DiffPaSS – High-performance differentiable pairing of protein sequences using soft scores

Umberto Lupo[1,2,†], Damiano Sgarbossa[1,2], Martina Milighetti[3,4], Anne-Florence Bitbol[1,2,†]

1 Institute of Bioengineering, School of Life Sciences, École Polytechnique Fédérale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland
2 SIB Swiss Institute of Bioinformatics, CH-1015 Lausanne, Switzerland
3 Division of Infection and Immunity, University College London, London, United Kingdom
4 Cancer Institute, University College London, London, United Kingdom
[†] Emails: umberto.lupo@epfl.ch, anne-florence.bitbol@epfl.ch

Abstract

Identifying interacting partners from two sets of protein sequences has important applications in computational biology. Interacting partners share similarities across species due to their common evolutionary history, and feature correlations in amino acid usage due to the need to maintain complementary interaction interfaces. Thus, the problem of finding interacting pairs can be formulated as searching for a pairing of sequences that maximizes a sequence similarity or a coevolution score. Several methods have been developed to address this problem, applying different approximate optimization methods to different scores. We introduce DiffPaSS, a framework for flexible, fast, and hyperparameter-free optimization for pairing interacting biological sequences, which can be applied to a wide variety of scores. We apply it to a benchmark prokaryotic dataset, using mutual information and neighbor graph alignment scores. DiffPaSS outperforms existing algorithms for optimizing the same scores. We demonstrate the usefulness of our paired alignments for the prediction of protein complex structure. DiffPaSS does not require sequences to be aligned, and we also apply it to non-aligned sequences from T cell receptors.

Introduction

Identifying which proteins interact together, using their sequence data alone, is an important and combinatorially difficult task. Mapping the network of protein-protein interactions, and predicting the three-dimensional structures of individual protein complexes, often requires determining which sequences are functional interaction partners among the paralogous proteins of two families. In the case of specific one-to-one interactons, this problem can be formulated as looking for a permutation of the sequences one family with respect to those of the other within each species. Sequence similarity-based scores [1–10] and coevolution-based ones [11–14], as well as a method combining both ingredients [15] have been proposed to tackle this problem.

Sequence similarity is employed because when two proteins interact in one species, and possess close homologs in another species, then these homologs are likely to also interact. More generally, interacting protein families share a similar evolutionary history [16–18], leading to sequence similarity-based pairing methods [1–10]. The use of neighbor graph alignment [8] or of orthology determined by closest reciprocal hits for pairing interaction partners [19–22] also relies on this idea.

The idea underlying coevolution-based methods for pairing interacting sequences is that amino acids that are in contact at the interface between two interaction partners need to maintain physicochemical complementarity through evolution, which gives rise to correlations in amino-acid usage between interacting proteins [11–14, 23]. Mutual information [24] and pairwise maximum entropy models [23, 25–28] can reveal such coevolution both within a protein sequence and between the sequences of interacting partners. Additional correlations come from the shared evolutionary history of

interacting partners [29, 30]. Thus, permutations maximizing coevolution scores are expected to encode correct interactions [12, 14, 15]. Coevolution-based approaches require large and diverse multiple sequence alignments (MSAs) to perform well, which limits their applicability. More recently, scores coming from protein language models have also been proposed [31, 32]. In particular, we proposed a method that outperforms traditional coevolution-based methods for shallow MSAs, comprising few sequences. However, this method is computationally intensive, and memory requirements limit its applicability to large MSAs.

We present DiffPaSS ("**Diff**erentiable **Pa**iring using **S**oft **S**cores"), a family of flexible, fast and hyperparameter-free algorithms for pairing interacting sequences among the paralogs of two protein families. DiffPaSS optimizes smooth extensions of coevolution or similarity scores to "soft" permutations of the input sequences, using gradient methods. It can be used to optimize any score, including coevolution scores and sequence similarity scores. Strong optima are reached thanks to a novel bootstrap technique, motivated by heuristic insights into this smooth optimization process. When using inter-chain mutual information (MI) between two MSAs as the score to be maximised, DiffPaSS outperforms existing coevolution- and sequence similarity-based pairing methods on difficult benchmarks composed of small MSAs from ubiquitous interacting prokaryotic systems. Thanks to its rapidity, DiffPaSS can easily produce paired alignments that can be used as input to AlphaFold-Multimer [21], in order to predict the three-dimensional structure of protein complexes. We show promising results in this direction, on a selection of eukaryotic complexes. DiffPaSS is a general method that is not restricted to coevolution scores. In particular, it can be used to pair sequences by aligning their similarity graphs. We demonstrate that DiffPaSS outperforms a Monte Carlo simulated annealing method for graph alignment-based pairing of interacting partners on our benchmark prokaryotic data. Importantly, DiffPaSS graph alignment can be used even when reliable MSAs are not available. We show that it outperforms Monte Carlo graph alignment on the problem of pairing non-aligned sequences of T-cell receptor (TCR) CDR3α and CDR3β loops. A `PyTorch` implementation and installable Python package are available at https://github.com/Bitbol-Lab/DiffPaSS.

Methods

Problem and general approach

Pairing interacting protein sequences. Consider two protein families A and B that interact together, and the associated MSAs $\mathcal{M}_\mathrm{A}$ and $\mathcal{M}_\mathrm{B}$. Each of them is partitioned into K species, and we denote by N_k the number of sequences in species k, with $k = 1, \ldots, K$. In practice, the number of members of family A and of family B in a species is often different. In this case, N_k is the largest of these two numbers, and we add sequences made entirely of gap symbols to the MSA with fewer sequences in that species, as in Ref. [32].

Our goal is to pair the proteins of family A with their interacting partners from family B within each species. We assume for simplicity that interactions are one-to-one within each species. This is the first and main problem addressed by our method, and is commonly known as the paralog matching problem, as the within-species protein variants are often paralogs.

Our method also extends to related matching problems involving non-aligned sequences. This will be discussed below, with an application to variable regions of T-cell receptors, where alignment is challenging. Then, $\mathcal{M}_\mathrm{A}$ and $\mathcal{M}_\mathrm{B}$ are ordered collections of non-aligned amino-acid sequences instead of MSAs.

Formalization. Let $\mathscr{S}$ be a score function of two MSAs (or ordered collections of non-aligned sequences) $\mathcal{M}_\mathrm{A}$ and $\mathcal{M}_\mathrm{B}$, which is sensitive to the relative ordering of the rows of $\mathcal{M}_\mathrm{A}$ with respect to those of $\mathcal{M}_\mathrm{B}$. Our matching problems can be formalized as searching for a permutation π of the rows of $\mathcal{M}_\mathrm{A}$ which maximises $\mathscr{S}(\pi(\mathcal{M}_\mathrm{A}), \mathcal{M}_\mathrm{B})$, denoted by $\mathscr{S}(\pi)$ for brevity. The permutation π should operate within each species and not across them, since interactions occur within a species. There are $\prod_{k=1}^{K} N_k!$ permutations satisfying this constraint, which usually renders unfeasible the brute-force approach of scoring all of them and picking the one with the largest score.

General approach. Optimizing a score $\mathscr{S}$ across permutations is a discrete problem, but we propose an approximate differentiable formulation. Briefly, we first construct a differentiable extension $\hat{\mathscr{S}}$ of $\mathscr{S}$ from permutation matrices to a larger space of matrices. This larger space comprises square matrices with non-negative entries and whose all rows and columns (approximately) sum to 1. In what follows, we refer to such matrices as "soft permutations", and to true permutation matrices as "hard permutations". Using real square matrices, referred to as "parameterization matrices" X, a method [33] based on the Sinkhorn operator [34] allows to smoothly navigate the space of soft permutations while keeping track of the "nearest" hard permutations (for more details, see the Supplementary material). When this navigation is guided by gradient ascent applied to the extended score function $\hat{\mathscr{S}}$, these hard permutations provide candidate solutions to the original problem.

In general, however, this differentiable optimization problem for $\hat{\mathscr{S}}$ may have several local optima. Besides, depending on the precise way in which the discrete score $\mathscr{S}$ is extended to a differentiable score $\hat{\mathscr{S}}$ for soft permutations, optimal soft permutations for $\hat{\mathscr{S}}$ may be too distant from any hard permutation to approximate a well-scoring hard permutation for the original problem. Indeed, we found empirically, for a variety of scores, extensions, and random initializations, that naïvely applying the procedure above often yielded hard permutations with sub-optimal scores, even after several gradient steps. Nevertheless, we empirically found the following to be true for a wide class of scores and extensions thereof:

1. When gradient ascent is initialized so that the initial soft permutation for each species is "as soft as possible" – i.e., it is a square matrix with all entries equal to the same positive value, and normalized rows and columns – the first gradient step leads to a nearby *hard* permutation with significantly higher score than given by random expectation. This point is illustrated in Fig. S1.

2. Pairing performance increases if some sequences are correctly paired and excluded from gradient optimization, while being included in the calculation of $\mathscr{S}$ and $\hat{\mathscr{S}}$. As the size of this fixed context is increased, pairing performance for the remaining sequences increases. This point is consistent with previous approaches using other methods [12, 14, 15, 32].

Together, these considerations motivated us to develop a bootstrapped approach to differentiable pairing, that we call DiffPaSS. Briefly, after a single gradient step using the procedure outlined above and the initialization described in point 1 above, a number n of pairings from the corresponding hard permutation is sampled uniformly at random, and used as fixed context in another run of gradient optimization using a single gradient step. The number n is gradually increased from 1 to the size of the collections of sequences to be paired. Fig. 1 illustrates the first two iterations of this method. See the Supplementary material for more details.

Using a single gradient step starting from our special initializations makes each step of the DiffPaSS bootstrap independent of gradient optimization hyperparameters such as learning rate and regularizaton strength. It also makes it independent of the two hyperparameters needed to define the (truncated) Sinkhorn operator, see Supplementary material.

Mutual-information–based scores for MSAs

When $\mathcal{M}_\mathrm{A}$ and $\mathcal{M}_\mathrm{B}$ are MSAs, denoting by $\mathcal{M}_{\mathrm{A},i}$ the i-th column of $\mathcal{M}_\mathrm{A}$ (and analogously for $\mathcal{M}_\mathrm{B}$), we define the *inter-chain mutual information* score $\mathscr{S}_\mathrm{MI}(\mathcal{M}_\mathrm{A}, \mathcal{M}_\mathrm{B})$ by summing the mutual information estimates of all pairs of columns composed of one column in $\mathcal{M}_\mathrm{A}$ and one in $\mathcal{M}_\mathrm{B}$ [14]:

$$\mathscr{S}_\mathrm{MI}(\mathcal{M}_\mathrm{A}, \mathcal{M}_\mathrm{B}) = \sum_{i,j} \mathrm{I}(\mathcal{M}_{\mathrm{A},i}; \mathcal{M}_{\mathrm{B},j}) = \sum_{i,j} [\mathrm{H}(\mathcal{M}_{\mathrm{A},i}) + \mathrm{H}(\mathcal{M}_{\mathrm{B},j}) - \mathrm{H}(\mathcal{M}_{\mathrm{A},i}, \mathcal{M}_{\mathrm{B},j})] \,, \tag{1}$$

where $\mathrm{I}(\cdot;\cdot)$ denotes the mutual information between two random variables, and H denotes the Shannon entropy. In practice, we replace these information quantities by their plug-in estimates, i.e. we use observed frequencies instead of probabilities (see [14]). Since, for any i and permutation π, $\mathrm{H}(\mathcal{M}_{\mathrm{A},i}) = \mathrm{H}(\pi(\mathcal{M}_\mathrm{A})_i)$, maximising $\mathscr{S}_\mathrm{MI}(\pi)$ over permutations π is equivalent to minimizing the *inter-chain two-body entropy loss* $\mathscr{L}_\mathrm{2BE}(\pi)$ over permutations, with $\mathscr{L}_\mathrm{2BE}(\pi)$ defined as

$$\mathscr{L}_\mathrm{2BE}(\pi) = \sum_{i,j} \mathrm{H}(\pi(\mathcal{M}_\mathrm{A})_i, \mathcal{M}_{\mathrm{B},j}). \tag{2}$$

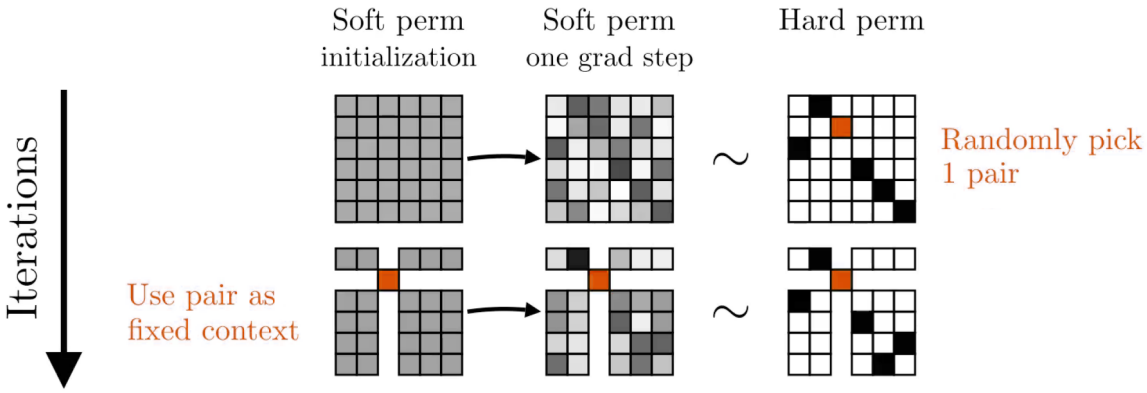

Figure 1: **First two iterations of a DiffPaSS bootstrap.** In this example, we pair sequences in a single species of size 6. Matrix entries are represented either as shades of gray – ranging from white (when the entry is 0) to black (when the entry is 1) – or in orange (also indicating a value of 1) when they identify pairs randomly sampled at the end of one iteration and used as fixed pairs in the next. A tilde ($\sim$) between a soft and a hard permutation indicates that the soft permutation is "close" to the hard permutation, see Eq. (7). An animation illustrating the full DiffPaSS bootstrap in this example is available at https://www.youtube.com/watch?v=G2rV4ldgTIY.

We define a smooth extension $\hat{\mathscr{L}}_{\text{2BE}}$ of $\mathscr{L}_{\text{2BE}}$ to soft permutations as follows. First, we represent the MSAs using one-hot encoding. Namely, let $\boldsymbol{m}_{\text{A},i,n}$ denote the one-hot vector corresponding to row (i.e. sequence) n and column (i.e. site) i in $\mathcal{M}_{\text{A}}$ – and similarly for $\mathcal{M}_{\text{B}}$. The matrix of observed counts of all joint amino-acid states at the column pair (i, j) can then be computed as $\sum_n \boldsymbol{m}_{\text{A},i,n} \otimes \boldsymbol{m}_{\text{B},j,n}$, where $\otimes$ denotes vector outer product. As this expression is well-defined and smooth for pairs of arbitrary vectors, it yields a smooth extension of counts and frequencies provided that all vector entries are non-negative. This leads to a smooth extension $\hat{\text{H}}(\cdot,\cdot)$ of the two-body entropy $\text{H}(\cdot,\cdot)$. In our case, for a soft permutation $\hat{\pi}$ represented as a matrix $\hat{P}$, we introduce a *soft MSA* as $\hat{\pi}(\mathcal{M}_{\text{A}}) = \hat{P}\boldsymbol{M}_{\text{A}}$, where $\boldsymbol{M}_{\text{A}}$ is the representation of $\mathcal{M}_{\text{A}}$ as a tensor with an additional one-hot dimension. We thus define the following smooth extension $\hat{\mathscr{L}}_{\text{2BE}}$ of the inter-chain two-body entropy loss $\mathscr{L}_{\text{2BE}}$:

$$\hat{\mathscr{L}}_{\text{2BE}}(\hat{\pi}) = \sum_{i,j} \hat{\text{H}}((\hat{P}\boldsymbol{M}_{\text{A}})_i, \boldsymbol{M}_{\text{B},j})\,. \tag{3}$$

Other scores

So far, we motivated the DiffPaSS bootstrap framework using the data structure of MSAs and the MI between MSA columns. This was based on the observation [14] that MI contains useful signal for matching paralogs between interacting protein families. However, a variety of other scores can also be used for paralog matching and for more general pairing problems, including scores based on sequence similarities, orthology, and phylogeny [1–10]. In some cases, these alternative scores are available even when alignments are not easy to construct or even meaningful. Importantly, the DiffPaSS framework is a general approach that can be applied to different scores. As an example, we extend the DiffPaSS framework to optimize graph alignment scores, and we consider applications to both aligned and non-aligned sequences.

Graph alignment scores. Let us consider two ordered collections of sequences $\mathcal{M}_{\text{A}}$ and $\mathcal{M}_{\text{B}}$. Let us define a weighted graph $\mathcal{G}_A$ (resp. $\mathcal{G}_B$), whose nodes represent the sequences in $\mathcal{M}_{\text{A}}$ (resp. $\mathcal{M}_{\text{B}}$), and whose pairwise weights are stored in a matrix $\mathcal{W}_A$ (resp. $\mathcal{W}_B$). Graph alignment (GA) can be performed between $\mathcal{G}_A$ and $\mathcal{G}_B$ using a variety of loss functions [15, 35–37]. When the GA loss function $\mathscr{L}_{\text{GA}}(\mathcal{W}_{\text{A}}, \mathcal{W}_{\text{B}})$ is differentiable, we propose the following variant of DiffPaSS: define $\hat{\mathscr{L}}_{\text{GA}}(\hat{\pi}) = \mathscr{L}_{\text{GA}}(\hat{P}\mathcal{W}_{\text{A}}\hat{P}^{\text{T}}, \mathcal{W}_{\text{B}})$, where $\hat{\pi}$ is a soft permutation encoded by the square matrix $\hat{P}$. This definition is natural in the special case where $\hat{\pi}$ is a hard permutation, as $\hat{P}\mathcal{W}_{\text{A}}\hat{P}^{\text{T}}$ is then the matrix

obtained from $\mathcal{W}_{\mathrm{A}}$ by permuting its rows and columns by $\hat{\pi}$. Then, we perform a DiffPaSS bootstrap procedure (see above), using $\hat{\mathscr{L}}$ as the differentiable loss.

Here, as in Refs. [8, 15], we align k-nearest neighbor graphs. Specifically, we use pairwise weight matrices $\mathcal{W}$ constructed from ordered collections $\mathcal{M} = (s_i)_i$ of sequences as follows. Considering a symmetric distance (or dissimilarity) metric d between sequences and an integer $k > 1$, we define the (i, j)-th entry of $\mathcal{W}$ as

$$\mathcal{W}_{ij} = \begin{cases} e^{-d(s_i,s_j)/D^2} & \text{if } s_i \text{ is among the } k \text{ nearest neighbors of } s_j\text{, or vice-versa,} \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Here, D is the average distance (over all sequences in $\mathcal{M}$) of the k-th nearest neighbor sequence. As distance metric, we use Hamming distances if the sequences are aligned, and edit distances otherwise. We set $k = 20$ or $k = 30$. Finally, as in Refs. [8, 15], we use

$$\mathscr{L}_{\mathrm{GA}}(\mathcal{W}_{\mathrm{A}}, \mathcal{W}_{\mathrm{B}}) = -\sum_i \sum_{j>i} (\mathcal{W}_{\mathrm{A}})_{ij} (\mathcal{W}_{\mathrm{B}})_{ij} \tag{5}$$

as a GA loss function.

Robust pairs and iterative variants of DiffPaSS

We empirically found that, when running a DiffPaSS bootstrap, some sequence pairs are found by all the hard permutations explored. We call these *robust pairs*, and notice that they tend to have high precision. This is illustrated in Fig. S3 for a benchmark prokaryotic dataset described in Appendix 2. This suggests that they can be used as a starting set $\mathcal{F}_{\mathrm{AB}}$ of fixed pairs in another run of DiffPaSS, where further fixed pairs are (randomly) chosen from the remaining sequences, see above. This process can be repeated several times by adding new robust pairs, found among the remaining sequences, to the set $\mathcal{F}_{\mathrm{AB}}$. We call this iterative procedure DiffPaSS-IPA ("Iterative Pairing Algorithm", following Bitbol et al. [12], Bitbol [14]). The final output of DiffPaSS-IPA is the hard permutation with lowest observed loss across all IPA runs. In practice, for all IPA variants of DiffPaSS, we use $N_{\mathrm{IPA}} = 3$ iterations here.

Results

DiffPaSS accurately and efficiently pairs paralogs using MI

DiffPaSS-MI pairs interacting sequences more accurately than other MI-based methods. Let us first consider coevolution-based pairing of partners from the MSAs of two interacting protein families. How does our differentiable pairing method compare to existing discrete pairing methods? To address this question, we test DiffPaSS and DiffPaSS-IPA, using the total inter-chain MI as a score (see Methods), on a benchmark dataset composed of ubiquitous prokaryotic proteins from two-component signaling systems [38, 39], which enable bacteria to sense and respond to environment signals. This dataset comprises cognate pairs of histidine kinases (HKs) and response regulators (RRs) determined using genome proximity, see Appendix 2. These proteins feature high specificity and generally interact one-to-one [40, 41]. We blind all partnerships and we compare different coevolution-based methods on the task of pairing these interacting sequences within each species, without giving any known paired sequences as input. Fig. 2 shows that both DiffPaSS and DiffPaSS-IPA significantly outperform the MI-IPA algorithm [14], which performs a discrete approximate maximization of the same MI score. The gain of performance obtained by using DiffPaSS instead of MI-IPA is particularly remarkable for relatively shallow MSAs, up to ∼2000 sequences deep. Recall that MI-IPA is quite data-thirsty [14]. Furthermore, Fig. 2 shows that, for MSAs up to depth 750 (resp. 1000), DiffPaSS (resp. DiffPaSS-IPA) outperforms a recent method which combines Monte-Carlo GA and MI-IPA [15]. This is significant because this combined GA and MI-IPA method exploits both phylogenetic sequence similarity via GA and coevolution via MI, while here we only used MI in DiffPaSS. However, the combined method slightly outperforms MI maximization (with DiffPaSS) for deeper MSAs (depths 2000 and 5000) – see

paragraph below for additional information. Note that for deeper MSAs (depth 5000), all methods, including MI-IPA, perform well and obtain correct results for more than 80% of the pairs [14, 15].

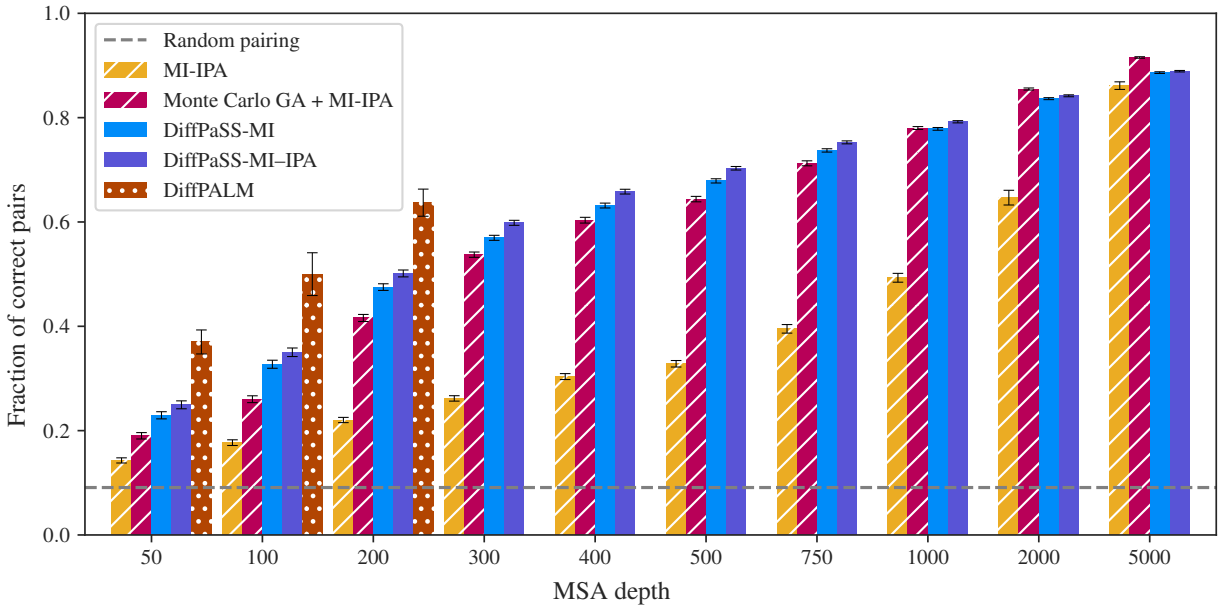

Figure 2: **Performance of coevolution-based pairing methods on HK-RR MSAs.** The performance of DiffPaSS-MI and DiffPaSS-MI-IPA is compared to that of MI-IPA [14], alone or combined with Monte Carlo GA [15], and to that of DiffPALM [32], for HK-RR MSAs with different depths. The HK-RR dataset is described in Appendix 2. With all methods, a full one-to-one within-species pairing is produced, and performance is measured as the fraction of correct pairs among all predicted pairs. For the combined Monte Carlo GA and MI-IPA method, we used the same settings as Gandarilla-Perez et al. [15]. Results for DiffPALM are taken from Lupo et al. [32].

While DiffPaSS-MI and MI-IPA are both based on MI, we recently introduced DiffPALM [32], an approach that leverages the MSA-based protein language model MSA Transformer [42]. Fig. 2 shows that DiffPaSS does not reach the performance achieved by DiffPALM. However, DiffPaSS is several orders of magnitude faster (see below), and easily scales to much deeper MSA depths for which DiffPALM cannot be run due to memory limitations.

DiffPaSS-MI is highly effective at extracting MI signal. How does DiffPaSS achieve better performance than MI-IPA? It was shown in Ref. [14] that, for relatively shallow MSAs, the approximate discrete optimization algorithm used to maximize inter-chain MI (Eq. (1)) in the MI-IPA approach does not provide pairings that reach MI values as high as the ground-truth pairings. Thus motivated, we compare the inter-chain MI of the paired MSAs produced by DiffPaSS, and by other methods, to the MI of the ground-truth pairings. We use the difference between the inter-chain two-body entropy loss (Eq. (2)) of a paired MSA and that of the ground-truth paired MSA . Indeed, this "excess loss" is equal to minus the difference between the corresponding inter-chain MIs, see Eqs. (1) and (2). Fig. 3 shows the distribution of excess inter-chain two-body entropy losses for each method. We observe that the median MI of DiffPaSS(-IPA) final pairings is indistinguishable or higher than that of the ground-truth pairings for all MSA depths considered (as excess losses are positive). Furthermore, the distribution of excess losses across MSAs becomes very narrow for deeper MSAs, showing that a score very close to the ground-truth MI is then systematically obtained. Meanwhile, for shallow MSAs, MI-IPA and the combined Monte Carlo GA + MI-IPA methods yield paired MSAs that have a substantially lower MI than the ground-truth pairings. This improves for deeper MSAs, faster for the combined method than for MI-IPA (see also [14] for MI-IPA). Note however that, for all MSA depths, the combined method yields paired MSAs with higher excess losses than DiffPaSS-MI(–IPA), even though it produces a higher fraction of correct pairs, on average, for the deepest MSAs (depths 2000 and 5000, see Fig. 2). While our focus is here on comparing to the ground-truth pairings, the

distributions of inter-chain two-body losses are shown in Fig. S2, for all methods as well as for random pairings. This figure shows the amount of available MI signal for pairing in these datasets. Overall, these results demonstrate that DiffPaSS is extremely effective at extracting the available MI signal.

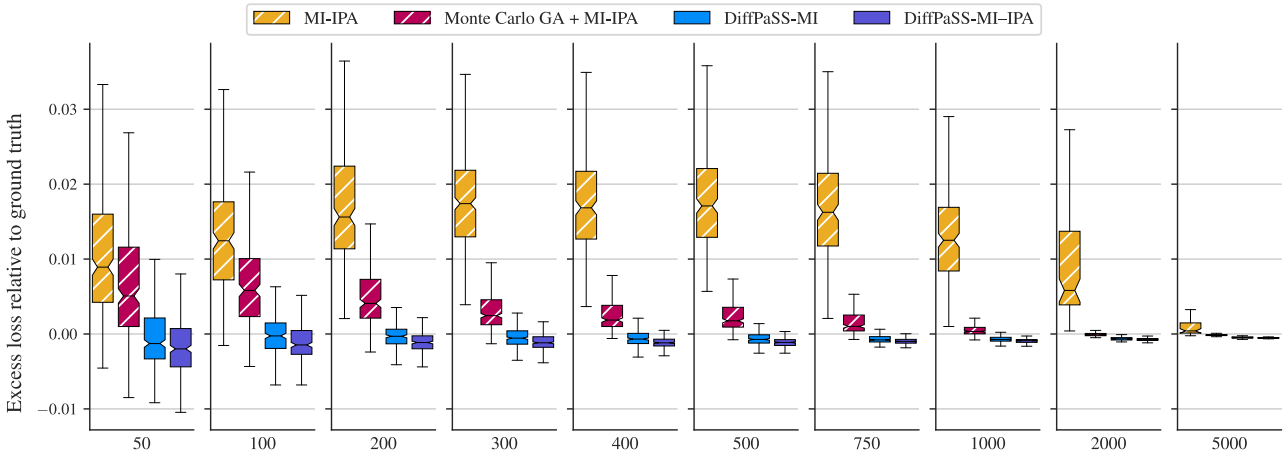

Figure 3: **Inter-chain two-body entropy losses for predicted pairings.** The distributions of the excess inter-chain two-body entropy losses of predicted pairings relative to ground-truth pairings are shown as box plots for several methods and several MSA depths, on the HK-RR dataset (same MSAs as in Fig. 2).

DiffPaSS-MI identifies robust pairs. The combined method Monte Carlo GA + MI-IPA relies on identifying robust pairs which are predicted by multiple independent runs of GA and then using them as a training set of known pairs for MI-IPA. Fig. S3 compares these robust pairs with those that are identified by DiffPaSS-MI(-IPA), specifically the set $\mathcal{F}_{\mathrm{AB}}$ after 3 IPA iterations, see Methods. It shows that, for relatively shallow MSAs, DiffPaSS-IPA typically identifies fewer correct pairs as robust than Monte Carlo GA, but its robust pairs are correct more often. For deeper MSAs, the number of correct robust pairs identified by DiffPaSS-MI–IPA sharply increases, while it plateaus for Monte Carlo GA.

DiffPaSS-MI is substantially faster than other approaches. How does DiffPaSS compare to existing algorithms in terms of computational runtime? DiffPaSS can easily be run on a modern GPU, while MI-IPA and Monte Carlo GA + MI-IPA are CPU-only algorithms. Fig. S6 demonstrates that DiffPaSS-MI is considerably faster than Monte Carlo GA + MI-IPA across all the MSA depths we analyzed. DiffPALM [32] (not shown in Fig. S6) has much longer runtimes than both methods, taking e.g. over three orders of magnitudes longer than DiffPaSS to pair HK-RR MSAs of depth ~ 50. Thus, a considerable asset of DiffPaSS is its rapidity, which makes it scalable to large datasets comprising many pairs of protein families.

DiffPaSS improves the structure prediction by AlphaFold-Multimer of some eukaryotic complexes

While genome proximity can often be used to pair interaction partners in prokaryotes, it is not the case in eukaryotes. Pairing correct interaction partners is thus a challenging problem in eukaryotes, which also often have many paralogs per species [43] while eukaryotic-specific protein families generally have fewer total homologs and smaller diversity than in prokaryotes. Solving this problem has an important application to the prediction of protein complex structure. Indeed, AlphaFold-Multimer (AFM) [21] relies on paired MSAs [21, 44]. Can DiffPaSS improve complex structure prediction by AFM [21] in eukaryotic complexes? To address this question, we consider the same 15 eukaryotic complexes as in Ref. [32], where improvements over the default AFM pairing methods were reported by pairing using the MSA-Transformer–based method DiffPALM [42]. More information on these structures and on the AFM setup can be found in Appendices 2 and 3. Fig. 4 compares the performance of AFM

on these complexes, using three different pairing methods (default AFM, DiffPALM, and DiffPaSS-MI) on the same initial unpaired MSAs. We use the DockQ score, a widely used measure of quality for protein-protein docking [45], as a performance metric for complex structure prediction. AFM confidence scores for these predictions are also shown in Fig. S4. These results show that DiffPaSS can improve complex structure prediction in some cases. Furthermore, our results are consistent with those we previously obtained with DiffPALM [32]: the two structures that are substantially improved are 6FYH and 6L5K.

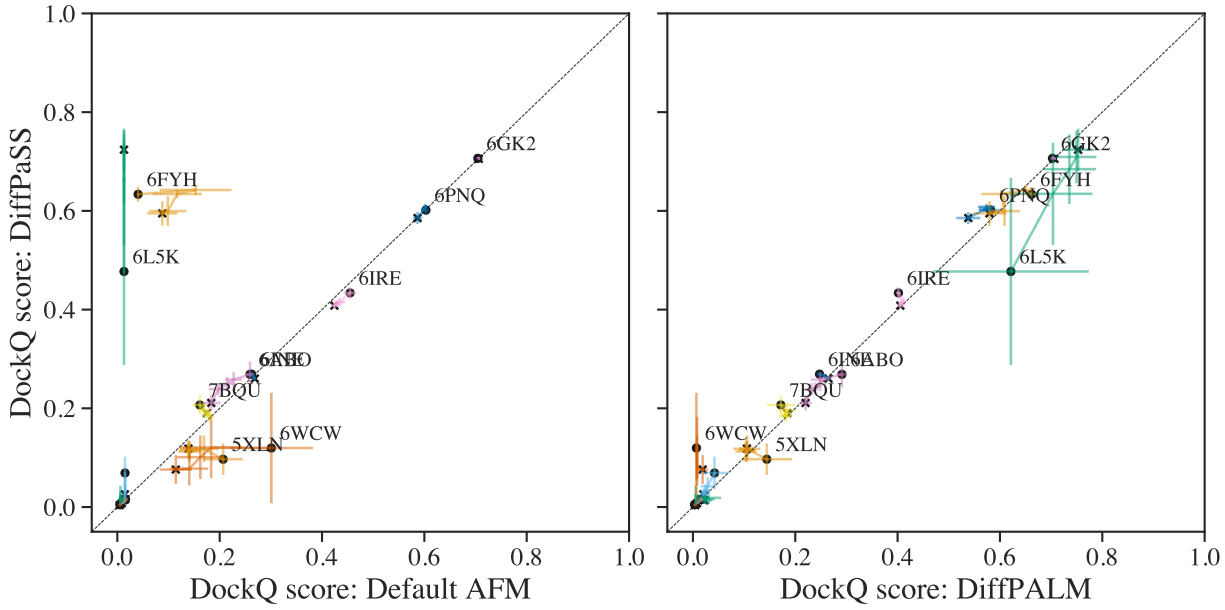

Figure 4: **Performance of structure prediction by AFM using different MSA pairing methods.** We report the performance of AFM, in terms of DockQ scores, for the 15 complexes considered in [32], using different pairing methods on the same MSAs. Left panel: DiffPaSS versus default AFM pairing. Right panel: DiffPaSS versus DiffPALM. For each complex, AFM is run five times, and the five top predicted structures by AFM confidence are considered each time, yielding 25 predicted structures total. For each complex, we show "trajectories" of performance starting from the top-confidence predicted structure (black circular marker) and ending with all predicted structures up to and including the fifth one (black cross marker). Results are averaged over the 5 runs and standard errors are shown as error bars. Points with DockQ below 0.1 are not labelled with their PDB ID for graphical reasons.

DiffPaSS allows accurate graph alignment

So far, we used DiffPaSS for the specific problem of paralog matching from the MSAs of two interacting protein families, using coevolution measured via MI. However, the optimization approach of DiffPaSS is general and can be applied to other scores, and to non-aligned sequences. In particular, it can be used for the problem of graph alignment, see Methods. To assess how well DiffPaSS performs at this problem, we first return to our benchmark HK-RR dataset. Instead of the inter-chain MI, to predict pairs we now use the GA loss [8, 15], defined in Eq. (5). Note that we use Hamming distances to define the weight matrices $\mathcal{W}_\mathrm{A}$ and $\mathcal{W}_\mathrm{B}$. Fig. 5 shows that DiffPaSS-GA is competitive with the Monte Carlo simulated annealing algorithm in Ref. [15], which optimizes the same GA loss. More precisely, DiffPaSS-GA performs slightly less well than Monte Carlo GA for shallow MSAs, but better than it for deeper ones, where the GA problem becomes trickier as the neighbor graph contains more and more inter-species edges. We also compare DiffPaSS-GA with DiffPaSS-MI, and find that DiffPaSS-MI outperforms DiffPaSS-GA for shallow MSAs, but performance becomes similar for deeper ones.

Contrary to MI, GA does not require sequences to be aligned. Indeed, one can use a distance measure between sequences which does not involve aligning them. This opens the way to broader applications. An interesting one, which is close to the paralog matching problem studied here, regards

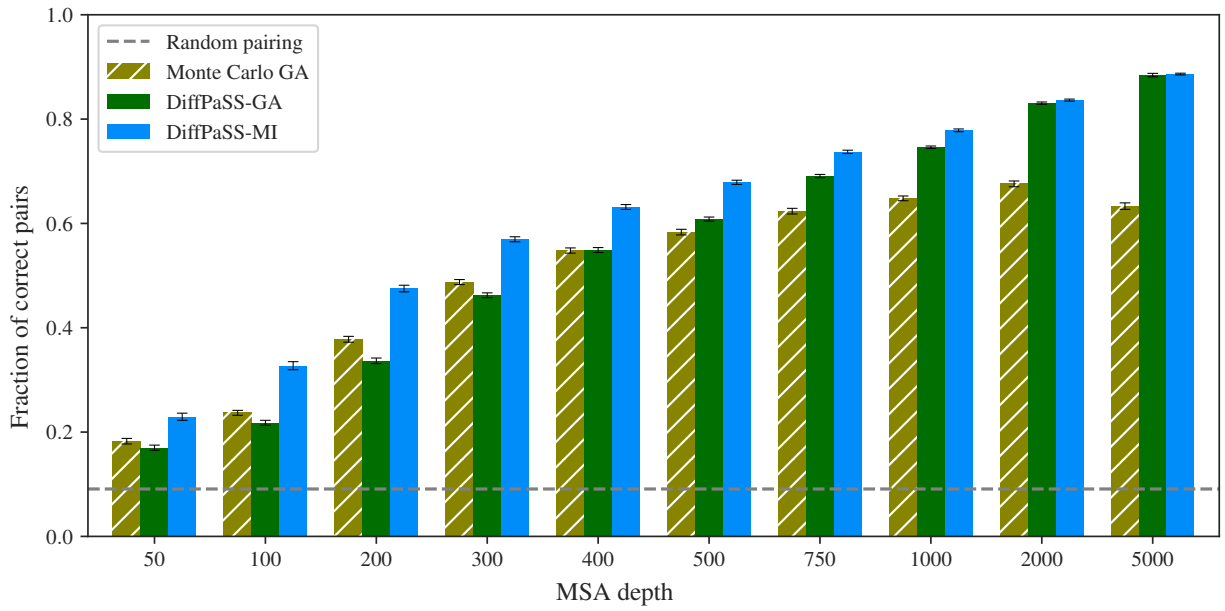

Figure 5: **Performance of GA- or MI-based pairing methods on HK-RR MSAs.** The fraction of correct pairs obtained is shown for DiffPaSS-GA and for the Monte Carlo-based GA implementation from [15], on the same HK-RR MSAs as in Fig. 2. The results for DiffPaSS-MI from Fig. 2 are shown for comparison.

pairing T cell receptor chains. Specifically, we consider collections of sequences from hypervariable CDR3α and CDR3β loops in TCRs binding to a fixed epitope, as in Ref. [46]. The task is to pair each α chain to its β chain within each human patient. Thus, here, patients play the part of species in our paralog matching problem. Our dataset of paired CDR3α-CDR3β loop sequences from TCRs is described in Appendix 2. These collections of CDR3α and CDR3β sequences are difficult to align due to their hypervariability, motivating the choice of GA for this problem, using the GA loss in Eq. (5) with weight matrices $\mathcal{W}_{\mathrm{A}}$ and $\mathcal{W}_{\mathrm{B}}$ defined using edit distances instead of Hamming distances (see Eq. (4)). Note also that here we use $k = 20$ nearest neighbors, in order to compare our results with those from Ref. [46], obtained using the same value of k. The differences between the losses obtained by DiffPaSS and those obtained by the Monte Carlo GA algorithm are shown in Fig. S5. DiffPaSS achieves substantially lower losses than the Monte Carlo GA algorithm, on average, in the four datasets containing the largest numbers of sequences to pair (from $\sim$700 to $\sim$2000). On the other hand, it yields slightly higher losses than Monte Carlo GA, on average, for datasets with 500 pairs or fewer. The good performance of DiffPaSS-GA on larger datasets is consistent with our HK-RR results (Fig. 5).

Discussion

We introduced DiffPaSS, a framework to pair interacting partners among two collections of biological sequences. DiffPaSS uses a versatile and hyperparameter-free differentiable optimization method that can be applied to various scores. It outperforms existing discrete optimization methods for pairing paralogs using MI, most spectacularly in the regime of shallow alignments. Strikingly, on our benchmark datasets, DiffPaSS is able to extract all the MI signal available for pairing. DiffPaSS is computationally highly efficient, compared to existing discrete optimization methods and, by far, to DiffPALM [32], our pairing method based on MSA Transformer. Thus, it has the potential be applied to large datasets. Our tests on some protein complexes show promise for paired alignments produced by DiffPaSS to improve complex structure prediction by AlphaFold-Multimer [21]. DiffPaSS does not need to start from aligned sequences, thus opening the way to broader applications. Our results on pairing TCR chains show promise for the use of DiffPaSS to optimize scores that cannot be constructed from MSAs.

Since DiffPaSS shows promise for structural biology, it would be very interesting to apply it more broadly to protein complex structure prediction problems. Besides, applications to TCRs are also promising. We expect CDR3α-CDR3β paired sequences to show lower co-evolution signal than other interacting protein families. Indeed, they are generated at random and selected for their joint ability to bind a given antigen, rather than directly co-evolving. Development of similarity measures that can cluster TCRs with the same specificity in sequence space is ongoing [47–53]. Using DiffPaSS-GA with these similarity measures could allow testing of metrics specifically optimised to capture TCR co-specificity.

One limitation of DiffPaSS in its current formulation is that it assumes one-to-one pairings. It would be very interesting to extend it to pairing problems between collections of different sizes. To achieve this, one could e.g. replace the Sinkhorn operator with Dykstra's operator, see Ref. [36].

Finally, despite DiffPaSS's ability to extract all available MI signal from our benchmark dataset, we found that our MSA-Transformer–based pairing method DiffPALM outperforms DiffPaSS. This shows that MSA-based protein language models such as MSA Transformer truly capture more signal usable for pairing than MI.

Acknowledgments

U. L., D. S., and A.-F. B. acknowledge funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 851173, to A.-F. B.). M. M. acknowledges funding by Cancer Research UK through a Non-Clinical Training Award [A29287].

References

[1] Chern-Sing Goh and Fred E Cohen. Co-evolutionary analysis reveals insights into protein–protein interactions. *Journal of Molecular Biology*, 324(1):177–192, 2002. doi: 10.1016/S0022-2836(02)01038-0.

[2] A. K. Ramani and E. M. Marcotte. Exploiting the co-evolution of interacting proteins to discover interaction specificity. *Journal of Molecular Biology*, 327(1):273–284, 2003. doi: 10.1016/S0022-2836(03)00114-1.

[3] J. Gertz, G. Elfond, A. Shustrova, M. Weisinger, M. Pellegrini, S. Cokus, and B. Rothschild. Inferring protein interactions from phylogenetic distance matrices. *Bioinformatics*, 19(16):2039–2045, 2003. doi: 10.1093/bioinformatics/btg278.

[4] J. M. Izarzugaza, D. Juan, C. Pons, J. A. Ranea, A. Valencia, and F. Pazos. TSEMA: interactive prediction of protein pairings between interacting families. *Nucleic Acids Research*, 34(Web Server issue):W315–319, 2006. doi: 10.1093/nar/gkl112.

[5] Elisabeth RM Tillier, Laurence Biro, Ginny Li, and Desiree Tillo. Codep: maximizing co-evolutionary interdependencies to discover interacting proteins. *Proteins*, 63(4):822–831, 2006. doi: 10.1002/prot.20948.

[6] J. M. Izarzugaza, D. Juan, C. Pons, F. Pazos, and A. Valencia. Enhancing the prediction of protein pairings between interacting families using orthology information. *BMC Bioinformatics*, 9:35, 2008. doi: 10.1186/1471-2105-9-35.

[7] E. R. Tillier and R. L. Charlebois. The human protein coevolution network. *Genome Res*, 19(10):1861–1871, 2009. doi: 10.1101/gr.092452.109.

[8] S. Bradde, A. Braunstein, H. Mahmoudi, F. Tria, M. Weigt, and R. Zecchina. Aligning graphs and finding substructures by a cavity approach. *EPL*, 89(3), 2010. doi: 10.1209/0295-5075/89/37009.

[9] I. Hajirasouliha, A. Schönhuth, D. de Juan, A. Valencia, and S. C. Sahinalp. Mirroring co-evolving trees in the light of their topologies. *Bioinformatics*, 28(9):1202–1208, 2012. doi: 10.1093/bioinformatics/bts109.

[10] M. El-Kebir, T. Marschall, I. Wohlers, M. Patterson, J. Heringa, A. Schonhuth, and G. W. Klau. Mapping proteins in the presence of paralogs using units of coevolution. *BMC Bioinformatics*, 14 Suppl 15:S18, 2013. doi: 10.1186/1471-2105-14-S15-S18.

[11] L. Burger and E. van Nimwegen. Accurate prediction of protein-protein interactions from sequence alignments using a Bayesian method. *Mol. Syst. Biol.*, 4:165, 2008. doi: 10.1038/msb4100203.

[12] Anne-Florence Bitbol, Robert S Dwyer, Lucy J Colwell, and Ned S Wingreen. Inferring interaction partners from protein sequences. *Proc. Natl. Acad. Sci. U.S.A.*, 113(43):12180–12185, 2016. doi: 10.1073/pnas.1606762113.

[13] T. Gueudre, C. Baldassi, M. Zamparo, M. Weigt, and A. Pagnani. Simultaneous identification of specifically interacting paralogs and interprotein contacts by direct coupling analysis. *Proc. Natl. Acad. Sci. U.S.A.*, 113(43):12186–12191, 2016. doi: 10.1073/pnas.1607570113.

[14] Anne-Florence Bitbol. Inferring interaction partners from protein sequences using mutual information. *PLoS Comput. Biol.*, 14(11):e1006401, 2018. doi: 10.1371/journal.pcbi.1006401.

[15] Carlos A. Gandarilla-Perez, Sergio Pinilla, Anne-Florence Bitbol, and Martin Weigt. Combining phylogeny and coevolution improves the inference of interaction partners among paralogous proteins. *PLoS Comput. Biol.*, 19(3):e1011010, 2023. doi: 10.1371/journal.pcbi.1011010.

[16] F. Pazos and A. Valencia. Similarity of phylogenetic trees as indicator of protein–protein interaction. *Protein Eng. Des. Sel.*, 14(9):609–614, 2001.

[17] D. Ochoa and F. Pazos. Studying the co-evolution of protein families with the Mirrortree web server. *Bioinformatics*, 26(10):1370–1371, 2010. doi: 10.1093/bioinformatics/btq137.

[18] D. Ochoa, D. Juan, A. Valencia, and F. Pazos. Detection of significant protein coevolution. *Bioinformatics*, 31(13):2166–2173, 2015. doi: 10.1093/bioinformatics/btv102.

[19] Qian Cong, Ivan Anishchenko, Sergey Ovchinnikov, and David Baker. Protein interaction networks revealed by proteome coevolution. *Science*, 365(6449):185–189, 2019. doi: 10.1126/science.aaw6718.

[20] A. G. Green, H. Elhabashy, K. P. Brock, R. Maddamsetti, O. Kohlbacher, and D. S. Marks. Large-scale discovery of protein interactions at residue resolution using co-evolution calculated from genomic sequences. *Nat Commun*, 12(1):1396, 2021. doi: 10.1038/s41467-021-21636-z.

[21] Richard Evans, Michael O'Neill, Alexander Pritzel, Natasha Antropova, Andrew Senior, Tim Green, Augustin Žídek, Russ Bates, Sam Blackwell, Jason Yim, Olaf Ronneberger, Sebastian Bodenstein, Michal Zielinski, Alex Bridgland, Anna Potapenko, Andrew Cowie, Kathryn Tunyasuvunakool, Rishub Jain, Ellen Clancy, Pushmeet Kohli, John Jumper, and Demis Hassabis. Protein complex prediction with AlphaFold-Multimer. *bioRxiv*, 2021. doi: 10.1101/2021.10.04.463034.

[22] Ian Humphreys, Jimin Pei, Minkyung Baek, Aditya Krishnakumar, Ivan Anishchenko, Sergey Ovchinnikov, Jing Zhang, Travis J. Ness, Sudeep Banjade, Saket R. Bagde, Viktoriya G. Stancheva, Xiao Han Li, Kaixian Liu, Zhi Zheng, Daniel J. Barrero, Upasana Roy, Jochen Kuper, Israel S. Fernández, Barnabas Szakal, Dana Branzei, Josep Rizo, Caroline Kisker, Eric C. Greene, Sue Biggins, Scott Keeney, Elizabeth A. Miller, J. Christopher Fromme, Tamara L. Hendrickson, Qian Cong, and David Baker. Computed structures of core eukaryotic protein complexes. *Science*, 374(6573), 2021. ISSN 10959203. doi: 10.1126/science.abm4805.

[23] M. Weigt, R. A. White, H. Szurmant, J. A. Hoch, and T. Hwa. Identification of direct residue contacts in protein-protein interaction by message passing. *Proc. Natl. Acad. Sci. U.S.A.*, 106 (1):67–72, 2009. doi: 10.1073/pnas.0805923106.

[24] S. D. Dunn, L. M. Wahl, and G. B. Gloor. Mutual information without the influence of phylogeny or entropy dramatically improves residue contact prediction. *Bioinformatics*, 24(3):333–340, 2008.

[25] Alexander Schug, Martin Weigt, José N. Onuchic, Terence Hwa, and Hendrik Szurmant. High-resolution protein complexes from integrating genomic information with molecular simulation. *Proceedings of the National Academy of Sciences*, 106(52):22124–22129, 2009. doi: 10.1073/pnas.0912100106. URL https://www.pnas.org/doi/abs/10.1073/pnas.0912100106.

[26] F. Morcos, A. Pagnani, B. Lunt, A. Bertolino, D. S. Marks, C. Sander, R. Zecchina, J. N. Onuchic, T. Hwa, and M. Weigt. Direct-coupling analysis of residue coevolution captures native contacts across many protein families. *Proc. Natl. Acad. Sci. U.S.A.*, 108(49):E1293–1301, 2011. doi: 10.1073/pnas.1111471108.

[27] D. S. Marks, L. J. Colwell, R. Sheridan, T. A. Hopf, A. Pagnani, R. Zecchina, and C. Sander. Protein 3D structure computed from evolutionary sequence variation. *PLoS ONE*, 6(12):1–20, 2011. doi: 10.1371/journal.pone.0028766.

[28] J. I. Sułkowska, F. Morcos, M. Weigt, T. Hwa, and J. N. Onuchic. Genomics-aided structure prediction. *Proc. Natl. Acad. Sci. U.S.A.*, 109(26):10340–10345, 2012.

[29] G. Marmier, M. Weigt, and A.-F. Bitbol. Phylogenetic correlations can suffice to infer protein partners from sequences. *PLoS Comput. Biol.*, 15(10):e1007179, 2019. doi: 10.1371/journal.pcbi.1007179.

[30] A. Gerardos, N. Dietler, and A.-F. Bitbol. Correlations from structure and phylogeny combine constructively in the inference of protein partners from sequences. *PLoS Comput. Biol.*, 18(5): e1010147, 2022. doi: 10.1371/journal.pcbi.1010147.

[31] Bo Chen, Ziwei Xie, Jiezhong Qiu, Zhaofeng Ye, Jinbo Xu, and Jie Tang. Improved the heterodimer protein complex prediction with protein language models. *Briefings in Bioinformatics*, 24(4):bbad221, 2023. doi: 10.1093/bib/bbad221.

[32] Umberto Lupo, Damiano Sgarbossa, and Anne-Florence Bitbol. Pairing interacting protein sequences using masked language modeling. *Proceedings of the National Academy of Sciences*, 121 (27):e2311887121, 2024. doi: 10.1073/pnas.2311887121. URL https://www.pnas.org/doi/abs/10.1073/pnas.2311887121.

[33] Gonzalo E. Mena, David Belanger, Scott Linderman, and Jasper Snoek. Learning latent permutations with Gumbel-Sinkhorn networks. *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, pages 1–22, 2018. URL https://openreview.net/forum?id=Byt3oJ-0W.

[34] Richard Sinkhorn. A Relationship Between Arbitrary Positive Matrices and Doubly Stochastic Matrices. *The Annals of Mathematical Statistics*, 35(2):876–879, 1964. doi: 10.1214/aoms/1177703591. URL https://doi.org/10.1214/aoms/1177703591.

[35] Hermina Petric Maretic, Mireille El Gheche, Giovanni Chierchia, and Pascal Frossard. Got: an optimal transport framework for graph comparison. *Advances in Neural Information Processing Systems*, 32, 2019.

[36] Hermina Petric Maretic, Mireille El Gheche, Matthias Minder, Giovanni Chierchia, and Pascal Frossard. Wasserstein-based graph alignment. *IEEE Transactions on Signal and Information Processing over Networks*, 8:353–363, 2022.

[37] Hermina Petric Maretic, Mireille El Gheche, Giovanni Chierchia, and Pascal Frossard. FGOT: Graph distances based on filters and optimal transport. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7710–7718, 2022.

[38] M. Barakat, P. Ortet, C. Jourlin-Castelli, M. Ansaldi, V. Mejean, and D. E. Whitworth. P2CS: a two-component system resource for prokaryotic signal transduction research. *BMC Genomics*, 10:315, 2009. doi: 10.1186/1471-2164-10-315.

[39] M. Barakat, P. Ortet, and D. E. Whitworth. P2CS: a database of prokaryotic two-component systems. *Nucleic Acids Research*, 39(Database issue):D771–776, 2011. doi: 10.1093/nar/gkq1023.

[40] M. T. Laub and M. Goulian. Specificity in two-component signal transduction pathways. *Annu. Rev. Genet.*, 41:121–145, 2007. doi: 10.1146/annurev.genet.41.042007.170548.

[41] R. R. Cheng, F. Morcos, H. Levine, and J. N. Onuchic. Toward rationally redesigning bacterial two-component signaling systems using coevolutionary information. *Proc. Natl. Acad. Sci. U.S.A.*, 111(5):E563–571, 2014. doi: 10.1073/pnas.1323734111.

[42] Roshan M Rao, Jason Liu, Robert Verkuil, Joshua Meier, John Canny, Pieter Abbeel, Tom Sercu, and Alexander Rives. MSA Transformer. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pages 8844–8856. PMLR, 2021. URL https://proceedings.mlr.press/v139/rao21a.html.

[43] Kira S. Makarova, Yuri I. Wolf, Sergey L. Mekhedov, Boris G. Mirkin, and Eugene V. Koonin. Ancestral paralogs and pseudoparalogs and their role in the emergence of the eukaryotic cell. *Nucleic Acids Research*, 33(14):4626–4638, 2005. ISSN 0305-1048. doi: 10.1093/nar/gki775.

[44] Patrick Bryant, Gabriele Pozzati, and Arne Elofsson. Improved prediction of protein-protein interactions using AlphaFold2. *Nat Commun*, 13(1):1265, 2022. doi: 10.1038/s41467-022-28865-w.

[45] Sankar Basu and Björn Wallner. DockQ: A quality measure for protein-protein docking models. *PLoS ONE*, 11(8):1–9, 2016. doi: 10.1371/journal.pone.0161879.

[46] Martina Milighetti, Yuta Nagano, James Henderson, Uri Hershberg, Andreas Tiffeau-Mayer, Anne-Florence Bitbol, and Benny Chain. Intra- and inter-chain contacts determine TCR specificity: applying protein co-evolution methods to TCR$\alpha\beta$ pairing. *bioRxiv*, 2024. doi: 10.1101/2024.05.24.595718. URL https://www.biorxiv.org/content/early/2024/05/29/2024.05.24.595718.

[47] Niclas Thomas, Katharine Best, Mattia Cinelli, Shlomit Reich-Zeliger, Hilah Gal, Eric Shifrut, Asaf Madi, Nir Friedman, John Shawe-Taylor, and Benny Chain. Tracking global changes induced in the CD4 T-cell receptor repertoire by immunization with a complex antigen using short stretches of CDR3 protein sequence. *Bioinformatics*, 30(22):3181–3188, 2014. doi: 10.1093/bioinformatics/btu523. URL https://doi.org/10.1093/bioinformatics/btu523.

[48] Pradyot Dash, Andrew J. Fiore-Gartland, Tomer Hertz, George C. Wang, Shalini Sharma, Aisha Souquette, Jeremy Chase Crawford, E. Bridie Clemens, Thi H. O. Nguyen, Katherine Kedzierska, Nicole L. La Gruta, Philip Bradley, and Paul G. Thomas. Quantifiable predictive features define epitope specific T cell receptor repertoires. *Nature*, 547(7661):89–93, 2017. ISSN 0028-0836. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5616171/.

[49] Neerja Thakkar and Chris Bailey-Kellogg. Balancing sensitivity and specificity in distinguishing TCR groups by CDR sequence similarity. *BMC Bioinformatics*, 20(1):241, 2019. ISSN 1471-2105. doi: 10.1186/s12859-019-2864-8. URL https://doi.org/10.1186/s12859-019-2864-8.

[50] Milena Vujovic, Kristine Fredlund Degn, Frederikke Isa Marin, Anna-Lisa Schaap-Johansen, Benny Chain, Thomas Lars Andresen, Joseph Kaplinsky, and Paolo Marcatili. T cell receptor

sequence clustering and antigen specificity. *Computational and Structural Biotechnology Journal*, 18:2166–2173, 2020. ISSN 2001-0370. doi: https://doi.org/10.1016/j.csbj.2020.06.041. URL https://www.sciencedirect.com/science/article/pii/S2001037020303305.

[51] Kevin Wu, Kathryn E. Yost, Bence Daniel, Julia A. Belk, Yu Xia, Takeshi Egawa, Ansuman Satpathy, Howard Y. Chang, and James Zou. TCR-BERT: learning the grammar of T-cell receptors for flexible antigen-binding analyses. *bioRxiv*, page 2021.11.18.469186, 2021. doi: 10.1101/2021.11.18.469186. URL https://www.biorxiv.org/content/early/2021/11/20/2021.11.18.469186.

[52] Pieter Meysman, Justin Barton, Barbara Bravi, Liel Cohen-Lavi, Vadim Karnaukhov, Elias Lilleskov, Alessandro Montemurro, Morten Nielsen, Thierry Mora, Paul Pereira, Anna Postovskaya, María Rodríguez Martínez, Jorge Fernandez de Cossio-Diaz, Alexandra Vujkovic, Aleksandra M. Walczak, Anna Weber, Rose Yin, Anne Eugster, and Virag Sharma. Benchmarking solutions to the T-cell receptor epitope prediction problem: IMMREP22 workshop report. *ImmunoInformatics*, 9:100024, 2023. ISSN 2667-1190. doi: https://doi.org/10.1016/j.immuno.2023.100024. URL https://www.sciencedirect.com/science/article/pii/S2667119023000046.

[53] Yuta Nagano, Andrew Pyo, Martina Milighetti, James Henderson, John Shawe-Taylor, Benny Chain, and Andreas Tiffeau-Mayer. Contrastive learning of T cell receptor representations. *arXiv*, page 2406.06397, 2024. URL https://arxiv.org/abs/2406.06397.

[54] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955. doi: 10.1002/nav.3800020109.

[55] Mikhail Goncharov, Dmitry Bagaev, Dmitrii Shcherbinin, Ivan Zvyagin, Dmitry Bolotin, Paul G. Thomas, Anastasia A. Minervina, Mikhail V. Pogorelyy, Kristin Ladell, James E. McLaren, David A. Price, Thi H.O. Nguyen, Louise C. Rowntree, E. Bridie Clemens, Katherine Kedzierska, Garry Dolton, Cristina Rafael Rius, Andrew Sewell, Jerome Samir, Fabio Luciani, Ksenia V. Zornikova, Alexandra A. Khmelevskaya, Saveliy A. Sheetikov, Grigory A. Efimov, Dmitry Chudakov, and Mikhail Shugay. VDJdb in the pandemic era: a compendium of T cell receptors specific for SARS-CoV-2. *Nature Methods 2022 19:9*, 19(9):1017–1019, August 2022. ISSN 1548-7105. doi: 10.1038/s41592-022-01578-0. URL https://www.nature.com/articles/s41592-022-01578-0. Publisher: Nature Publishing Group.

Supplementary material

Contents

1 Detailed methods **15**
1.1 Preliminaries 15
1.2 Initialization and bootstrapped optimization 16
1.3 Independence of number of Sinkhorn normalizations 17

2 Datasets **18**

3 General points on AlphaFold-Multimer (AFM) **18**

4 Supplementary figures **19**

1 Detailed methods

In this section, we present a more detailed account of the methods developed in this work. We acknowledge that there are some redundancies with the main text methods. We retained them for the sake of completeness and readability of this Supplementary material.

1.1 Preliminaries

Let $\mathcal{M}_{\mathrm{A}}$ and $\mathcal{M}_{\mathrm{B}}$ be ordered collections of amino-acid sequences that are partitioned into K groups, each of size N_k where $k = 1, \ldots, K$. In the important special case where $\mathcal{M}_{\mathrm{A}}$ and $\mathcal{M}_{\mathrm{B}}$ are collections of proteins from two interacting protein families, the "groups" will be species, with species k assumed to contain N_k paralogous proteins in both families.

Let $\mathscr{S}$ be a score function of the two ordered collections. We would like to find a permutation π of the entries in $\mathcal{M}_{\mathrm{A}}$ which maximises $\mathscr{S}(\pi(\mathcal{M}_{\mathrm{A}}), \mathcal{M}_{\mathrm{B}})$, under the constraint that π does not send sequences from any one group into a different group. A priori, there are $\prod_{k=1}^{K} N_k!$ permutations satisfying this constraint. Note that we can equivalently describe this as the problem of finding $\mathscr{S}$-maximising one-to-one matchings between the sequences in $\mathcal{M}_{\mathrm{A}}$ and those in $\mathcal{M}_{\mathrm{B}}$. Since $\mathcal{M}_{\mathrm{A}}$ and $\mathcal{M}_{\mathrm{B}}$ will remain fixed, by abusing notation we denote $\mathscr{S}(\pi(\mathcal{M}_{\mathrm{A}}), \mathcal{M}_{\mathrm{B}})$ simply by $\mathscr{S}(\pi)$.

Let $N > 0$, and denote the set of permutation matrices of N elements by $\mathcal{P}_N$. For any integer $\ell > 0$ and arbitrary $N \times N$ square matrix X, define the *ℓ-truncated Sinkhorn operator*

$$S^{\ell}(X) = (\mathcal{C} \circ \mathcal{R})^{\ell}(\exp(X)) \tag{6}$$

consisting of first applying the componentwise exponential function exp to X, and then iteratively normalizing rows ($\mathcal{R}$) and columns ($\mathcal{C}$) ℓ times. It can be shown [33] that $S(X) := \lim_{\ell \to \infty} S^{\ell}(X)$ defines a smooth operator mapping to bistochastic matrices[1] and that, for almost all X,

$$\lim_{\tau \to 0^+} S(X/\tau) = M(X) := \arg\max_{P \in \mathcal{P}_N} [\mathrm{trace}(P^{\mathrm{T}} X)]. \tag{7}$$

The operator M defined in Eq. (7), which maps onto permutation matrices, can be computed using standard discrete algorithms for linear assignment problems [54]. Eq. (7) implies that, using the "parameterization matrices" X, one can smoothly navigate the space $\mathcal{B}_N$ of all $N \times N$ bistochastic (resp. near-bistochastic) matrices using S (resp. S^{ℓ}), while keeping track of the "nearest" permutation matrices using M. In what follows, we will refer to (near-)bistochastic matrices as "soft permutations" and to true permutations as "hard permutations".

[1] A bistochastic matrix is a matrix with non-negative entries whose all rows and columns sum to 1.

We may therefore hope to find optimal hard permutations for the original score $\mathscr{S}$ by optimizing a suitable smooth extension $\hat{\mathscr{S}}$ of $\mathscr{S}$ to soft permutations, since this can be done efficiently using gradient methods. In general, optimization of $\hat{\mathscr{S}}$ can be very sensitive to hyperparameters such as the "temperature" $\tau > 0$ in Eq. (7), the standard deviation of the entries of X at initialization, the optimizer learning rate, and the strength of regularization.

1.2 Initialization and bootstrapped optimization

We found that solving $X^* = \arg\min_X \hat{\mathscr{L}}_{\text{2BE}}(S^\ell(X/\tau))$, for some choice of ℓ and τ, using several steps of gradient descent, generally yields sub-optimal hard permutations $M(X^*)$ for the original loss $\mathscr{L}_{\text{2BE}}$, see Eq. (7). Informally, this is because hard permutations are local minima for $\hat{\mathscr{L}}$, a situation reminiscent of how the entropy of a single Bernoulli random variable with parameter p has minima at the "sharp" cases $p = 0$ and $p = 1$. Nevertheless, we found that outcomes improved when all entries of X are initialized to be zero. Indeed, as shown in Fig. S1 for the benchmark HK-RR prokaryotic dataset described in Appendix 2, the *first* gradient step alone, when $X \equiv 0$ at initialization, is often competitive with a full-blown discrete algorithm for approximate maximization of $\mathscr{S}_{\text{MI}}$, called MI-IPA [14]. Finally, as previously observed using several methods [12, 14, 15, 32], pairing performance can be expected to increase if correct pairings are used as fixed context, biasing the computation of the two-body entropies. Together, these considerations led us to DiffPaSS, our proposed bootstrapped approach to differentiable pairing. Given a loss $\mathscr{L}$ and its smooth extension $\hat{\mathscr{L}}$, two ordered collections $\mathcal{M}_{\text{A}}$ and $\mathcal{M}_{\text{B}}$ containing D sequences, and (optionally) a pre-existing set $\mathcal{F}_{\text{AB}} = \{(a_i, b_i)\}_{i=1}^{D_{\text{fix}}}$ containing D_{fix} matched pairs of sequences, we proceed as follows.[2] Initialize $\mathcal{F}' = \mathcal{F}_{\text{AB}}$; then, for every $n = 1, \ldots, D - D_{\text{fix}}$:

1. define P' as the $(D_{\text{fix}}+n-1)\times(D_{\text{fix}}+n-1)$ permutation matrix corresponding to the matchings in $\mathcal{F}'$;
2. initialize two zero $D \times D$ matrices P and $\hat{P}$, and copy P' into the row-column pairs belonging to $\mathcal{F}'$ in both cases;
3. initialize a $(D - D_{\text{fix}} - n + 1) \times (D - D_{\text{fix}} - n + 1)$ parameterization matrix $X \equiv 0$, to be used for sequences not involved in pairs in $\mathcal{F}'$;
4. update $X \leftarrow -\nabla(\hat{\mathscr{L}} \circ \tilde{S}^\ell)(X = 0)$, where $\tilde{S}^\ell(X)$ denotes copying $S^\ell(X)$ into the submatrix of $\hat{P}$ obtained by removing rows and columns involved in pairs in $\mathcal{F}'$;
5. compute the hard permutation $\pi = \tilde{M}(X)$, where $\tilde{M}(X)$ denotes copying $M(X)$ into the submatrix of P obtained by removing rows and columns involved in pairs in $\mathcal{F}'$;
6. pick n pairs of sequences matched by π, but not in $\mathcal{F}_{\text{AB}}$, uniformly at random;
7. update $\mathcal{F}' \leftarrow \mathcal{F}_{\text{AB}} \cup \{\text{pairs selected in step 6}\}$.

For every $n = 1, \ldots, D - D_{\text{fix}}$, we record the loss $\mathscr{L}(\pi)$ at step 5, and the final output of DiffPaSS is the hard permutation π^* corresponding to the lowest recorded loss.

Note that the gradient of $S^\ell(X/\tau)$, evaluated at $X = 0$, only changes by a global scale factor if τ is changed, and the matching operator M is scale-invariant. Hence, we can set $\tau = 1$ as the obtained hard permutations are independent of it. Similarly, all hard permutations obtained are independent of the choice of learning rate and regularization strength. Perhaps more surprisingly, for any $\ell > 1$, the gradient of S^ℓ evaluated at $X = 0$ is equal to corresponding gradient of $S^{\ell=1}$. We prove this in Appendix 1.3. Hence, we can set $\ell = 1$ throughout, leading to significant runtime gains.

[2] An animation illustrating this algorithm in the special case $D_{\text{fix}} = 0$ is available at the following URL: https://www.youtube.com/watch?v=G2rV4ldgTIY.

Robust pairs and DiffPaSS-IPA. We noticed that, even when the starting set $\mathcal{F}_{\mathrm{AB}}$ of fixed pairs is empty, some pairs are matched by all the $D - D_{\mathrm{fix}}$ hard permutations π explored by DiffPaSS (see step 5 in Appendix 1.2). We call these *robust pairs*, and notice that they tend to have high precision, see Fig. S3. This suggests that they can be used as the starting set $\mathcal{F}_{\mathrm{AB}}$ of fixed pairs in a further run of DiffPaSS. We call this procedure DiffPaSS-IPA ("Iterative Pairing Algorithm", following Bitbol et al. [12], Bitbol [14]). It can be iterated several times, enlarging the set of robust pairs after each iteration, or stopping if no further robust pairs are found. The final output of DiffPaSS-IPA is the hard permutation with lowest observed loss across all IPA runs. We use $N_{\mathrm{IPA}} = 3$ iterations throughout.

1.3 Independence of number of Sinkhorn normalizations

Let S^ℓ (for an integer $\ell > 0$) be the ℓ-truncated Sinkhorn operator defined in Eq. (6). In this section we prove that, for any $\ell > 1$, all first-order derivatives of S^ℓ, when evaluated at the zero matrix, are equal to the corresponding derivatives of $S^{\ell=1}$.

Let $\mathcal{R}$ (resp. $\mathcal{C}$) be the row-wise (resp. column-wise) matrix normalization operator on $D \times D$ matrices. Denote the partial derivative operator with respect to the (i, j)-th matrix entry by ∂_{ij}, and the (k, l)-th matrix component of a matrix-valued operator $\mathcal{O}$ by $\mathcal{O}_{kl}$. Furthermore, let $\mathcal{T}^* = \mathcal{C} \circ \mathcal{R}$. Let $\mathbf{1}_{\mathrm{mat}}$ denote the $D \times D$ matrix whose entries are all equal to 1. Given the definition of the Sinkhorn operator in Eq. (6), and since the componentwise exponential of the zero matrix is a matrix of ones, it suffices, for our purposes, to show that

$$[\partial_{ij}(\mathcal{T}^* \circ \mathcal{T}^*)_{kl}](\mathbf{1}_{\mathrm{mat}}) = [\partial_{ij}\mathcal{T}^*_{kl}](\mathbf{1}_{\mathrm{mat}}) \tag{8}$$

for all $i, j, k, l = 1, \dots, D$. Indeed, we will prove that this actually holds when both sides of Eq. (8) are evaluated at $\mu\mathbf{1}_{\mathrm{mat}}$, for any real number $\mu > 0$.

Proof of Eq. (8). Let $\boldsymbol{X}$ denote a $D \times D$ matrix with positive entries. We begin by noting that

$$[\partial_{ij}\mathcal{R}_{kl}](\boldsymbol{X}) = \delta_{ik}\,[\partial_j\mathcal{T}_l](\boldsymbol{X}_{i\cdot}) \quad \text{and} \quad [\partial_{ij}\mathcal{C}_{kl}](\boldsymbol{X}) = \delta_{jl}\,[\partial_i\mathcal{T}_k](\boldsymbol{X}_{\cdot j})\,, \tag{9}$$

where δ denotes the Kronecker delta, $\mathcal{T}$ the normalization operator for D-dimensional vectors, and $\boldsymbol{X}_{i\cdot}$ (resp. $\boldsymbol{X}_{\cdot j}$) the i-th row (resp. j-th column) of $\boldsymbol{X}$.

Let $\boldsymbol{x}$ denote a D-dimensional vector. The partial derivatives of the components of $\mathcal{T}$, evaluated at $\boldsymbol{x}$, are given by

$$\partial_\alpha\mathcal{T}_\beta(\boldsymbol{x}) = \frac{\partial}{\partial\alpha}\frac{x_\beta}{\sum_\gamma x_\gamma} = \frac{\delta_{\alpha\beta}}{\sum_\gamma x_\gamma} - \frac{x_\beta}{(\sum_\gamma x_\gamma)^2}. \tag{10}$$

Let $\mathbf{1}$ denote the D-dimensional vector whose entries are all equal to 1. Applying Eq. (10) to $\boldsymbol{x} = \mu\mathbf{1}$ yields, for any real number $\mu > 0$,

$$\partial_\alpha\mathcal{T}_\beta(\mu\mathbf{1}) = \frac{1}{D\mu}(\delta_{\alpha\beta} - 1/D) = \frac{1}{D\mu}\boldsymbol{\Delta}_{\alpha\beta}, \tag{11}$$

where we have defined the square matrix $\boldsymbol{\Delta} = \mathrm{Id} - \mathbf{1}_{\mathrm{mat}}/D$, with Id denoting the $D \times D$ identity matrix. We remark for later that $\mathbf{1}_{\mathrm{mat}}/D$ is idempotent (i.e., its matrix square is itself), and that therefore so is $\boldsymbol{\Delta}$.

Using the chain rule and the fact that $\mathcal{R}(\mu\mathbf{1}_{\mathrm{mat}}) = \mathbf{1}_{\mathrm{mat}}/D$ for any $\mu > 0$, we can write

$$\begin{aligned}
[\partial_{ij}\mathcal{T}^*_{kl}](\mu\mathbf{1}_{\mathrm{mat}}) &= \sum_{m,n}[\partial_{mn}\mathcal{C}_{kl}](\mathbf{1}_{\mathrm{mat}}/D)\,[\partial_{ij}\mathcal{R}_{mn}](\mu\mathbf{1}_{\mathrm{mat}}) \\
&= \sum_{m,n}\delta_{nl}\boldsymbol{\Delta}_{mk}\,\frac{\delta_{im}}{D\mu}\boldsymbol{\Delta}_{jn} \\
&= \frac{1}{D\mu}\boldsymbol{\Delta}_{ik}\boldsymbol{\Delta}_{jl}\,,
\end{aligned} \tag{12}$$

where we used Eq. (9) and Eq. (11) to obtain the second line.

We now compute

$$
\begin{aligned}
[\partial_{ij}(\mathcal{T}^* \circ \mathcal{T}^*)_{kl}](\mu \mathbf{1}_{\text{mat}}) &= \sum_{mn} [\partial_{mn}\mathcal{T}^*_{kl}](\mathbf{1}_{\text{mat}}/D)\, [\partial_{ij}\mathcal{T}^*_{mn}](\mu \mathbf{1}_{\text{mat}}) \\
&= \frac{1}{D\mu} \sum_{mn} \mathbf{\Delta}_{mk}\mathbf{\Delta}_{nl}\mathbf{\Delta}_{im}\mathbf{\Delta}_{jn} \\
&= \frac{1}{D\mu} (\mathbf{\Delta}^2)_{ik}(\mathbf{\Delta}^2)_{jl} \\
&= [\partial_{ij}\mathcal{T}^*_{kl}](\mu \mathbf{1}_{\text{mat}}),
\end{aligned}
$$

where the last equality follows from the idempotency of $\mathbf{\Delta}$ and from Eq. (12). When $\mu = 1$, this proves Eq. (8). □

2 Datasets

Benchmark prokaryotic datasets. We developed and tested DiffPaSS using joint MSAs extracted from a dataset composed of 23,632 cognate pairs of histidine kinases (HK) and response regulators (RR) from the P2CS database [38, 39], paired using genome proximity, and previously described in Bitbol et al. [12], Bitbol [14]. Our focus is on pairing interaction partners among paralogs within each species. Pairing is trivial for a small number of species comprising only one pair of sequences. Hence, these species were discarded from the dataset. The average number of pairs per species in the resulting dataset is 11.0.

From this benchmark dataset of known interacting pairs, we extract paired MSAs of average depth 50, 100, 200, 300, 400, 500, 750 or 1000, constructed by selecting all the sequences of randomly sampled species from the full dataset. Each depth bin contains at least 200 MSAs. More precisely, for a target MSA depth $\overline{D} = 50$, 100, 200, 300, 400, 500, 750 or 1000, we add randomly sampled complete species one by one; if the first m species (but no fewer) give an MSA depth $D \geq 0.9\overline{D}$, and the first $n \geq m$ species (but no more) give $D \leq 1.1\overline{D}$, then we select the first k species in our final MSA, with k picked uniformly at random between m and n.

Eukaryotic complexes. We consider 15 heteromeric eukaryotic targets whose structures are not in the training set of AFM with v2 weights, already considered in Lupo et al. [32].

T-cell receptor paired CDR3α-CDR3β data. We downloaded the full VDJDb database [55] and removed all entries where only a single TCR chain is available. For each epitope, we removed duplicate TCRs (defined at the level of α/β) and retained only epitopes for which at least 100 and no more than 10,000 sequences were available. Patient and study metadata from the database is used to define "groups" that permutations are to be restricted to, and that play the part of species in our paralog matching problem. Our final dataset contains 22 sets of CDR3α-CDR3β sequence collections of highly variable total size (ranging from 103 to 1894 sequence pairs) and mean group size, for which the ground-truth matchings are known.

3 General points on AlphaFold-Multimer (AFM)

For all structure prediction tasks, we use the five pre-trained AFM models with v2 weights [21]. We use full genomic databases and code from release v2.3.1 of the official implementation in https://github.com/deepmind/alphafold. We use no structural templates, and perform 3 recycles for each structure, without early stopping. We relax all models using AMBER.

We pair the same subset of pairable sequence retrieved by AFM as Lupo et al. [32], and refer to Lupo et al. [32, Table S1] for details on the pairable MSAs. For all structures, we use the query protein pair as fixed context for DiffPaSS.

The AFM confidence score is defined as $0.8 \cdot \text{iptm} + 0.2 \cdot \text{ptm}$, where iptm is the predicted TM-score in the interface, and ptm the predicted TM-score of the entire complex [21].

4 Supplementary figures

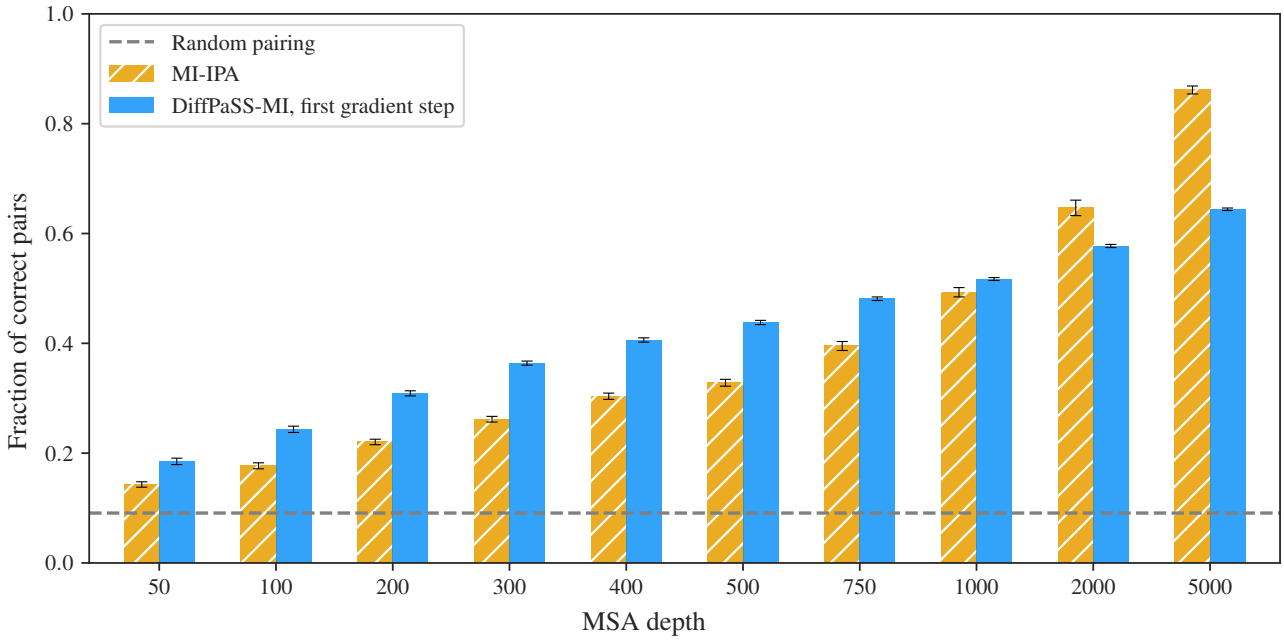

Figure S1: **DiffPaSS pairing performance after the first gradient step.** Pairing performance by DiffPaSS is shown after the first gradient step and compared to MI-IPA [14] on HK-RR MSAs with various depths. Same as in Fig. 2 but restricting to the first gradient step. Here, our procedure, after only one gradient step, is competitive with MI-IPA [14], for approximate maximization of the same discrete score $\mathscr{S}_{\mathrm{MI}}$.

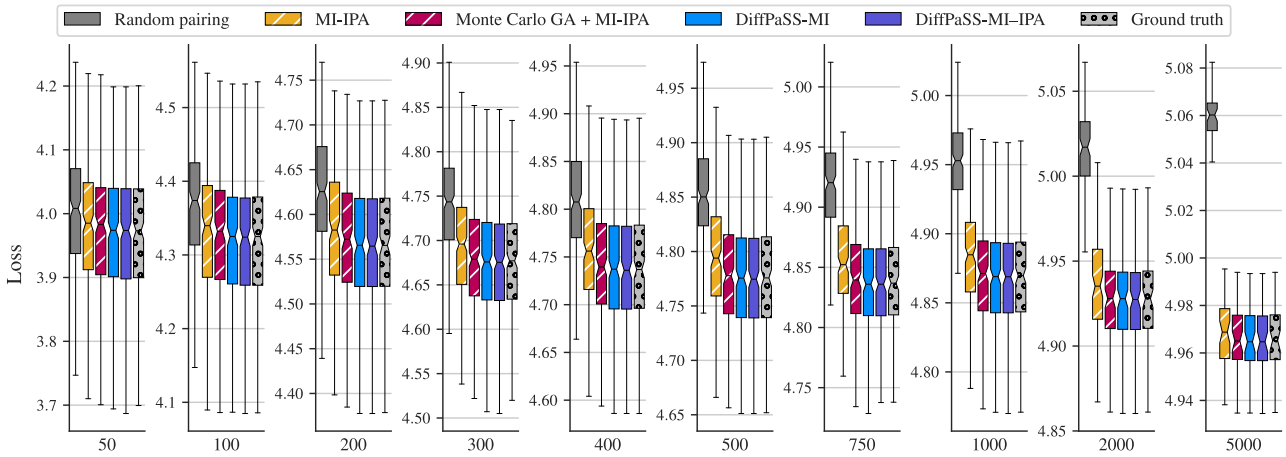

Figure S2: **Inter-chain two-body entropy losses for random, predicted and ground-truth pairings.** The distributions of the inter-chain two-body entropy losses are shown for several methods and several MSA depths, on the HK-RR dataset. Same MSAs as in Fig. 2.

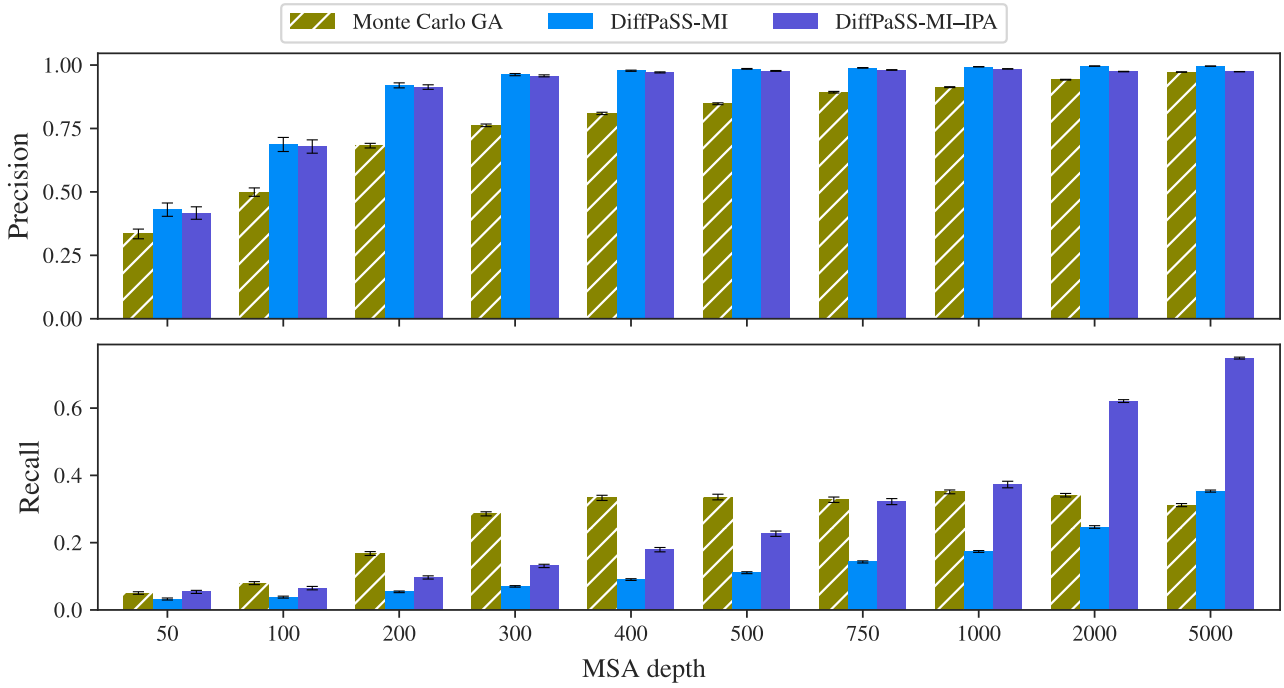

Figure S3: **Robust pairs.** Precision and recall for the robust pairs found by the Monte Carlo GA plus MI-IPA method (using 20 independent GA runs, as in [15]), DiffPaSS-MI, and DiffPaSS-MI–IPA with $N_{\mathrm{IPA}} = 3$ for several MSA depths, on the HK-RR dataset. Same MSAs as in Fig. 2.

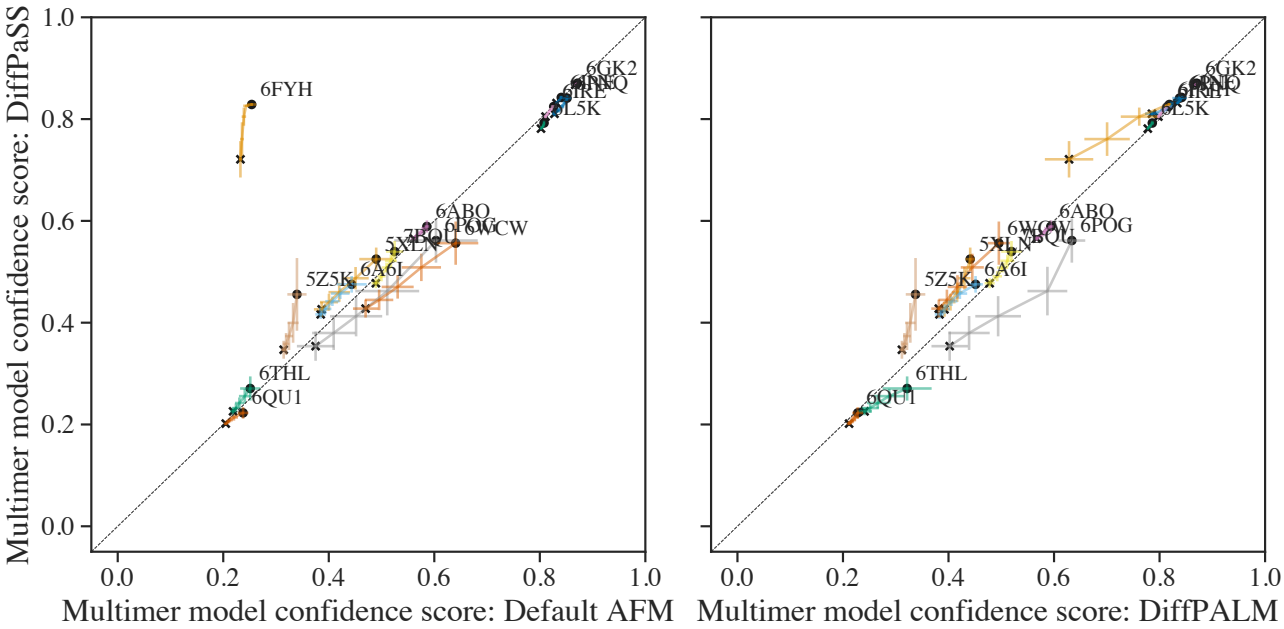

Figure S4: **AlphaFold-Multimer confidence scores on eukaryotic complexes using different MSA pairing methods.** Same comparisons as in 5, but showing the confidence scores for structure prediction by AFM instead of DockQ scores. See Appendix 3 for a definition of this confidence score. Results for Default AFM and DiffPALM-based pairing are as in Lupo et al. [32, Fig. S5].

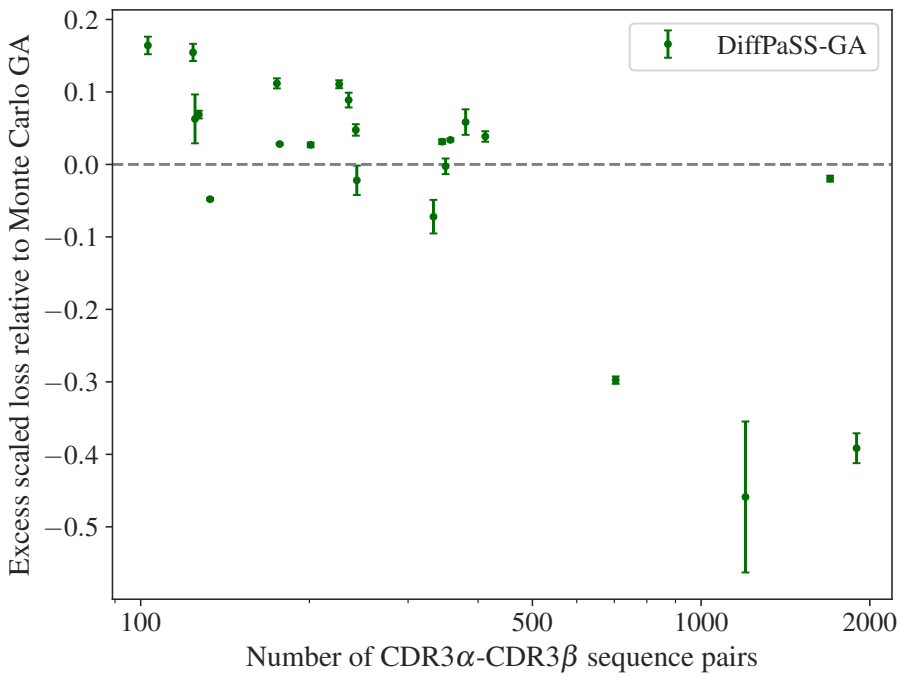

Figure S5: **DiffPaSS optimization and pairing quality on the CDR3α-CDR3β dataset.** (Normalized) graph alignment scores per epitope were computed in the paired CDR3α-CDR3β dataset described in Appendix 2 with both DiffPaSS-GA and the Monte Carlo GA from Ref. [15]. We plot the difference between the loss obtained by DiffPaSS-GA and by Monte Carlo GA versus the number of pairs for each epitope. Negative values indicate that Monte DiffPaSS-GA achieves a more successful optimization. Each point represents one of the epitopes in the dataset. Markers show mean values across 20 runs, and error bars are defined as $\sqrt{\sigma^2_{\text{MC}} + \sigma^2_{\text{DiffPaSS}}}$, where σ_{MC} (resp. σ_{DiffPaSS}) is the standard deviation obtained by Monte Carlo GA (resp. DiffPass-GA). Graph alignment scores are as defined in [15], but normalized by the number of CDR3α and CDR3β sequences to pair in each case.

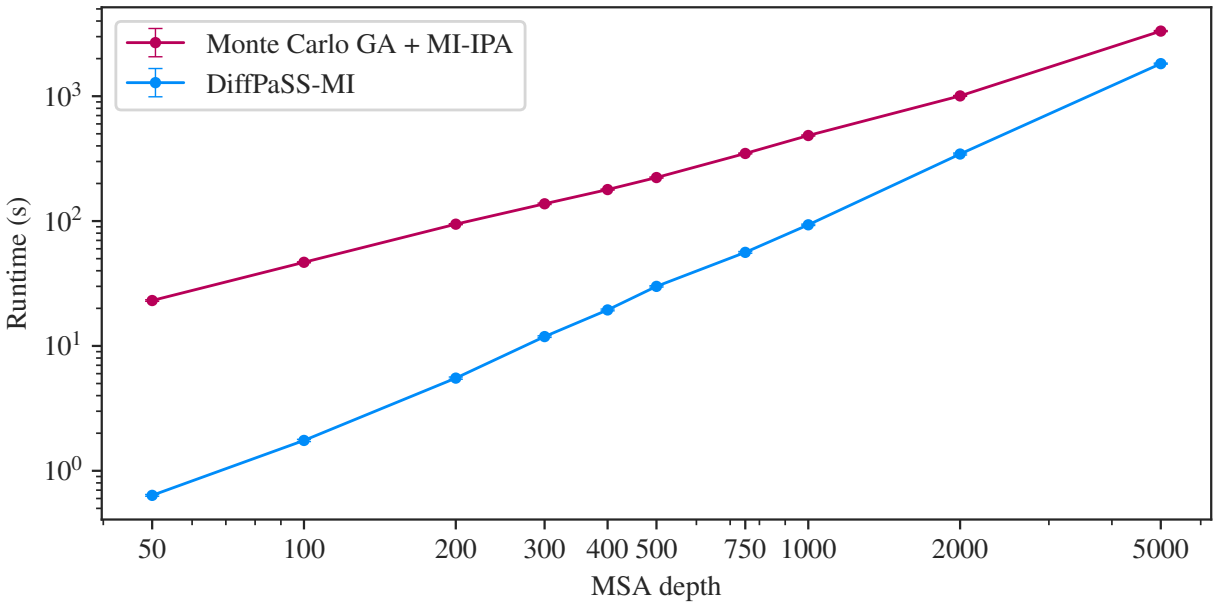

Figure S6: **Computational runtime.** Runtime comparison between DiffPaSS-MI and Monte Carlo GA + MI-IPA [15], on the same HK-RR MSAs as in Fig. 2. DiffPaSS was implemented in `PyTorch` v2.2.1, and run on an NVIDIA® GeForce RTX™ A6000 GPU. We used the original implementation of the combined Monte Carlo GA and MI-IPA method, in the Julia programming language (Julia v1.10.0). All 20 GA replicates were run in parallel on separate Intel® Xeon® Platinum 8360Y CPUs running at 2.4 GHz and then MI-IPA was run on one of these CPUs [OK?].