\{c\{b\}\}\}$	0	2	1	3	1
2	$\{b\}$		0	1	3	1
3	$\{b\{c\}\}$			0	2	2
4	$\{a\{c\}\{a\}\}$				0	3
5	$\{c\{b\}\}$					0
Strings, Optimal String Alignment, $n = 5$, $\lambda_n \approx 0.102$						
1	abc	0	1	2	3	1
2	acc		0	3	2	2
3	cba			0	1	2
4	caa				0	2
5	bac					0
Strings, Jaro-Winkler, $n = 4$, $\lambda_n \approx 0.046$						
1	bbbb	0	1	1/6	3/6	
2	aaaa		0	3/6	1/6	
3	bbba			0	3/6	
4	aaab				0	