Electronic Supplementary Material (ESI) for Digital Discovery.

SUPPORTING INFORMATION:

Hybrid Computational-Experimental Data-Driven Design of Self-Assembling π-Conjugated Peptides

Kirill Shmilovich,[†] Sayak Subhra Panda,[‡] Anna Stouffer,[‡] John D. Tovar,[‡,¶] and Andrew L. Ferguson[*,†]

†*Pritzker School of Molecular Engineering, University of Chicago, Chicago, Illinois 60637*

‡*Department of Chemistry, Johns Hopkins University, Baltimore, Maryland 21218*

¶*Department of Materials Science and Engineering, Johns Hopkins University, Baltimore, Maryland 21218*

E-mail: andrewferguson@uchicago.edu

1 Supporting Methods

1.1 All-atom molecular dynamics (MD) simulation

We perform all-atom molecular dynamics simulations of multi-molecule X_n-4T-X_n systems using the GROMACS 2019.2 simulation suite.[1] The AMBER99SB-ILDN forcefield[2] is used to model each π-conjugated peptide. Partial charges for each molecule were obtained using the Restrained Electrostatic Potential (RESP) method[3,4] at the B3LYP/6-31G(d) level of theory implemented in Gaussian16.[5] Each molecule is prepared with any Asp or Glu residues within the oligopeptide wings in a fully-protonated state corresponding to the low-pH conditions where self-assembly is triggered in experiment. All-atom topologies are then generated using AnteChamber PYthon Parser InterfacE (ACPYPE)[6] wrapper for ANTECHAMBER package within the AmberTools20[7] tool set. Simulations are initialized by randomly placing 24 identical X_n-4T-X_n molecules in a $10\times10\times10$ nm^3 cuboidal simulation box with 3D periodic boundary conditions that is subsequently solvated using the TIP3P water model.[8] Each system is then equilibrated by first removing any forces larger than 1000 kJ/mol·nm with steepest decent energy minimization followed by short 100 ps NVT and NPT simulations where temperature and pressure are controlled using a velocity rescaling thermostat[9] at 300 K and Parerinello-Rahman barostat[10] at 1 bar, respectively. Initial velocities are sampled from a 300 K Maxwell-Boltzmann distribution and Newton's equations of motion are numerically integrated using the leapfrog algorithm with a 2 fs timestep.[11] Covalent bonds involving hydrogen atoms are constrained using the LINCS algorithm[12] and a 1.0 nm cutoff is used to smoothly shift Lennard-Jones interactions to zero. Electrostatics are treated using particle mesh Ewald (PME)[13] with an initial 0.16 nm Fourier grid spacing and a 1.0 nm real space cutoff, with these optimized for performance during runtime. Following the steepest descent, NVT, and NPT equilibration, 200 ns production runs are performed in the NPT ensemble after which time we notice the measured average number of contacts, κ, and the radius of gyration, R_g, of the π-conjugated peptide nanoaggregate are observed to plateau

(Fig. S1). Snapshots of each production run simulation frame are extracted and saved every 1 ps for analysis. All calculations are performed on a single NVIDIA Tesla V100 GPU achieving speeds of ∼130 ns/day.

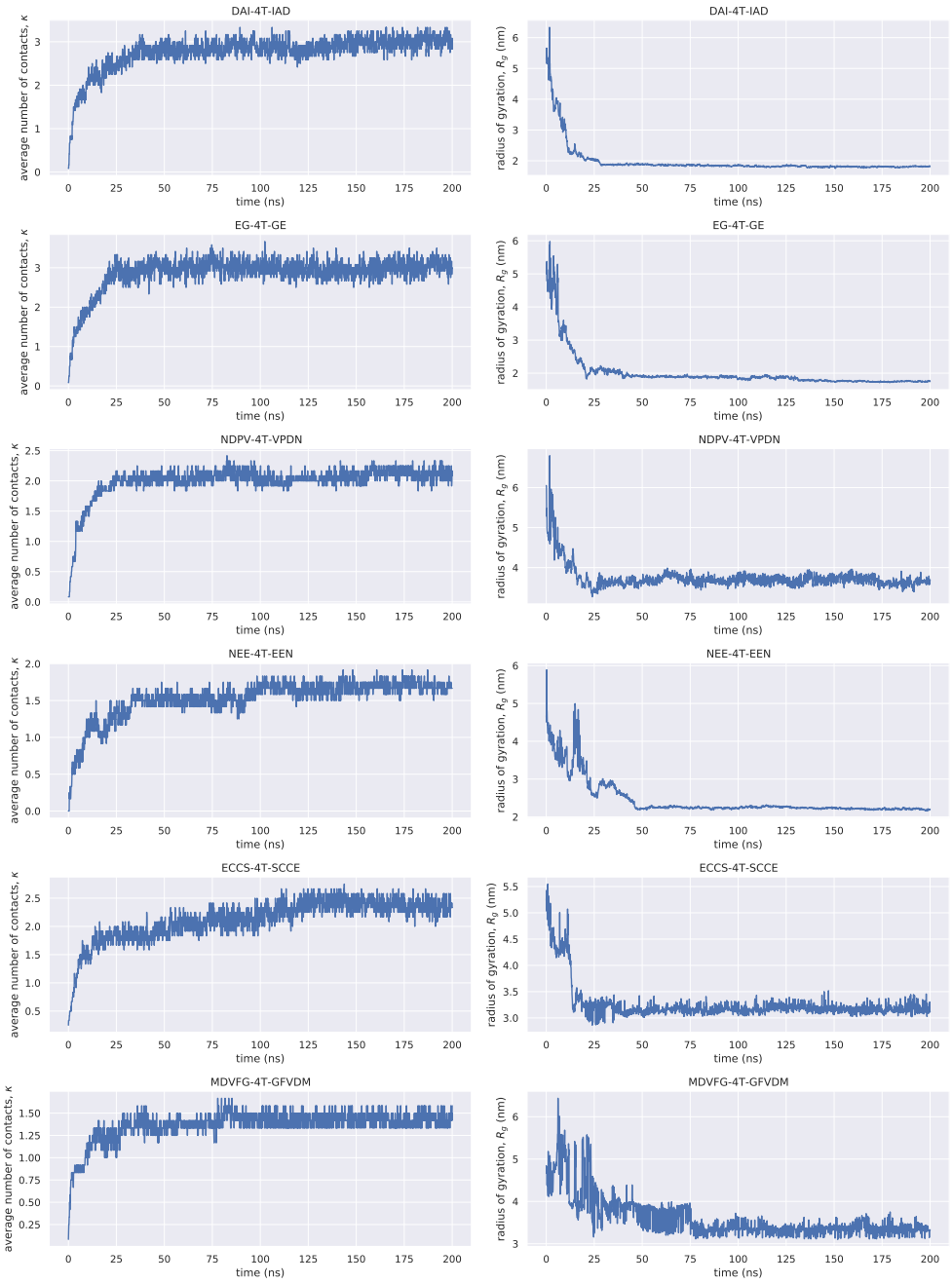

Figure S1: Convergence of structural metrics for π-conjugated peptide self-assembly during MD simulation. For six representative π-conjugated peptide systems we track the evolution of the average number of contacts, κ, (left panels) and the radius of gyration, R_g, (right panels) during the 200 ns MD simulation and observe κ and R_g to both plateau after $\sim$100-150 ns. As the molecules self-assemble from their initial monodisperse state κ steadily increases reflecting progressive association of π-conjugated peptides, while R_g simultaneously decreases as the molecules collapse into their terminal nanoaggregate structure.

1.2 Active learning framework

To effectively navigate the high-dimensional and discrete space of π-conjugated peptides we employ a hybrid computational/experimental active learning screen to minimize the number of simulations and experiments required to discover and validate new high-performing X_n-4T-X_n moelcules. The premise of this combined search is to leverage high-throughout simulation data as a cheaply available low-fidelity data stream that broadly samples the design space helping to identify more- and less-promising regions to direct experimental testing and characterization. A schematic illustration of integrated computational-experimental active learning workflow is presented in Fig. 1.

Independent active learning loops are maintained for the computational and experimental workflows that are executed asynchronously and in parallel. Each active learning loop requires the definition of a fitness function that defines the desirability of each candidate molecule. Fitness measurements collected by computation and experiment are used to train supervised regression models that serve as a surrogate predictors of the fitness of untested candidates. Active learning proceeds by using these models to identify the next best molecules to consider within the experimental and computational screens and define an iterative cycle of prediction/measurement and data-driven model building. The computational screening loop operates autonomously, wherein the computational predictions are used to fit the data-driven surrogate models and power the cycle. The experimental pipeline, however, integrates both experimental measurements and computational predictions. In doing so, the abundant computational data is used to supplement the sparse experimental data and produce a higher accuracy surrogate model than would be available from the experimental data stream alone. The ultimate goal of the hybrid computational/experimental screening process is to discover and experimentally validate X_n-4T-X_n molecules with unprecedentedly large values of λ indicative of exceptional in-register π-stacking and H-type character that is a prerequisite for supramolecular electronic delocalization and emergent optical and electronic functionality.

We now proceed to present the details of the four principal steps involved in the active learning protocol (i) definition of fitness functions, (ii) learning a smooth, low-dimensional embedding of the molecular design space, (iii) training of surrogate sequence-property models, and (iv) Bayesian optimization selection of candidate molecules for subsequent rounds of computational/experimental screening.

Code and data for the different components of the active learning framework is publicly available via Zenodo at DOI:10.5281/zenodo.5048398.[14] Our implementation makes use of open source Python libraries, including: PyTorch[15] version `1.5.0`, CUDA version `10.1`, PyTorch Geometric[16] version `1.4.3`, NetworkX[17] version `2.4`, GPy[18] version `1.9.9`, and NumPy[19] version `1.18.2`.

1.3 Step 1: Definition of fitness functions

To perform active learning requires defining a fitness function that is used to evaluate the quality of a candidate molecule such that this quantity can be optimized as a function of a continuous low-dimensional molecular representation. While the quality measures may differ between the experimental and computational active learning pipelines, provided the fitness function used in simulation is correlated with and can serves as a (possibly noisy) low-fidelity proxy for the experimental fitness function, we can transfer of information from the voluminous simulation data to the sparse and expensive-to-collect experimental data in order to more accurately direct the experimental active learning search.

1.3.1 Computational fitness

While electronic structure calculations are the gold standard for evaluating optoelectronic functionality in π-conjugated peptide assemblies, these calculations become prohibitively expensive at the scale of multi-molecule assemblies. To enable higher-throughput calculations we perform all-atom classical molecular dynamics simulations to access to the requisite time and length scales to observe self-assembly and nanoaggregate formation. From these simu-

lations we develop a structural proxy for optoelectronic activity related to the propensity of molecules to form pseudo-1D lineally stacked structures in simulation. To quantify the extent of pseudo-1D nanoaggregates formation we measure both the average number of contacts per molecule, κ, and the radius of gyration, R_g, of the molecular self-assembly in simulation. We previously employed a similar protocol for evaluating fitness of π-conjugated peptides with fixed-length DXXX-OPV3-XXXD oligopeptide wings from coarse-grained molecular simulation where we maximized κ alone .[20] In this work we consider multiple objectives where maximizing κ supports the formation of structures with a high degree of intermolecular π-π stacking between π-cores while simultaneously maximizing R_g encourages the formation of more elongated structures targeting the formation of linearly arranged pseudo-1D nanoaggregates.

To calculate the average number of contacts per molecule from a simulation frame at time t requires defining both a distance metric for calculating the distance between pairs of molecules and a criterion for specifying if two molecules should be considered in contact. We adopt the so-called "optical distance" metric that we have previously employed in measuring asphaltene and DXXX-OPV3-XXXD aggregation[20–23] that in our application calculates the minimum distance between pairs of thiophene rings within the 4T π-cores between a pair of molecules a and b,

$$d_{a,b}^{optical} = \min_{i \in \text{core(a)}} \min_{j \in \text{core(b)}} r_{i,j}, \tag{1}$$

where $r_{i,j}$ is the intermolecular distance between the center-of-mass of thiophene ring i of the 4T core within molecule a and thiophene ring j within the 4T core of molecule b. Pairs of molecules that satisfy the criterion $d_{a,b}^{optical} < r_{cut} = 0.7$ nm are then considered to be in contact, where the cutoff $r_{cut} = 0.7$ nm is motivated by the previously reported mean separation between good in-register stacks of DFAG-OPV3-GAFD π-conjugated peptides in simulation.[20,22,23] This metric ensures close contact of at least one thiophene ring between pairs of molecules that ultimately promotes the emergence of optoelectronic functionality via π electron overlap and electron delocalization.[22–24]

Given the definition in Eqn. 1, the instantaneous average number of contacts per molecule $\kappa(t)$ is simply calculated by enumerating all contacts based on pairwise distances for each π-conjugated peptide molecule within the simulation frame at time t and then averaging over all 24 molecules within the simulation box. The instantaneous radius of gyration $R_g(t)$ is also calculated analogously making use of the Python MDTraj trajectory analysis library.[25] The scalar-valued objectives used in our computational search $\kappa^{(k)}$ and $R_g^{(k)}$ for a simulated molecule k in our X_n-4T-X_n design space are subsequently calculated by time averaging,

$$\kappa^{(k)} = \overline{\kappa(t; (X_n\text{-4T-}X_n)^{(k)})}, \tag{2}$$

$$R_g^{(k)} = \overline{R_g(t; (X_n\text{-4T-}X_n)^{(k)})}, \tag{3}$$

where the over bar represents a time average over the terminal 50 ns of the 200 ns simulation. Uncertainties quantified as standard errors in κ and R_g are estimated by block averaging the terminal 50 ns in five contiguous 10 ns blocks. Without *a priori* knowledge for the correct balance of κ and R_g within a scalar optimization the computational active learning pipeline performs a multi-objective optimization to estimate the full κ-R_g Pareto frontier containing all candidate X_n-4T-X_n molecules that are not strictly dominated by any other molecule in both κ and R_g.[26]

1.3.2 Experimental fitness

Our goal ultimately is to discover and identify molecules within the X_n-4T-X_n design space that self-assemble into pseudo-1D nanoaggregates possessing superior optoelectronic activity. We experimentally quantify the quality of the self-assembled aggregates using photophysical spectroscopy. Kasha et al.[27] detailed the use of UV-vis spectroscopy to determine the types of assemblies: H-type or co-facial assembly is usually characterized by the blue-shift in absorption maxima upon going from monomeric to assembled condition whereas J-type or side-wise assembly is characterized by the red-shift in absorption maxima from monomeric

to assembled state. We previously observed that X_n-4T-X_n triblock molecules show H-type assembly with a blue-shifted UV-vis maxima upon acid-triggered assembly.[28,29] Larger blue shifts are correlated with improved intermolecular π-stacking between the 4T cores, and it is our objective to discover peptide wing sequences that promote high H-type character within the self-assembled nanostructures by maximizing the measured spectral shift λ. Uncertainties in λ are quantified by standard errors in repeated measurements performed at varying concentrations.

1.4 Step 2: Learning a smooth, low-dimensional embedding of the molecular design space

Part of the challenge of inverse molecular design is addressing large molecular design spaces that are inherently discrete and high-dimensional in nature. Deep representational learning enables us to learn low-dimensional representations of molecules that embeds them as continuous and real-valued vectors within a low-dimensional latent space. These low-dimensional representations tend to favor the training of robust surrogate models that predict the fitness of untested molecules based on the subset of molecules that have undergone simulations and/or experimental characterization. Their smooth and continuous nature are naturally suited to optimization wherein the surrogate model predictions are passed to Bayesian optimization routines that select batches of molecules predicted to be most promising for the next round of computational/experimental screening. In this work we generate low-dimensional molecular representations of our X_n-4T-X_n design space using regularized autoencoder (RAEs)[30] as a deterministic adaptation of the popular variational autoencoder (VAE) architecture.[31]

1.4.1 Representing X_n-4T-X_n molecules as linear graphs

Since all members of the X_n-4T-X_n family possess the same 4T π-core, the only differentiating factor between molecules is the X_n oligopeptide wing. Accordingly, we represent

and expose each molecule to the RAE as a linear graph of amino acids reflecting the primary structure of the X_n oligopeptide. Representing molecules as abstract graphs that model a collection of objects (nodes) and their relations (edges) has recently been a popular choice among practitioners applying deep learning to molecular design.[32–34] By using graph-structured data we leverage the powerful capabilities of graph neural networks to effectively incorporate relational information between elements while naturally remaining invariant to atomic permutations.[35] Henceforth, by using a graph neural network encoder within our RAE we can ensure to learn a latent space embedding of our X_n-4T-X_n design space that respects permutational symmetries present in the X_n oligopeptide wing.

Each X_n-4T-X_n molecule k is represented as a linearly connected graph $G^{(k)}$ with $(n+1)$ nodes comprising the n amino acids in the X_n oligopeptide wing along with a dummy node endowing directionality to the sequence of amino acids (e.g., differentiating DVGA-4T-AGVD from AGVD-4T-DVGA) (Fig. S2). These $(n+1)$ nodes are then sequentially connected with n edges to yield the linear graph representation. Each graph additionally includes self-loops where another $(n+1)$ edges are added connecting each node to itself. The nodes and edges of each graph are featurized using the Amino Acid Index (AAindex) database.[36] The AAindex database contains 566 scalar values related to individual amino acid physio-chemical properties along with 141 amino acid mutation matrices and pairwise contact potentials. A subset of 553/566 node features are assigned to each amino acid node corresponding to the features present for all the amino acids in our design library. Similarly, 135/141 features are assigned to each edge that connects pairs of amino acids corresponding to the values present for all amino acids pairs within the mutation matrices and pairwise contact potentials. We discarded 13 of the single amino acid features and six pairwise amino acid features because these descriptors were missing values for at least one amino acid. We elect to take an unbiased approach exposing all available node and edge features from AAindex in our representation, rather than hand-selecting a subset of features that we believe would be relevant to our application. The dummy node is always appended to

the beginning of the graph and is assigned a zero-padded vector as its node features (e.g. the linear graph with the sequence dummy-A-V-G corresponds to the π-conjugated peptide AVG-4T-GVA). All edges that connect to this dummy node are also set as zero-padded vector. Computationally the dummy node may be interpreted as corresponding to a start token of the sequence, while physically it can be interpreted as corresponding to the location of the oligopeptide wing C-terminus.

Each graph-structured object is then compactly represented by three matrices $G^{(k)} = G(N^{(k)}, A^{(k)}, E^{(k)})$ where: $N^{(k)} \in \mathbb{R}^{(n+1)\times 553}$ is a matrix of node features where $N_i^{(k)}$ corresponds to the node features for ith node within the graph, $A^{(k)} \in \mathbb{R}^{(n+1)\times(n+1)}$ is an adjacency matrix capturing the connectivity of the nodes where $A_{i,j}^{(k)} = 1$ indicates an edge is present that connects node i to node j, and $E^{(k)} \in \mathbb{R}^{(n+1)\times(n+1)\times 135}$ is the matrix of edge features where $E_{i,j}^{(k)}$ are the features for the edge connecting node i to node j. At training time we standardize the sizes of the node feature matrix $N^{(k)} \in \mathbb{R}^{6\times 553}$, the adjacency matrix $A^{(k)} \in \mathbb{R}^{6\times 6}$, and the edge feature matrix $E^{(k)} \in \mathbb{R}^{6\times 6\times 135}$ for all our graph-structured data $G^{(k)} = G(N^{(k)}, A^{(k)}, E^{(k)})$ to have the maximum possible number of rows/columns needed to accommodate the largest possible graphs in our dataset which contain six nodes. For graphs with less than six nodes the additionally added rows/columns in $N^{(k)}$, $A^{(k)}$ and $E^{(k)}$ corresponding to non-existent nodes in the graph are padded with zeroes. We note that operationally the edge feature matrix is more compactly represented as $E^{(k)} \in \mathbb{R}^{5\times 135}$ because of the linear nature of our graphs. For notational convenience and generality we elect to depict $E^{(k)} \in \mathbb{R}^{6\times 6\times 135}$ in our explication here which allows for arbitrary graph connectivity.

1.4.2 Regularized autoencoder (RAE)

The architecture of our regularized autoencoder (RAE) for embedding X_n-4T-X_n molecules is composed of two parts: (i) an encoder that converts graph-structured representations of molecules $G^{(k)}$ into their corresponding d-dimensional latent space embedding $\text{Encoder}(G^{(k)}) = \mathbf{z}^{(k)} \in \mathbb{R}^d$, and (ii) a decoder that attempts to reconstruct $\hat{G}^{(k)}$ from their latent space vector

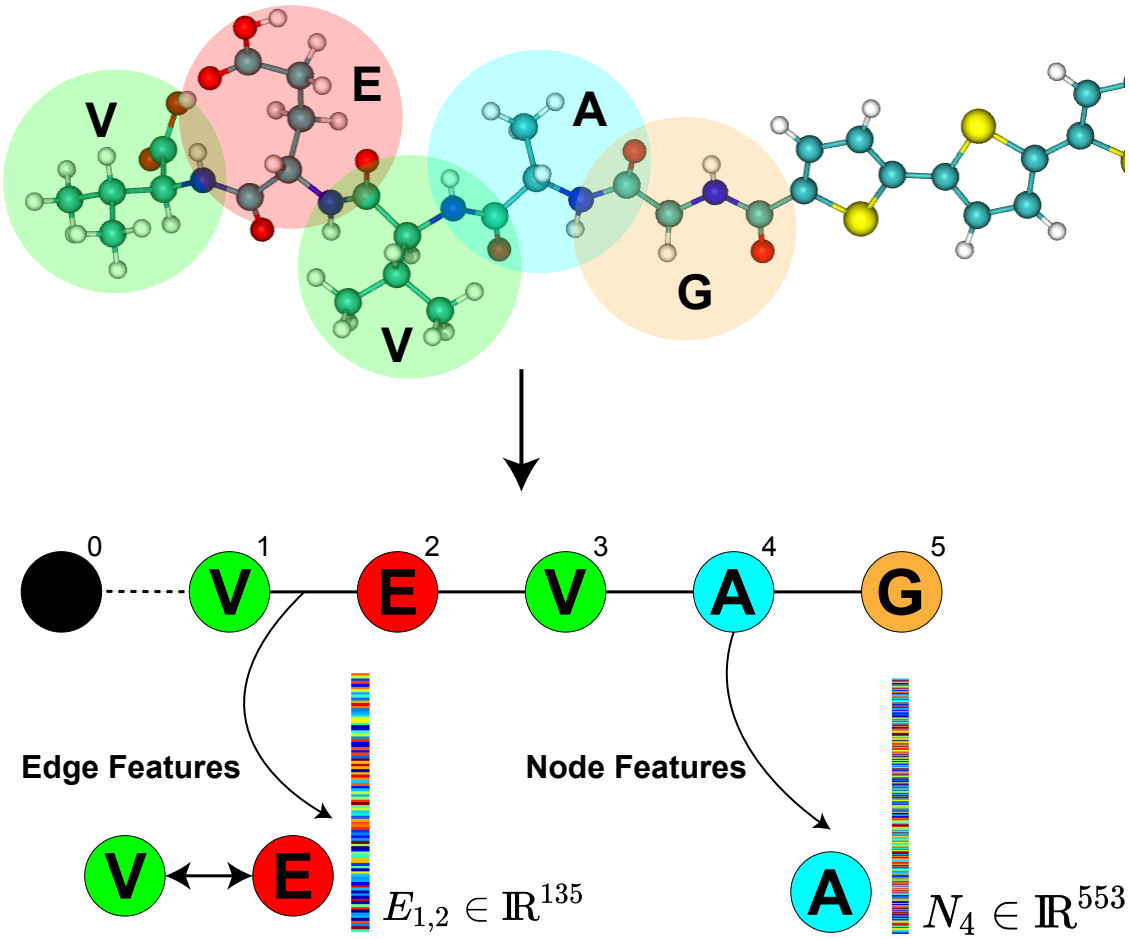

Figure S2: Representing X_n-4T-X_n molecules as linear amino acid graphs. Each molecule in our design space is differentiated by the X_n oligopeptide sequence attached to the 4T core. The sequence of amino acids is converted into a linear graph where the nodes N_i are amino acids and edges $E_{i,j}$ reflect the connectivity of the amino acids. By appealing to the AAindex database,[36] each node is featurized using $N_i \in 553$ scalar single amino acid properties, while the edges are featurized according to $E_{i,j} \in 135$ scalar values extracted for mutation matrices and pairwise contact potentials corresponding to pairwise amino acid properties. A dummy node with zero-padded node and edge features is appended to the beginning of the sequence endowing directionality to the oligopeptide wing.

representation Decoder($\mathbf{z}^{(k)}$) = $\hat{G}^{(k)}$ (Fig. S3). The encoder uses a message passing neural network (MPNN) nearly identical to that used by Glimer et al.[37] for quantum mechanical property prediction of small organic molecules, while the decoder is a partially autoregressive model that first reconstructs the node features $\hat{N}^{(k)}$ from $\mathbf{z}^{(k)}$ alone and subsequently predicts the adjacency matrix $\hat{A}^{(k)}$ using both the reconstructed node features $\hat{N}^{(k)}$ and the

latent code $\mathbf{z}^{(k)}$. We elect not to reconstruct the edge features $\hat{E}^{(k)}$ because the identity of each molecule k in our X_n-4T-X_n design space is fully specified from the reconstructed node features $\hat{N}^{(k)}$ and the reconstructed adjacency matrix $\hat{A}^{(k)}$ alone. Regardless, the encoder is still capable of utilizing information captured in the relations defined within the edge features $E^{(k)}$ when constructing the latent code $\mathbf{z}^{(k)}$ that is ultimately used by the decoder in reconstructing the node features $\hat{N}^{(k)}$ and adjacency matrix $\hat{A}^{(k)}$.

1.4.3 Encoder

The encoder is used to derive a d-dimensional permutation invariant continuous latent vector $\mathbf{z}^{(k)} = \text{Encoder}(G^{(k)})$ from the graph-structured representation $G^{(k)}$ of each X_n-4T-X_n molecule using a message passing neural network (MPNN) architecture introduced by Glimer et al.[37] Message passing is performed by sequentially updating the hidden state $h_i^{(t+1)}$ of each node i using messages $m_i^{(t+1)}$ that accumulate information from the neighboring nodes,

$$m_i^{(t+1)} = \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \Psi_\Theta(E_{i,j}) \cdot h_j^t, \tag{4}$$

$$h_i^{(t+1)} = \text{GRU}(m_i^{(t+1)}, h_i^t), \tag{5}$$

where $\mathcal{N}(i)$ are all the neighbors of node i and the $\cdot$ in Eq. 4 represents matrix multiplication. A single dense layer $\Omega_\theta : \mathbb{R}^{553} \mapsto \mathbb{R}^{d_h}$ transforms the node features N_i for node i into the initial d_h-dimensional hidden state representation $\Omega_\theta(N_i) = h_i^{t=0} \in \mathbb{R}^{d_h}$. The multi-layer perceptron (MLP) $\Psi_\Theta : \mathbb{R}^{135} \mapsto \mathbb{R}^{d_h \times d_h}$ in Eqn. 4 maps the initial edge features $E_{i,j}$ to a matrix of dimension $d_h \times d_h$. A GRU cell[38] is used in Eq. 5 as in the gated graph neural network variant of message passing[39] to update the hidden state of each node $h_i^{(t+1)}$ based on the message $m_i^{(t+1)}$ propagated from the neighbors of node i. Message passing is performed to update the hidden states of each node for a total of $T = 5$ steps after which point we use the set2set model of Vinyals et al.[40] to aggregate the final hidden states $\{h_i^T\}_{i \in G^{(k)}}$ and

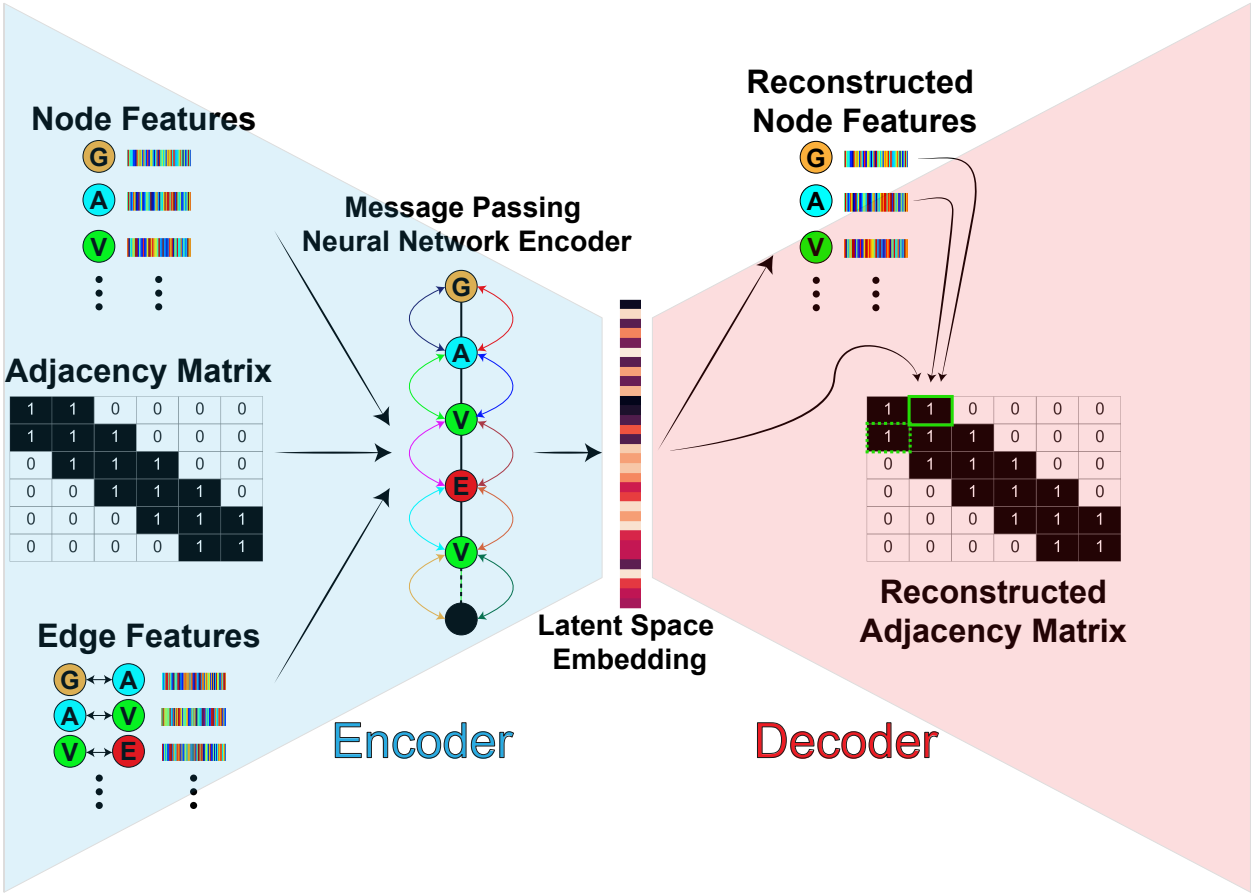

Figure S3: Regularized autoencoder (RAE) architecture used to generate the chemical space embedding of the X_n-4T-X_n design space. Each molecule k is exposed to the RAE as a graph structured object $G^{(k)} = G(N^{(k)}, A^{(k)}, E^{(k)})$ defined by node features $N^{(k)}$, an adjacency matrix $A^{(k)}$, and edge features $E^{(k)}$. A message passing neural network (MPNN) encoder is then used to compress this high-dimensional representation into a fixed-sized real-valued vector representation $\text{Encoder}(G^{(k)}) = \mathbf{z}^{(k)}$. The decoder then attempts to reconstruct the identity of the input from the latent code $\mathbf{z}^{(k)}$ by reconstructing the node features and adjacency matrix $\text{Decoder}(\mathbf{z}^{(k)}) = (\hat{N}^{(k)}, \hat{A}^{(k)})$, which are alone sufficient to specify the identity of the input molecule in absence of the edge features. The RAE network is trained end-to-end by minimizing the loss given in Eqs. 10, 11 and 12 that balances reconstruction accuracy and regularization terms to ensure the latent space remains smooth and proximally embeds chemically similar molecules.

readout a graph-level representation,

$$h_{G^{(k)}} = \text{set2set}(\{h_i^T\}_{i \in G^{(k)}}). \tag{6}$$

The set2set model is an expressive global pooling operator that is invariant to the order of the inputs and uses and iterative attention mechanism to produce a continuous feature vector for each graph $h_{G^{(k)}}$. Finally, this representation is processed by a final MLP Φ_Θ to ultimately yield $\mathbf{z}^{(k)}$ defining the latent representation within the chemical space embedding for the molecule defined by the graph-structured representation $G^{(k)}$,

$$\mathbf{z}^{(k)} = \Phi_\Theta(h_{G^{(k)}}). \tag{7}$$

1.4.4 Decoder

The decoder attempts to reconstruct the node features $\hat{N}^{(k)}$ and adjacency matrix $\hat{A}^{(k)}$ from the latent space embedding vector derived from the encoder $(\hat{N}^{(k)}, \hat{A}^{(k)}) = \text{Decoder}(\mathbf{z}^{(k)}) = \text{Decoder}(\text{Encoder}(G^{(k)}))$ using an autoregressive approach first reconstructing the node features $\hat{N}^{(k)}$ from $\mathbf{z}^{(k)}$ alone, which are then used to reconstruct the adjacency matrix $\hat{A}^{(k)}$ from $\hat{N}^{(k)}$ and $\mathbf{z}^{(k)}$ together. The first step in our decoding process is using an MLP $\psi_\Theta : \mathbb{R}^d \mapsto \mathbb{R}^{6\times 553}$ that directly projects latent vectors $\mathbf{z}^{(k)}$ into reconstructed node features $\hat{N}^{(k)}$,

$$\hat{N}^{(k)} = \psi_\Theta(\mathbf{z}^{(k)}). \tag{8}$$

From this reconstruction $\hat{N}^{(k)}$ we define the the function ϕ_Θ composed of MLPs that assigns entries into each component of the reconstructed adjacency matrix $\hat{A}^{(k)}_{i,j}$ which is a function of the predicted node features $\hat{N}^{(k)}_i$, $\hat{N}^{(k)}_j$, and the latent code $\mathbf{z}^{(k)}$. Further, because we know the adjacency matrix must be symmetric we construct ϕ_Θ such that the function is invariant to the order of $\hat{N}^{(k)}_i$ and $\hat{N}^{(k)}_j$ enforcing explicitly that $\hat{A}^{(k)}_{i,j} = \hat{A}^{(k)}_{j,i}$,

$$\hat{A}^{(k)}_{i,j} = \phi_\Theta(\{\hat{N}^{(k)}_i, \hat{N}^{(k)}_j\}, \mathbf{z}^{(k)}). \tag{9}$$

The internals of the ϕ_Θ function are primarily composed of three MLPS: (i) an MLP $\phi^{(i)}_\Theta : \mathbb{R}^{553} \mapsto \mathbb{R}^{d_n}$ that maps reconstructed node features $\hat{N}^{(k)}_i$ into a d_n-dimensional intermediate

representation, (ii) an MLP $\phi_{\Theta}^{(ii)} : \mathbb{R}^{d} \mapsto \mathbb{R}^{d_z}$ that maps the latent vector $\mathbf{z}^{(k)}$ into a d_z-dimensional intermediate representation, and lastly an MLP (iii) $\phi_{\Theta}^{(iii)} : \mathbb{R}^{d_n+d_z} \mapsto \mathbb{R}^{1}$ that takes as input the concatenation $[\frac{\phi_{\Theta}^{(i)}(\hat{N}_i^{(k)})+\phi_{\Theta}^{(i)}(\hat{N}_j^{(k)})}{2} || \phi_{\Theta}^{(ii)}(\mathbf{z}^{(k)})]$, where $||$ is a concatenation operator, and outputs a scalar value used as the entry into the reconstructed adjacency matrix $\hat{A}_{i,j}^{(k)}$. The addition operation between $\phi_{\Theta}^{(i)}(\hat{N}_i^{(k)})$ and $\phi_{\Theta}^{(i)}(\hat{N}_j^{(k)})$ used as input for $\phi_{\Theta}^{(iii)}$ ensures the function ϕ_{Θ} remains invariant to the order of $\hat{N}_i^{(k)}$ and $\hat{N}_j^{(k)}$ and strictly enforces that the reconstructed adjacency matrix remain symmetric $\hat{A}_{i,j}^{(k)} = \hat{A}_{j,i}^{(k)}$. Overall, the decoder reconstructs the node features and adjacency matrix $(\hat{N}^{(k)}, \hat{A}^{(k)}) = \text{Decoder}(\mathbf{z}^{(k)})$ from the latent code $\mathbf{z}^{(k)}$ extracted from the encoder which performs message passing on the graph-structured input $\text{Encoder}(G^{(k)}) = \mathbf{z}^{(k)}$. The reconstructions $(\hat{N}^{(k)}, \hat{A}^{(k)})$ fully specify the identity of molecule k within the X_n-4T-X_n design space and are used to train the RAE neural network in training procedures outlined in the next section.

1.4.5 RAE training

The primary difficulty in training encoder-decoder architectures on graph-structured data lies in the $n!$ valid permutations of a graph containing n nodes. While the encoder uses a MPNN that yields a latent space embedding that is insensitive to permutations, the decoder is incapable of reconstructing the precise input permutation because this information has been effectively "integrated-out" of the latent code. Many techniques have been proposed to remedy training procedures to account for this mismatch between input and output permutations. Winter et al.[41] train a *permuter* model alongside the encoder and decoder that learns probable reordering of the output graph from the order present in the input graph. Our approach is most similar to the GraphVAE approach by Simonovsky et al.[42] where they use approximate graph matching with soft discretization to approximate an alignment between input and output graphs when computing the reconstruction loss. Here, we perform exact graph matching by explicitly enumerating all valid permutations of each reconstructed graph and selecting the permutation that minimizes the reconstruction loss for each input-

output pair. While explicit graph matching is typically prohibitively expensive for large graphs, this approach becomes tractable in our case because the largest graph in our training dataset contains at most six nodes requiring us to consider a maximum of only $6! = 720$ permutations.

The RAE is trained by minimizing the loss $\mathcal{L}$ composed of three terms: the reconstruction loss $\mathcal{L}_{REC}$, the RAE loss $\mathcal{L}_{RAE}$, and the regularization loss term $\mathcal{L}_{REG}$,

$$\begin{aligned}
\mathcal{L}_{REC} &= \underset{\pi}{\text{argmin}}[\text{MSE}(P_\pi \hat{N}^{(k)}, N^{(k)}) + \lambda_{REC}\text{BCE}(P_\pi \hat{A}^{(k)} P_\pi^T, A^{(k)})] \\
&= \underset{\pi}{\text{argmin}}[\sum_{i=1}^{6}[\sum_{j=1}^{553}((P_\pi \hat{N}^{(k)})_{i,j} - N_{i,j}^{(k)})^2 - \\
&\lambda_{REC}\sum_{j=1}^{6}(A_{i,j}^{(k)} \log((P_\pi \hat{A}^{(k)} P_\pi^T)_{i,j}) + (1 - A_{i,j}^{(k)}) \log(1 - (P_\pi \hat{A}^{(k)} P_\pi^T)_{i,j})]], \qquad (10) \\
\mathcal{L}_{RAE} &= \frac{1}{2}||\mathbf{z}^{(k)}||_2^2 = \frac{1}{2}\sum_{i=1}^{d}(\mathbf{z}_i^{(k)})^2, \qquad (11) \\
\mathcal{L}_{REG} &= \mathcal{L}_{L2} = ||W_{\text{Decoder}}||_2^2, \qquad (12) \\
\mathcal{L} &= \mathcal{L}_{REC} + \lambda_{RAE}\mathcal{L}_{RAE} + \lambda_{REG}\mathcal{L}_{REG}, \qquad (13)
\end{aligned}$$

where $\{P_\pi \in \mathbb{R}^{6\times 6}\}_{\pi=1}^{6!}$ are all valid permutation matrices of size six indexed by π. $\text{MSE}(\hat{y}, y)$ and $\text{BCE}(\hat{y}, y)$ in Eq. 10 corresponds to the mean squared error and binary cross entropy between the reconstructions $\hat{y}$ and ground truth y, respectively. The hyperparameter λ_{REC} balances the reconstruction losses between the node feature matrix $N^{(k)}$ and the adjacency matrix $A^{(k)}$ in $\mathcal{L}_{REC}$, while λ_{RAE} and λ_{REG} controls the relative weights of the three losses that comprise $\mathcal{L}$. The RAE loss $\mathcal{L}_{RAE}$ in Eq. 11 computes the squared L2-norm of the d-dimensional latent vector outputted from the encoder $\mathbf{z}^{(k)} = \text{Encoder}(G^{(k)})$ and serves to restrict the size of the latent space while preventing unbounded optimization of $\mathbf{z}^{(k)}$ and stopping the network from otherwise simply memorizing the data by allocating distant latent space locations to each training point. Lastly, the regularization loss $\mathcal{L}_{REG}$ in Eq. 12 is a weight decay on all the parameters of the decoder W_{Decoder}. The purpose of this term is

to help promote a smooth latent space embedding by enforcing the decoder have a bounded Lipschitz constant resulting in points nearby one another in the latent space getting decoded into similar graph structures in the output data space. We note that this type of regularization enforcing a smooth embedding with a bounded Lipschitz constant is implicitly done when training variational autoencoders (VAEs) via the injection of a noisy input to the decoder performed during the reparameterization trick,[31] but must be constituted as an explicit component of the loss for RAE training. In Fig. S4 we qualitatively validate that our learned latent space represents a smooth encoding of X_n-4T-X_n molecules by presenting an example of latent space interpolation and confirming points nearby one-another in the latent space are encoded as chemically similar structures in data space. Along with weight decay Ghosh et al.[30] outline other methods for regularizing the decoder when training RAEs such as spectral normalization and gradient penalty, however in this work we elect to use weight decay due to its simplicity of implementation yet comparable performance and effectiveness compared to these other more complicated methods. Training of the RAE is performed using mini-batch gradient descent with the Adam optimizer.[43] A full list of training parameters and neural network specifications are presented in Table S1. The particular model specification used in this work was selected based on a hyperparameter search over the learning rate within the range [5×10^{-5}, 5×10^{-3}], the batch size within the range [16, 64], the intermediate hidden dimension d_h within the range [128, 1024], and the latent space dimension d within the range [2, 256]. The objective function was minimization of the loss while simultaneously assuring good utilization of the latent space dimensions. The latter criterion was assessed by performing a singular value decomposition of the latent space, estimating its effective dimensionality d^{eff} as the dimension at which approximately 85% of the variance is explained, and assuring that $d^{\text{eff}} \approx d$. In our application we do not perform a training/validation/testing split because the purpose of our RAE model is *not* as a predictive tool for out-of-sample inference, but rather we train our RAE model on the complete dataset of 694,982 unique π-conjugated peptides with the intention of learning a low-dimensional latent space embed-

ding of our discrete molecular design space for downstream application within our active learning workflow. The PyTorch machine learning framework is used to build and train the RAE model.[15]

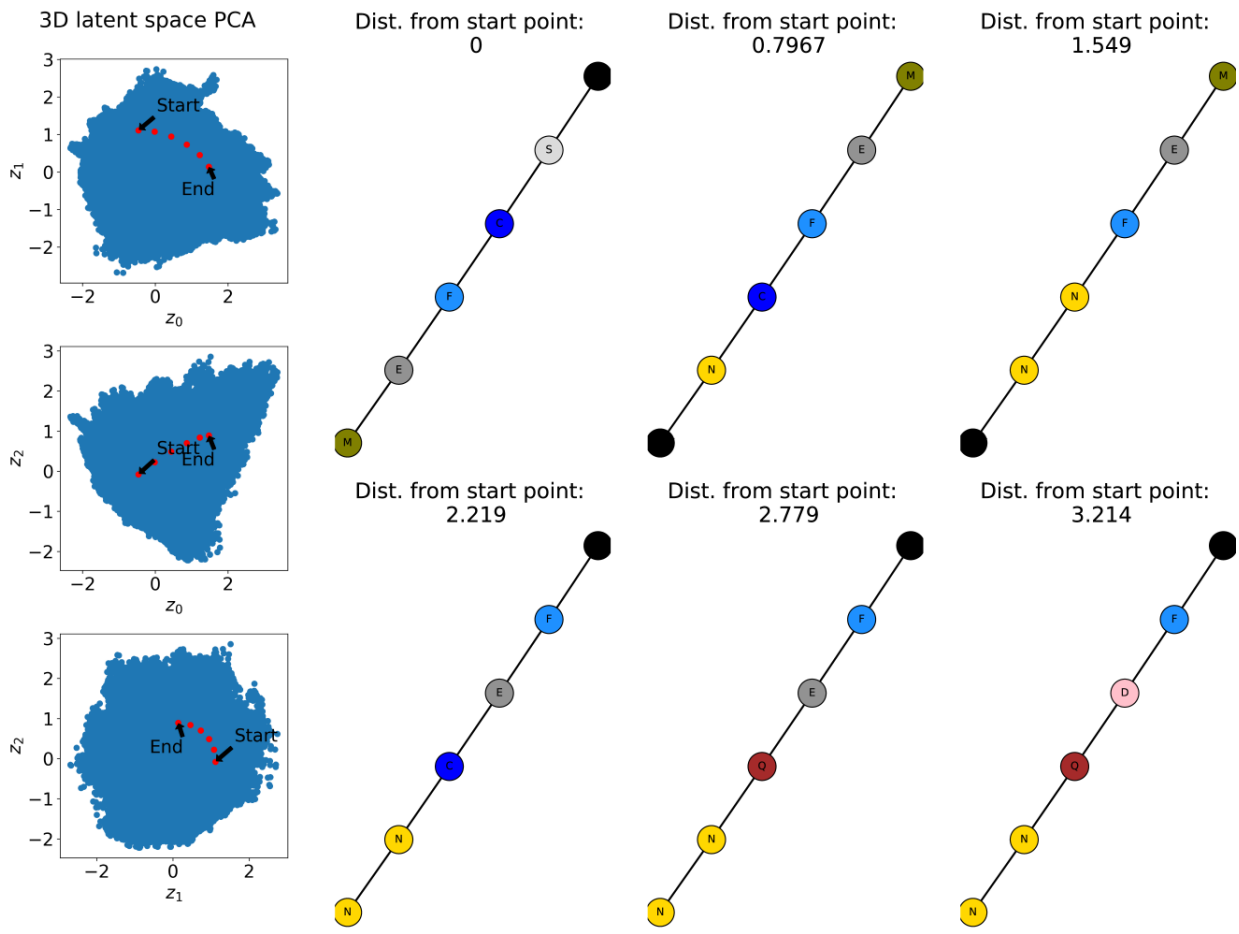

Figure S4: Representative example of regularized autoencoder (RAE) latent space interpolation. Blue points in the left three panels are 2D elevations of a 3D PCA projection of the full dimensional RAE learned latent space of all 694,982 X_n-4T-X_n molecules. Two random points within the latent space are selected and spherical linear interpolation[44] between the two points is performed to identify four interpolate locations shown as the red points within the 3D PCA projection. The corresponding latent space representations of each molecule nearest to each interpolate is selected and shown as a linear amino acid graphs in the right visualizations where the solid black node of each graph is the dummy node identifying the start of the X_n oligopeptide sequence. The values shown above each amino acid graph indicate the distance within the full-dimensional latent space of the interpolate from the starting point of the interpolation, such that a distance of zero corresponds to the start of the interpolation and the location of which is annotated within the left 3D PCA projection panels. Movements to nearby points within the latent space correspond to minor changes in the amino acid sequence within the data space, reflective of the smoothness of the latent space embedding where chemically similar molecules are embedded nearby one another within their latent space projections.

Table S1: Neural network hyperparamters. Description and specification of hyperparamters used to build the RAE neural network. The grammar used to specify the neural network architecture defines a sequence of operations performed on the input. For example, the sequence [Linear(10, 50), Leaky_ReLU, Linear(50, 100)] means a linear layer transforms a 10-dimensional input into a 50-dimensional output that is then acted upon by a Leaky_ReLU non-linearity and lastly a linear layer transforms these activations into the final 100-dimensional output.

Parameter	Description	Value/Specification
d	Latent space dimension	32
d_h	Intermediate hidden dimension in the encoder layers	128
d_n	Intermediate hidden dimension of transformed reconstructed node features in the decoder layers	512
d_z	Intermediate hidden dimension of transformed latent space representation in the decoder layers	512
Ω_Θ	Dense layer within the encoder that transforms node features N_i into the initial hidden states $h_i^{t=0}$	[Linear(553,d_h), Leaky_ReLU)]
Ψ_Θ	MLP within the encoder that transforms edge features $E_{i,j}$ into a $d_h \times d_h$ representation	[Linear(135,256), Leaky_ReLU, Linear(256,512), Leaky_ReLU, Linear(512,$d_h \times d_h$)]
Φ_Θ	MLP within the encoder that transforms output of set2set global pooling operator $h_{G^{(k)}}$ into latent space representation $\mathbf{z}^{(k)}$	[Linear($2d_h$,d_h), Leaky_ReLU, Linear(d_h,d)]

ψ_Θ	MLP within the decoder that transforms output the latent space representation $\mathbf{z}^{(k)}$ into reconstructed node features $\hat{N}^{(k)}$	[Linear(d,256), Leaky_ReLU, Linear(256,512), Leaky_ReLU, Linear(512,6×553), Sigmoid]
$\phi_\Theta^{(i)}$	MLP within the decoder that transforms reconstructed node features $\hat{N}_i$ into a d_n-dimensional intermediate representation	[Linear(553,256), Leaky_ReLU, Linear(256,d_n), Leaky_ReLU]
$\phi_\Theta^{(ii)}$	MLP within the decoder that transforms the latent space representation $\mathbf{z}^{(k)}$ into a d_z-dimensional intermediate representation	[Linear(d,256), Leaky_ReLU, Linear(256,d_z), Leaky_ReLU]
$\phi_\Theta^{(iii)}$	MLP within the decoder that transforms the concatenation $[\frac{\phi_\Theta^{(i)}(\hat{N}_i^{(k)})+\phi_\Theta^{(i)}(\hat{N}_j^{(k)})}{2}\|\|\phi_\Theta^{(ii)}(\mathbf{z}^{(k)})]$ into scalar-valued entries of the reconstructed adjacency matrix $\hat{A}_{i,j}$	[Linear(d_n+d_z,256), Leaky_ReLU, Linear(256,512), Leaky_ReLU, Linear(512,256), Leaky_ReLU, Linear(256,1), Sigmoid]
Learning rate used by the Adam optimizer	7×10^{-5}	
β_1,β_2	Coefficients used by the Adam optimizer for computing running averages of gradient and its square	β_1=0.9, β_2=0.999
Batch size	Mini-batch size used when performing mini-batch gradient descent	64

λ_{REC}	Weight of reconstructed adjacency matrix relative to the reconstructed node features in the reconstruction loss term $\mathcal{L}_{REC}$	$\frac{553}{6} \sim 92.167$
λ_{RAE}	Weight of the RAE loss term $\mathcal{L}_{RAE}$	1.0
λ_{REG}	Weight of the L2 weight decay weight in the regularization loss term $\mathcal{L}_{REG}$ applied to the parameters of the decoder W_{Decoder}	0.1
Epochs	Number of training epochs performed	300

1.4.6 Seeding the initial round of active learning

Having trained our RAE to recover d-dimensional latent space representations $\{\mathbf{z}^{(k)} \in \mathbb{R}^d\}_{k=1}^N$ for all N = 694,982 molecules in our X_n-4T-X_n design space, we proceed to select a small subset of molecules that we will subject to all-atom MD simulations that will comprise the data set used to initialize the computational and experimental active learning cycle. A good initial training data set should include a broad and representative sampling of points throughout the latent space embedding to ensure the initial surrogate model can properly capture the global structure of the sequence-property relationship over the design space.

Computational. Selecting this initial data set with simple random sampling fails to account for the embedded data distribution and can result in large regions of the latent space being under sampled leading to skewed latent space coverage and poor initial surrogate model performance. Instead, to properly account for the distribution of points within our latent space embedding we perform k-means clustering on our set of all N latent space representations $\{\mathbf{z}^{(k)} \in \mathbb{R}^d\}_{k=1}^N$. Employing k-mean++ initialization,[45] we identify 100 centroids $\{\mathbf{c}^{(i)}\}_{i=1}^{100}$ that we convert to 100 candidate molecules by identifying the latent vectors

$\mathbf{z}^{(k)}$ most proximate each centroid $\{\underset{\mathbf{z}^{(k)}}{\text{argmin}}||\mathbf{z}^{(k)} - \mathbf{c}^{(i)}||_2\}_{i=1}^{100}$. We augment this set of 100 molecules with 128 hand-selected molecules within our initial dataset selected to ensure our initial training data set also contains a diversity of molecules with different properties such as presence of heteroatoms, varying peptide wing sizes, amino acid polarity, aromaticity, and hydrophobicity. We term this initial round of simulation data collection computational Round 0.

Experimental. We seed Round 0 of the experimental active learning search by selecting 11 of the 228 molecules from computational Round 0 for synthesis and testing. The higher time and labor costs of the experimental screen means that it is a lower throughput process and so we hand-select $\sim$5% of the computational Round 0 molecules designed to span a range of oligopeptide wing lengths and predicted κ and R_g values.

1.5 Step 3: Supervised training of Gaussian process regression (GPR) surrogate models

Using data collected after each round, in the form of either simulated trajectories or experimentally recorded spectral shifts, we aim to predict the computational or experimental fitness of untested candidates. This may be framed as a supervised learning problem where the training data consists of latent embedding vectors and corresponding fitness measurements for the molecules that have been tested to date, with the goal of predicting the fitness of the remaining untested candidates based on the location of their learned latent space molecular representations. Alongside predicting the anticipated fitness of each molecule, it is also vital to quantify our uncertainty in these predictions so that we may balance selecting candidates with high predicted fitness (exploitation) and directing sampling to more unexplored regions of molecular space with high uncertainty (exploration)[46] when performing Bayesian optimization (BO) directed active learning. It is therefore a natural choice to select Gaussian process regression (GPR) as our surrogate model of choice as it constitutes a non-parametric Bayesian regression model equipped with built-in uncertainty quantification.[46,47]

Building this surrogate model to predict the performance of untested candidates effectively circumvents the need to exhaustively simulate or synthesize all the molecules in our design space in favor of data-driven fitness predictions leveraging learned molecular similarities reflected in our chemical space embedding.

1.5.1 Computational GPRs

We independently build two GPRs to separately predict the fitness of our two computational objectives κ and R_g. The two GPRs are identical in their construction with the only difference being the target variable of either κ or R_g, and we henceforth refer to these two models as GPR_κ and GPR_{R_g}. These GPRs together will interface with our Bayesian optimization framework to direct sampling of candidates along the κ-R_g Pareto frontier.

The collection of n π-conjugated peptides simulated after each round comprise our training dataset $\mathbf{X} = \{(\mathbf{z}^{(k)}, y^{(k)}, \sigma^{(k)})\}_{k=1}^{n}$, where $\{\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \ldots, \mathbf{z}^{(n)}\}$ are the latent space embeddings of the simulated molecules, $\{y^{(1)}, y^{(2)}, \ldots, y^{(n)}\}$ are the corresponding response variables $y^{(k)} = \kappa^{(k)}$ or $R_g^{(k)}$, and $\{\sigma^{(1)}, \sigma^{(2)}, \ldots, \sigma^{(n)}\}$ the associated uncertainty in the response variables determined from block averaging. Each GPR is entirely specified by the choice of covariance function, which we select in this work to be the popular radial basis function (RBF) kernel,

$$k(\mathbf{z}, \mathbf{z}') = \nu^2 \exp\left(-\frac{||\mathbf{z} - \mathbf{z}'||^2}{2l^2}\right). \tag{14}$$

$\mathbf{z}$ and $\mathbf{z}'$ correspond to two vectors within our latent space embedding and the marginal variance ν^2 and characteristic length scale l are two kernel hyperparameters. The emukit wrapper[48] to the GPy[18] GPR implementations Python libraries are used to fit ν^2 and l via maximum likelihood estimation of the log marginal likelihood of the training data.[47] We note that we could have also selected a directionally anisotropic automatic relevance determination (ARD)[47] structure for the kernel function k where separate length scales parameters are assigned to each input dimension, but we elected to take the simpler and more computationally efficient approach by selecting the isotropic RBF kernel wherein each

dimension of the latent space is treated on an equal footing.

The trained GPR fitted over all computational data collected to date is then deployed to predict the fitness of the remaining molecules within our design space that have not been simulated and are not contained within the training dataset $\mathbf{X}$. The predicted fitness $y(\mathbf{z}_*) \sim \mathcal{N}(\mu(\mathbf{z}_*), \sigma(\mathbf{z}_*))$ of a vector within our latent space embedding $\mathbf{z}_*$ is a Gaussian distributed random variable with mean $\mu(\mathbf{z}_*)$ and variance $\sigma^2(\mathbf{z}_*)$,

$$\mu(\mathbf{z}_*) = K(\mathbf{z}_*, \cdot)[K + \boldsymbol{\sigma}^T\mathbf{I}]^{-1}\mathbf{y}, \tag{15}$$

$$\sigma^2(\mathbf{z}_*) = k(\mathbf{z}_*, \mathbf{z}_*) - K(\mathbf{z}_*, \cdot)[K + \boldsymbol{\sigma}^T\mathbf{I}]^{-1}K(\mathbf{z}_*, \cdot)^T, \tag{16}$$

where $K \in \mathbb{R}^{n\times n}$ and $K_{i,j} = k(\mathbf{z}^{(i)}, \mathbf{z}^{(j)})$, $K(\mathbf{z}_*, \cdot) = [k(\mathbf{z}_*, \mathbf{z}^{(1)}), k(\mathbf{z}_*, \mathbf{z}^{(2)}), \ldots, k(\mathbf{z}_*, \mathbf{z}^{(n)})]$, $\mathbf{y} = [y^{(1)}, y^{(2)}, \ldots, y^{(n)}]^T$, and $\mathbf{I}$ is a n-by-n identity matrix. The $\boldsymbol{\sigma}^T$ vector in Eqn. 15 and Eqn. 16 added to the diagonal of the kernel matrix K is a heteroskedastic noise built into the GPR to incorporate the uncertainties $\sigma^{(k)}$ inherent to each response $y^{(k)}$. The noise addition also acts as Tikhonov (a.k.a. nugget) regularization that also helps stabilize the matrix inverse and is particularly useful when two or more points in the domain of the training dataset lie in close proximity and cause K to be ill-conditioned.[49,50] The predicted mean and uncertainty for any particular latent space embedding vector $\mathbf{z}^{(k)}$ predicted by GPR_κ and GPR_{R_g} are denoted $(\mu_\kappa(\mathbf{z}^{(k)}), \sigma_\kappa(\mathbf{z}^{(k)}))$ and $(\mu_{R_g}(\mathbf{z}^{(k)}), \sigma_{R_g}(\mathbf{z}^{(k)}))$, respectively.

1.5.2 Multi-fidelity experimental GPR

Our goal ultimately is to identify experimentally high-performing X_n-4T-X_n molecules possessing large values of the spectral blue shift λ. Purely experimentally data-driven active learning screening is frustrated by the high time and labor costs of oligopeptide synthesis and characterization. We can therefore leverage useful correlations and trends present in our comparatively inexpensive and voluminous supply of simulation data to fit a more accurate and generalizable surrogate model by building a multi-fidelity GPR (mfGPR) as described

by Perdikaris et al.[51] The operational principle is to train an autoregressive predictor to learn a mapping from the low-fidelity computational predictions of κ and R_g to the high-fidelity experimental measurements of λ, thereby fusing the two data streams and furnishing a higher accuracy surrogate model for λ than would be possible by training over the experimental data alone.[51] The incorporation of computational data in this manner effectively broadens the purview of our experimental surrogate model to regions of the design space that are otherwise outside the support of the limited experimental data. Importantly, the computational and experimental fitness functions need not measure the same observable and the computational and experimental screening loops can be operated asynchronously and in parallel. The primary requirement for success is that the low-fidelity computational predictions are correlated with and predictive of the high-fidelity experimental measurements. This is expected to be the case since the two computational fitness measures κ and R_g are structural measures of the degree of in-register π-stacking within elongated pseudo-1D nanoaggregates that are characteristics of elevated H-aggregate character that is manifest in larger values of λ. We denote the multi-fidelity surrogate model built to predict spectral shift measurements as GPR_λ.

The multi-fidelity GPR model is constructed by incorporating the posterior mean predictions of a low-fidelity GPR model f_l into the covariance function of the high-fidelity GPR $k_h(\mathbf{z}, \mathbf{z}')$ enabling us to capture nonlinear and space-dependent cross-correlations between low- and high-fidelity data. Given we have fit two single-fidelity GPR models GPR_κ and GPR_{R_g} over the computational data, our low-fidelity model is defined as the linear combination,

$$f_l(\mathbf{z}^{(k)}) = \alpha\ \mu_\kappa(\mathbf{z}^{(k)}) + (1-\alpha)\ \mu_{R_g}(\mathbf{z}^{(k)}), \tag{17}$$

where α is a hyperparameter between 0 and 1. The covariance kernel for the high-fidelity GPR then takes the form,

$$k_h(\mathbf{z}, \mathbf{z}') = \theta_0 \exp\left(-\frac{||\mathbf{z}-\mathbf{z}'||^2}{2\theta_1^2}\right) \exp\left(-\frac{||f_l(\mathbf{z})-f_l(\mathbf{z}')||^2}{2\theta_2^2}\right) + \theta_3 \exp\left(-\frac{||\mathbf{z}-\mathbf{z}'||^2}{2\theta_4^2}\right), \tag{18}$$

where $\{\theta_0, \theta_1, \theta_2, \theta_3, \theta_4\}$ are kernel hyperparameters. The hyperparameters $\{\theta_1, \theta_2, \theta_4\}$ control the bandwidth of the three constituent RBF kernels, whereas $\{\theta_0, \theta_3\}$ control the linear mixing between the first and second terms. Conceptually, $k_h(\mathbf{z}, \mathbf{z}')$ can be understood as a weighted linear sum of two terms. The second term can be viewed as an RBF of a standard single-fidelity GPR operating over the latent space vectors $\mathbf{z}$ that, in isolation, would yield a surrogate model trained over only the high-fidelity, experimental data. The first term is the product of two RBF kernels: one operating directly over the latent space vectors $\mathbf{z}$ and the other over the low-fidelity, computational predictions $f_l(\mathbf{z})$. The second term in this product can be conceived as modulating the first to effectively "squeeze together" two latent space vectors $\mathbf{z}$ and $\mathbf{z}'$ if the computational model predicts that their low-fidelity responses $f_l(\mathbf{z})$ and $f_l(\mathbf{z}')$ are close and, conversely, "push apart" two latent space vectors with divergent low-fidelity predicted responses. As a result, the first term of $k_h(\mathbf{z}, \mathbf{z}')$ acts to provide information transfer from the computational surrogate model to the experimental one by modulating the perceived proximity of $\mathbf{z}$ and $\mathbf{z}'$ as measured by the kernel function. Finally, we observe that the structure of the first term in $k_h(\mathbf{z}, \mathbf{z}')$ as a product operates akin to an AND gate – both $\exp(-\frac{||\mathbf{z}-\mathbf{z}'||^2}{2\theta_1^2})$ and $\exp(-\frac{||f_l(\mathbf{z})-f_l(\mathbf{z}')||^2}{2\theta_2^2})$ must be large for this term to be large – whereas the structure of $k_h(\mathbf{z}, \mathbf{z}')$ itself as a sum operates akin to an OR gate – either $\exp(-\frac{||\mathbf{z}-\mathbf{z}'||^2}{2\theta_1^2})\exp(-\frac{||f_l(\mathbf{z})-f_l(\mathbf{z}')||^2}{2\theta_2^2})$ or $\exp(-\frac{||\mathbf{z}-\mathbf{z}'||^2}{2\theta_4^2})$ can be large for the overall kernel output to be large.[52]

Equipped with this modified kernel function k_h and n observations of high-fidelity training data $\mathbf{X}_h = \{(\mathbf{z}^{(k)}, \lambda^{(k)}, \sigma^{(k)})\}_{k=1}^n$ consisting of latent space representations of the experimentally tested π-conjugated peptides $\{\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \ldots, \mathbf{z}^{(n)}\}$ and spectral shift measurements $\{\lambda^{(1)}, \lambda^{(2)}, \ldots, \lambda^{(n)}\}$ with their corresponding uncertainties $\{\sigma^{(1)}, \sigma^{(2)}, \ldots, \sigma^{(n)}\}$, we can proceed to fit this multi-fidelity GPR_λ using the emukit wrapper[48] to the GPy[18] GPR implementation Python libraries to fit the kernel hyperparameters via maximum likelihood estimation of the log marginal likelihood of the training data.[47] Spectral shift predictions for latent space vectors $\mathbf{z}^{(k)}$ are produced from the fitted mfGPR using Eqs. 15 and 16 to predict

the mean spectral shift $\mu_\lambda(\mathbf{z}^{(k)})$ and corresponding uncertainty $\sigma_\lambda(\mathbf{z}^{(k)})$ for the remaining untested molecules in our design space. Since the posterior distribution of the multi-fidelity GPR_λ is the result of a composition of two Gaussian process priors, it is no longer guaranteed to itself be Gaussian.[51] Analytical expressions for the posterior of this deep Gaussian process are typically unavailable, and the mean and variance of the posterior distribution accounting for the propagation of uncertainty through the multi-layer Gaussian process must instead be numerically estimated by Monte-Carlo estimation.[51] This procedure is very computationally expensive and we instead make the simplifying approximation that the posterior may be approximately represented as a Gaussian in order to enable the use of the analytical expressions Eqs. 15 and 16 that require this assumption. The hyperparameter α in Eq. 17 is determined by five-fold cross validation of the mean absolute error in predicting the spectral shift measurements. Determining the value of α that yields the lowest generalization error of the experimental surrogate model provides an empirical fit for the optimal balance between κ and R_g in simulation that correlates best with the experimentally measured spectral shifts λ. We hesitate to accord too much significance to the optimal value of α, but present in Fig. S5 the results of our α grid search for the terminal experimental active learning round. In the vicinity of $\alpha \approx 0.67$, we observe substantial improvements in the mean average error (MAE) predictive performance of the multi-fidelity GPR_λ used to predict the spectral shift λ relative to a single-fidelity GPR model trained over only experimental measurements. This $\sim$27% improvement in the MAE of GPR_λ clearly illustrates the benefit of the multi-fidelity approach.

The benefit of our multi-fidelity GPR approach can also be clearly seen by examining the distribution of candidates selected throughout our experimental screening and comparing the mfGPR posterior distribution to a single-fidelity GPR analog. The latent space locations of molecules which were simulated throughout all computational rounds is shown in Fig. S6a colored by to the number of contacts per molecule κ and in Fig. S6b colored by to the radius of gyration R_g, while the latent space locations of molecules which that underwent

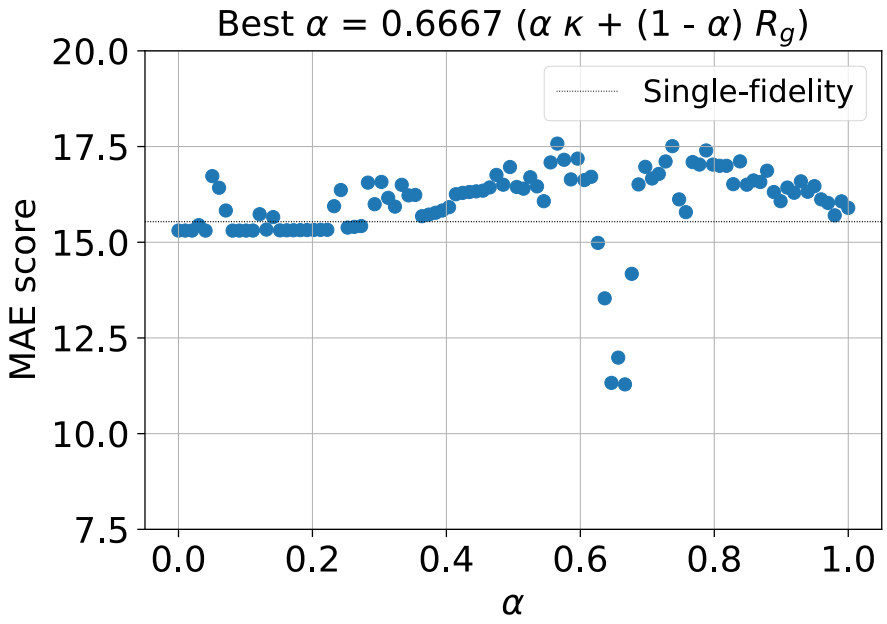

Figure S5: Cross validation determination of α in Eq. 17 determining the balance between κ and R_g in the low-fidelity model used to fit the multi-fidelity GPR. The Leave One Out (LOO) cross validation score of the multi-fidelity GPR is calculated for 100 values of $\alpha \in [0, 1.0]$. The value of α yielding the lowest LOO mean absolute error (MAE) is then selected to build the multi-fidelity GPR_λ surrogate model to predict spectral shift measurements and interfaced with the Bayesian optimizer to select candidates for the next round of experimental testing. The plot above shows the cross validation after the terminal experimental active learning round. The dotted horizontal line represents the LOO MAE of a single-fidelity GPR fit to predict spectral shift measurements in the absence of the low-fidelity computational data. The $\sim$27% improvement in the MAE for the mfGPR in the vicinity of the optimal α=0.6667 illustrates the value of the computational data stream in substantially improving predictive accuracy of GPR_λ.

experimental testing are isolated and is shown in Fig. S6c colored by the measured spectral shift λ. Latent space locations for the experimentally tested molecules are interspersed in regions of dense computational sampling indicative of the transfer of information from the single-fidelity GPRs helping identify promising regions of latent space to explore. This information fusion can also be clearly illustrated when comparing the mfGPR posterior (Fig. S7a and Fig. S7c) to a single-fidelity GPR trained only on all accumulated experimental data to predict spectral shift measurements (Fig. S7b and Fig. S7d). We notice the mfGPR produces favorable (high λ) predictions broadly throughout the latent space and outside the domain of dense experimental sampling, while the single-fidelity GPR has largely limited

favorable predictions to latent space locations nearby experimental training data points. This additional predictive breadth exemplifies the information fusion capabilities of multi-fidelity modeling to successfully integrate voluminous simulation data with sparse of experimental measurements.

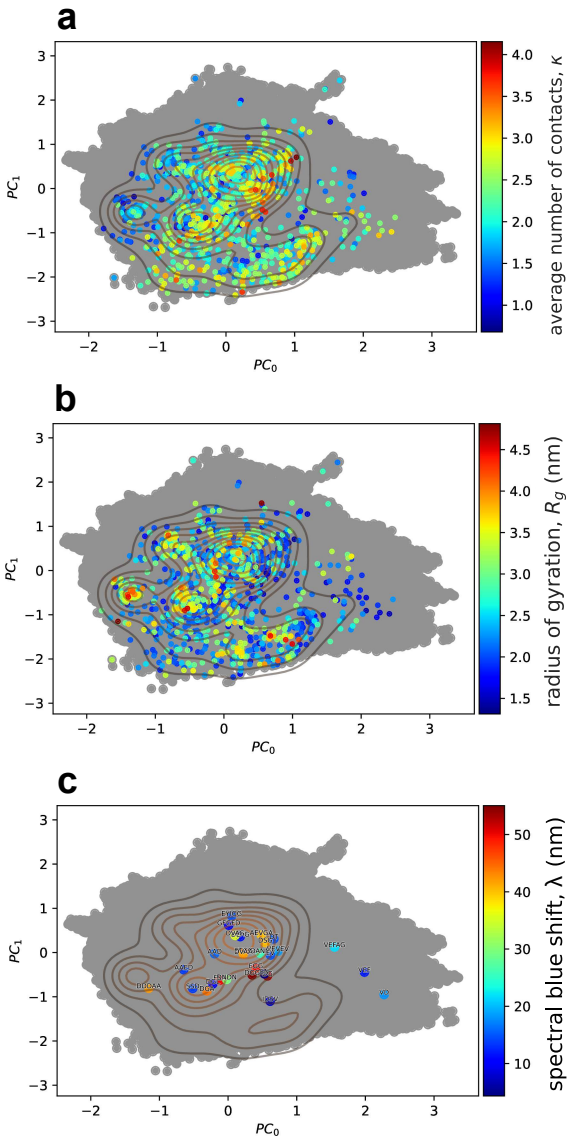

Figure S6: Regularized autoencoder (RAE) latent space locations of all simulated and experimentally tested molecules throughout the active learning procedure. Points in gray represent all 694,982 molecules in our X_n-4T-X_n design space projected into the top two principal components of the full-dimensional latent space embedding. All molecules which were simulated are colored according to their associated average number of contacts per molecule κ (a) and radius of gyration R_g (b). Faded contours represent a kernel density estimate of the distribution of simulated points within the first two principal components generated with Seaborn.[53] (c) Molecules also subject to experimental testing are annotated and colored according to their measured spectral shift. Latent space locations of many experimentally tested candidates are selected nearby areas of dense computational sampling, reflective of the multi-fidelity GPR incorporating simulation data to help identify promising regions of latent space and direct sampling.

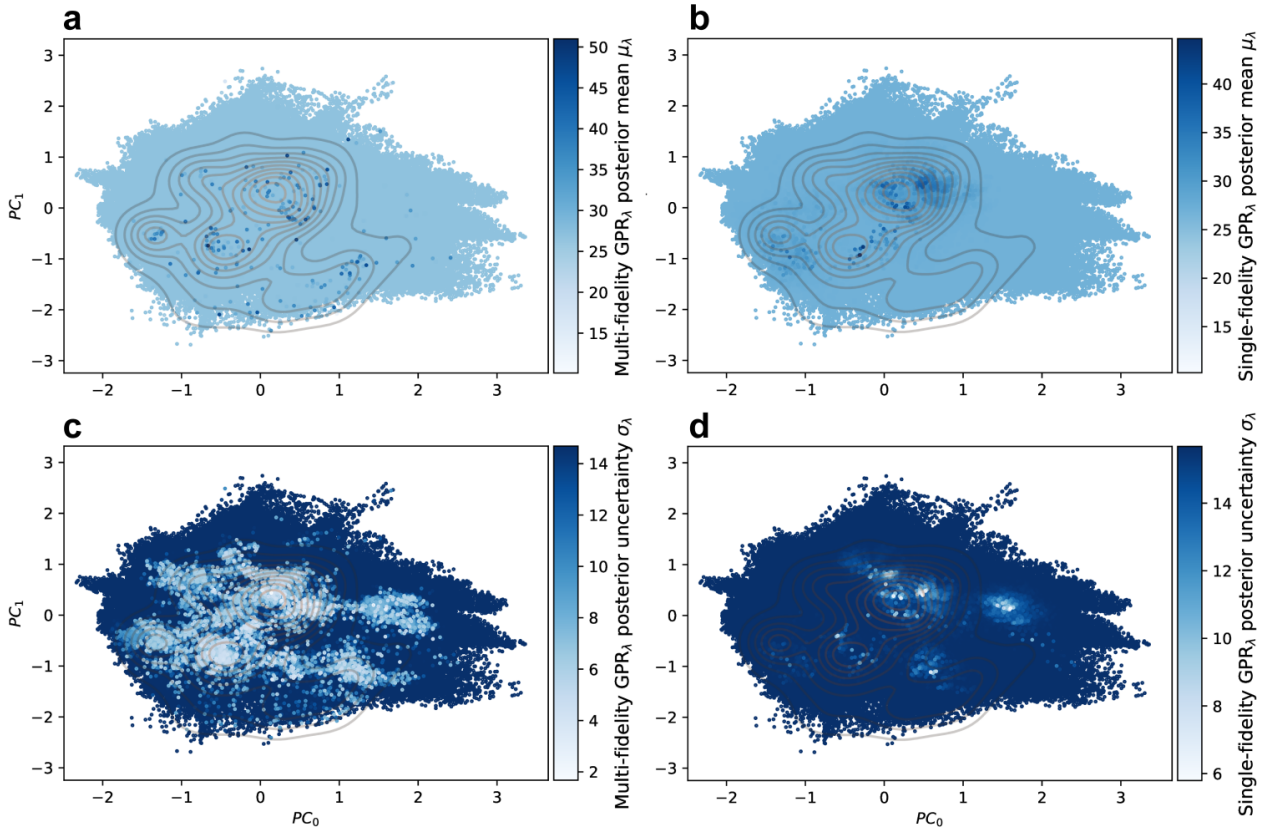

Figure S7: Posterior mean μ_λ and uncertainty σ_λ of multi-fidelity and single-fidelity spectral shift λ predicting Gaussian Process Regression (GPR$_\lambda$) surrogate models. (a, c) A multi-fidelity GPR$_\lambda$ surrogate model is fit to predict the mean spectral shift μ_λ and uncertainty σ_λ trained over the 28 experimentally tested molecules and 1181 simulated molecules. (b, d) A single-fidelity GPR$_\lambda$ surrogate model fit to predict μ_λ and σ_λ of untested molecules trained only over the 28 experimentally tested molecules. The faded gray contours in panels a and b represent a kernel density estimate for the distribution of simulated data within the first two principal components constructed and displayed in Fig. S6, and are meant to guide the eye when comparing the different panels. The single-fidelity GPR$_\lambda$ only produces favorable predictions near regions of dense experimental sampling (cf. Fig. S6c), with the remaining points in the extrapolative regime of the GPR$_\lambda$ producing mean-centered predictions.[47] The multi-fidelity GPR$_\lambda$ produces more favorable and confident predictions distributed broadly throughout the latent space due to the information incorporated from simulation data via the low-fidelity model.[51]

1.6 Step 4: Bayesian optimization (BO) selection of next molecules for computational/experimental screening

The last step in our active learning cycle is to interrogate the surrogate model predictions to select the most promising next X_n-4T-X_n molecules for the next round of computational simulation and experimental testing. Bayesian optimization is a natural choice of optimization algorithm here because we have expensive, nondifferentiable, and noisy measurements that we strive to optimize in the minimum number of necessary iterations.[46,54,55] Multi-objective Bayesian optimization using random scalarizations[56] is performed within the computational active learning pipeline where we seek to recover the κ-R_g Pareto front. The experimental active learning pipeline fuses experimental and computational data within a multi-fidelity GPR that that is then used to perform single-objective Bayesian optimization to maximize experimentally measured spectral blue-shifts λ. Together, the hybrid computational/experimental active learning protocol guides an efficient search for X_n-4T-X_n molecules with large spectral blue shifts λ indicative of strong H-type π-stacking, the emergence of supramolecular electronic delocalization, and the potential for superior electronic and optical functionality.

1.6.1 Computational multi-objective Bayesian optimization to recover the $\kappa - R_g$ Pareto frontier

Within the computational active learning loop, we use a method for multi-objective Bayesian optimization based on random scalarizations of acquisition functions.[56] Single-objective Bayesian optimization operates by passing the surrogate model predictions through an acquisition function u, the maximizer of which identifies the most promising candidate to query $\mathbf{z}^\dagger$ based on the past n observations $\{(\mathbf{z}^{(k)}, \lambda^{(k)}, \sigma^{(k)})\}_{k=1}^n$,

$$\mathbf{z}^\dagger = \underset{\mathbf{z}^{(l)}}{\operatorname{argmax}}\ u(\mathbf{z}^{(l)} \mid \{(\mathbf{z}^{(k)}, \lambda^{(k)}, \sigma^{(k)})\}_{k=1}^n). \tag{19}$$

While some methods for multi-objective Bayesian optimization such as ParEGO[57] consider a linear combination of multiple objectives to compute a single scalar objective, the method of random scalarizations by Paria et al.[56] maintains separate surrogate models for each objective and then considers a linear combination of the corresponding acquisition functions. This formulation for multi-objective optimization achieves sublinear regret bounds contingent on the choice of acquisition function being either Thompson sampling[58] or Upper Confidence Bound[59] (UCB).[56] In this work, we choose to implement the UCB acquisition function,

$$UCB(\mathbf{z}^{(k)};\beta) = \mu(\mathbf{z}^{(k)}) + \beta\sigma(\mathbf{z}^{(k)}), \tag{20}$$

where β is a hyperparameter controlling the explore-exploit trade-off and $\mu(\mathbf{z}^{(k)})$, $\sigma(\mathbf{z}^{(k)})$ are the GPR predicted mean and uncertainty for the latent space vector $\mathbf{z}^{(k)}$ within the smooth, low-dimensional embedding of the molecular design space within the learned RAE latent space. In our particular application, the two independent GPR surrogate models GPR_κ and GPR_{R_g} are used to build two independent acquisition functions UCB_κ and UCB_{R_g} from which we construct the scalarized acquisition function $u(\mathbf{z}^{(k)};\beta,\delta)$ under the method of random scalarizations,

$$\begin{aligned} u(\mathbf{z}^{(k)};\beta,\delta) &= \delta\ UCB_\kappa + (1-\delta)\ UCB_{R_g} \\ &= \delta\ (\mu_\kappa(\mathbf{z}^{(k)}) + \beta\sigma_\kappa(\mathbf{z}^{(k)})) + (1-\delta)\ (\mu_{R_g}(\mathbf{z}^{(k)}) + \beta\sigma_{R_g}(\mathbf{z}^{(k)})), \end{aligned} \tag{21}$$

where δ is a hyperparameter between 0 and 1 controlling the balance between UCB_κ and UCB_{R_g}.

The next best candidate to query by computational simulation $\mathbf{z}^\dagger = \underset{\mathbf{z}^{(k)}}{\text{argmax}}\ u(\mathbf{z}^{(k)};\beta,\delta)$ is the X_n-4T-X_n molecule that has not been simulated to date that maximizes the scalaraized acquisition function $u(\mathbf{z}^{(k)};\beta,\delta)$ for a particular instantiation of β and δ. Rather than pre-specifying values of the hyperparameters β and δ, which would correspond to a particular choice of a trade-off between exploit and explore and one particular scalarization, we rather

integrate over these two hyperparameters by performing repeated queries for the optimal candidate molecule under different choices for β and δ under a batched selection procedure that makes optimal utilization of our parallel compute resources by testing multiple molecules within each active learning iteration. Specifically, in each iteration of the active learning process we instantiate randomly selected values of β over the $\log_{10}$-uniform range $\beta \in [10^{-4}, 10^{4}]$ and δ over the uniform range $\delta \in [0, 1]$ and select from the X_n-4T-X_n molecules that which maximizes $u(\mathbf{z}^{(k)}; \beta, \delta)$. We repeat this procedure until 25 unique candidate molecules are selected and pass these for simulation by all-atom MD. The results of these MD calculations are then appended to our library of simulated X_n-4T-X_n molecules and we commence a new round of computational active learning.

1.6.2 Experimental single-objective Bayesian optimization to maximize λ

We employ a single-fidelity Bayesian optimizer to interrogate the spectral shift predictions of the trained surrogate model GPR_λ and identify the most promising molecular candidates for further rounds of experimental screening as those molecules with RAE latent space coordinates $\mathbf{z}^\dagger$ that maximize an acquisition function $u(\mathbf{z})$. While there are many choices of acquisitions functions for single-fidelity Bayesian optimization,[46] here employ the popular expected improvement (EI) acquisition function.[54,60] Appealing again to the simplifying assumption that the posterior distribution of GPR_λ may be approximated as Gaussian, we circumvent the need from computationally burdensome Monte Carlo estimation of its mean and variance[51] and can employ analytical expressions for the EI acquisition function,[54,60]

$$\text{EI}(\mathbf{z}^{(k)}) = \begin{cases} (\mu_\lambda(\mathbf{z}^{(k)}) - \lambda^{\max} - \xi)\Phi(Z) + \sigma_\lambda(\mathbf{z}^{(k)})\phi(Z) & \sigma_\lambda(\mathbf{z}^{(k)}) > 0 \\ 0 & \sigma_\lambda(\mathbf{z}^{(k)}) = 0 \end{cases}, \tag{22}$$

$$Z = \begin{cases} \frac{\mu_\lambda(\mathbf{z}^{(k)}) - \lambda^{\max} - \xi}{\sigma_\lambda(\mathbf{z}^{(k)})} & \sigma_\lambda(\mathbf{z}^{(k)}) > 0 \\ 0 & \sigma_\lambda(\mathbf{z}^{(k)}) = 0 \end{cases}, \tag{23}$$

where $\lambda^{\max} = \max_{\lambda^{(k)}}\{\lambda^{(k)}\}_{k=1}^{n}$ is the maximum observed spectral shift of all n experimentally tested molecules, ξ is a hyperparameter controlling the exploitation-exploration trade-off, ϕ is the standard normal probability density function, and Φ is the standard normal cumulative distribution function. Exploration is promoted by the second term in Eq. 22 helping to select points with high posterior uncertainty $\sigma_\lambda(\mathbf{z}^{(k)})$, while the first term promotes exploitation selecting points with high predicted posterior mean $\mu_\lambda(\mathbf{z}^{(k)})$. In this work we fix the value of $\xi = 0.01$ as recommended by Lizotte.[60]

The typical Bayesian optimization workflow selects the next candidate to query $\mathbf{z}^\dagger$ by identifying the maximizer of the EI acquisition function over all experimentally untested candidates at each iteration $\mathbf{z}^\dagger = \underset{\mathbf{z}^{(k)}}{\operatorname{argmax}}\ EI(\mathbf{z}^{(k)})$. With the capacity to synthesize and characterize many molecules in parallel we can more rapidly populate our training dataset by testing a batch of molecules each iteration to save time and collect experimental data faster than would otherwise be possible with simple sequential sampling. To select a batch of π-conjugated peptides each iteration we perform a slightly modified human-in-the-loop version of the Kringing believer[61] algorithm. Given an the initial training data set of n labeled molecules $\{\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \ldots, \mathbf{z}^{(n)}\}$ with measured spectral shifts $\{\lambda^{(1)}, \lambda^{(2)}, \ldots, \lambda^{(n)}\}$ and uncertainties $\{\sigma^{(1)}, \sigma^{(2)}, \ldots, \sigma^{(n)}\}$, the Kringing believer method selects a batch of q candidates by first selecting the first member of the batch $\mathbf{z}^{(n+1)}$ by identifying the maximizer of the EI acquisition function given the initial n training data points,

$$\mathbf{z}^{(n+1)} = \underset{\mathbf{z}^{(k)}}{\operatorname{argmax}}\ EI(\mathbf{z}^{(k)} \mid \{\mathbf{z}^{(l)}, \lambda^{(l)}, \sigma^{(l)}\}_{l=1}^{n},\ \mathbf{z}^{(k)} \notin \{\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \ldots, \mathbf{z}^{(n)}\}). \tag{24}$$

The corresponding GPR_λ predicted mean $\mu_\lambda(\mathbf{z}^{(n+1)})$ and uncertainty $\sigma_\lambda(\mathbf{z}^{(n+1)})$ are then appended to the training dataset $\{\mathbf{z}^{(l)}, \lambda^{(l)}, \sigma^{(l)}\}_{l=1}^{(n+1)}$ and the GPR_λ is refit to the artificially augmented training data. Using the posterior predictions from this newly refit GPR_λ, the next member of the batch $\mathbf{z}^{(n+2)}$ is then selected from the pool of untested candidates

excluding the molecules that have been artificially appended to the training dataset,

$$\mathbf{z}^{(n+2)} = \underset{\mathbf{z}^{(k)}}{\operatorname{argmax}}\ EI(\mathbf{z}^{(k)} \mid \{\mathbf{z}^{(l)}, \lambda^{(l)}, \sigma^{(l)}\}_{l=1}^{(n+1)},\ \mathbf{z}^{(k)} \notin \{\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \ldots, \mathbf{z}^{(n)}, \mathbf{z}^{(n+1)}\}). \quad (25)$$

This process is iterated until q candidates have been selected $\{\mathbf{z}^{(n+1)}, \mathbf{z}^{(n+2)}, \ldots, \mathbf{z}^{(n+q)}\}$. Rather than simply taking these first q selected candidates and passing them onto the next round of experimentation, we enumerate a larger list of q=75 candidates and select a subset of 8-9 molecules based on human-in-the-loop down-selection. A common failing of the Kriging Believer method is the tendency to identify largely correlated candidates within a small batch due to an over-confidently high prediction at a point leading to an erroneously large EI in the vicinity of that point.[61] Our human-in-the-loop approach circumvents this issue by enumerating a large number of candidates that we may use our expert intuition to select a subset from for the next round of experimentation based on diversity and anticipated performance. The success of this collaborative human-machine paradigm has been previously demonstrated in the data-driven discovery of molecular organic light emitting diodes.[62] The selected molecules are then synthesized and the measured UV-vis spectral shifts λ appended to our pool of labeled experimental data before commencing another round of experimental active learning.

1.7 Nonlinear manifold learning of low-dimensional oligopeptide assembly pathways using diffusion maps

After completing the hybrid computational/experimental screen, we possess a library of MD simulation trajectories for the self-assembly dynamics of 1181 (1178 molecules suggested from the computational screen and three molecules suggested from the experimental screen which we also simulated) different X_n-4T-X_n candidate molecules. We subject this ensemble of trajectory data to nonlinear dimensionality reduction using diffusion maps[63,64] to furnish a low-dimensional embedding of the trajectories through the high-dimensional Cartesian co-

ordinate space.[21,65–67] This low-dimensional projection of the assembly trajectories preserves the important high-variance collective motions of the self-assembly process and can expose the self-assembly pathways and mechanisms followed by the various oligopeptides. We interrogate this embedding to expose differences in the assembly pathways followed by the high-performing and low-performing candidates in our screen, and in doing so gain insight and understanding of the molecular mechanisms underpinning desirable self-assembling behaviors. We have previously employed this approach to discover assembly pathways within the family of DX_3-(1,4-distyrylbenzene)-X_3D oligopeptides.[20] We note that our diffusion maps furnish a low-dimensional embedding of the *configurational coordinate space* traversed by our MD simulations, and that this embedding is completely independent of the low-dimensional embedding of the *molecular design space* furnished by the RAE that was employed within our active learning protocol.

The application of diffusion maps over the configurational space of π-conjugated peptide assemblies requires a distance metric $d_{i,j}$ that calculates the similarity between pairs of simulation snapshots i and j. The metric must satisfy two criteria: (i) it should be sufficiently general such that the snapshots can come from the same molecular simulation trajectory of one particular X_n-4T-X_n molecule, or from trajectories of two different molecules (i.e., with different X_n peptide wings) and (ii) it should be invariant to rotations, translations, and permutations of atoms within our multi-molecule assemblies such that the diffusion map embedding does not learn and project these trivial transformations into the low-dimensional embedding. A popular and elegant metric satisfying these criteria is the smooth overlap of atomic positions (SOAP) kernel of Csanyi and coworkers[68] that we apply to the heavy (i.e., C and S) atoms comprising the 4T π-cores. The SOAP framework has been previously deployed in a number of applications to derive continuous and real-valued fingerprints of atomic environments used for downstream analysis and machine learning tasks.[69–72]

The key idea of SOAP is describing the local environment around each atom by considering the distribution of neighboring atoms while remaining sensitive to their relative locations.

SOAP encapsulates this information on the local atomic geometry about a particular location in space $\mathbf{r}_i$ using a descriptor called the power spectrum $\mathbf{p}^{Z_1,Z_2}_{n,n',l}(\mathbf{r}_i)$,

$$\mathbf{p}^{Z_1,Z_2}_{n,n',l}(\mathbf{r}_i) = \pi\sqrt{\frac{8}{2l+1}}\sum_m c(\mathbf{r}_i)^{Z_1}_{n,l,m} c(\mathbf{r}_i)^{Z_2}_{n',l,m}, \tag{26}$$

$$c(\mathbf{r}_i)^{Z_i}_{n,l,m} = \iiint_{\mathcal{R}^3} g_n(r) Y_{l,m}(\theta,\phi)\rho^{Z_i}(\mathbf{r}_i)dV, \tag{27}$$

where r, θ, ϕ are variables of integration, $\rho^{Z_i}(\mathbf{r}_i)$ is a Gaussian smoothed density of atoms with atomic number Z_i centered at location $\mathbf{r}_i$, $g_n(r)$ are radial basis functions indexed by n and n', and $Y_{l,m}(\theta,\phi)$ are real spherical harmonics indexed by l and m. For our radial basis functions $g_n(r)$ we use the spherical Gaussian type orbitals and consider basis functions up to n_{max}=12, the maximum angular degree of spherical harmonics up to l_{max}=9, and a cutoff radius of r_{max}=5 nm. This formalism yields a fixed-length, real-valued power spectrum $\mathbf{p}^{Z_1,Z_2}_{n,n',l}(\mathbf{r}_i)$ fingerprint calculated about each atom center $\mathbf{r}_i$ within our simulation frame. The SOAP power spectrum is not particularly amenable to simple interpretability, but can be conceived of as a measure of the spatial distribution of atoms that respects rotational, translational, and permutational invariance, and has proven to be a powerful featurization for machine learning applications in atomic environments.[68–72]

We obtain N_{at} power spectra for each simulation snapshot centered on the N_{at} atomic sites of all atoms in the system. A global descriptor $(\mathbf{p}^{Z_1,Z_2}_{n,n',l})^{(\alpha)}$ for simulation snapshot α is extracted by averaging the power spectra over all N_{at} atomic sites,

$$(\mathbf{p}^{Z_1,Z_2}_{n,n',l})^{(\alpha)} = \frac{1}{N_{at}}\sum_{i=1}^{N_{at}} \mathbf{p}^{Z_1,Z_2}_{n,n',l}(\mathbf{r}_i^{(\alpha)}), \tag{28}$$

where $\mathbf{r}_i^{(\alpha)}$ is the location of atom i within simulation frame α.

The distance metric $d_{\alpha,\beta}$ we use to compare global environments between the atomic

geometries in simulation frames α and β is then compactly represented via a dot product,[69]

$$d_{\alpha,\beta} = \sqrt{2 - 2\ \chi_{\alpha,\beta}} \tag{29}$$

$$\chi_{\alpha,\beta} = (\widehat{\mathbf{p}^{Z_1,Z_2}_{n,n',l}})^{(\alpha)} \cdot (\widehat{\mathbf{p}^{Z_1,Z_2}_{n,n',l}})^{(\beta)} \tag{30}$$

where $\cdot$ denotes the dot product and the $\widehat{\ }$ adornment indicates a unit-length vector such that $\widehat{\mathbf{x}}_i = \frac{\mathbf{x}_i}{\sqrt{\mathbf{x}_i \cdot \mathbf{x}_i}}$.

By applying this calculation to only the heavy C and S atoms constituting the 4T π-cores shared by all members of the X_n-4T-X_n family, $d_{\alpha,\beta}$ is a π-core centric metric that can be applied to any pair of X_n-4T-X_n molecules. Although the metric explicitly neglects the spatial conformations of the X_n peptide wings, the influence of the wings is implicitly preserved within the $d_{\alpha,\beta}$ distance metric through their influence on the spatial arrangement of the π-cores. All SOAP calculations are carried out using the implementation provided in the DScribe python package.[73]

The SOAP-based distance metric $d_{\alpha,\beta}$ defining the (dis)similarity of simulation frames α and β is then passed to the diffusion map as a pairwise distance matrix $\mathbf{d}$ of size 118,100-by-118,100, where 118,100 is the total number of snapshots harvested from the 200 ns production runs over all 1181 simulation trajectories at a period of 2 ns. The first step in the diffusion map analysis is to convolve the $\mathbf{d}$ matrix with a Gaussian kernel to form the matrix $\mathbf{A}$,

$$A_{\alpha,\beta} = \exp\left(\frac{-d^{2\delta}_{\alpha,\beta}}{2\epsilon}\right). \tag{31}$$

The kernel bandwidth ϵ defines the step size of a random walk over the 118,100 snapshots in the high-dimensional coordinate space and can be colloquially interpreted as the characteristic size of the neighborhood around each point or how far each point can "see" via the distance metric $d_{\alpha,\beta}$. An appropriate value of ϵ can be automatically tuned based on the structure of the $\mathbf{A}$ matrix by specifying it to lie in the sigmoidal region of a plot of $\sum_{\alpha,\beta} A_{\alpha,\beta}$ vs. ϵ.[21,63,66] Due to the heterogeneous density distribution of points over the high-

dimensional coordinate space, we also find it useful to apply the density-adaptive variant of diffusion maps proposed by Wang et al.,[74] which introduces a hyperparameter $\delta \in (0, 1]$ within the definition of $A_{\alpha,\beta}$ that helps to smooth out density fluctuations within the data while still preserving the metric properties of $d_{\alpha,\beta}$. Standard diffusion maps correspond to $\delta = 1$, while as $\delta \to 0$ the distribution of distances narrows and effectively helps to smooth out the embedding. In this work we select δ=0.5 and ϵ=0.175.

The next step is to row normalize the kernelized distance matrix $\mathbf{A}$ to form a right stochastic Markov transition matrix $\mathbf{M}$ defining a discrete random walk over the data,

$$\mathbf{M} = \mathbf{D}^{-1}\mathbf{A}, \tag{32}$$

$$D_{\alpha,\beta} = \sum_{\beta=1}^{N} A_{\alpha,\beta}. \tag{33}$$

The probability of hopping from point i to point j in t steps within this random walk over the data is given by the exponentiated matrix element $M^t_{\alpha,\beta}$.[63,64] Provided $\mathbf{M}$ is irreducible and aperiodic, the theory of Markov chains guarantees that it possesses a unique largest eigenvalue of $\lambda_1 = 1$ with associated (trivial) eigenvector $\boldsymbol{\psi}_1 = \mathbf{1}$ corresponding to the stationary distribution of the random walk.[63] The higher order eigenvalues $1 \geq \lambda_2 \geq \lambda_3 \geq \lambda_4 \geq \ldots$ corresponding to eigenvectors $\boldsymbol{\psi}_2, \boldsymbol{\psi}_3, \boldsymbol{\psi}_4, \ldots$ characterize progressively faster relaxing modes of the random walk. By identifying a gap in eigenvalue spectrum we can perform dimensionality reduction and retain the k non-trivial eigenvectors $\boldsymbol{\psi}_2, \boldsymbol{\psi}_3, \ldots, \boldsymbol{\psi}_{k+1}$ corresponding to the leading k eigenvalues $\lambda_2, \lambda_3, \ldots, \lambda_{k+1}$. These eigenvectors effectively describe the slow subspace of the random walk and define an embedding of the data onto a low-dimensional intrinsic manifold to which the dynamical evolution of the random walk is effectively restrained and which contains the leading high-variance collective motions of the system governing its long-time evolution.[63,64] The diffusion map embedding of a simulation frame α into the

leading k non-trivial eigenvectors is defined as,

$$\text{frame}_\alpha \rightarrow [\boldsymbol{\psi}_2(\alpha), \boldsymbol{\psi}_3(\alpha), \ldots, \boldsymbol{\psi}_{k+1}(\alpha)]. \tag{34}$$

While the eigenvalue spectrum displays the most prominent gap between λ_4 and λ_5 suggesting a 3D manifold (Fig. S8a), the embedding between $\boldsymbol{\psi}_2$ and $\boldsymbol{\psi}_4$ (Fig. S8b) reveals a functional dependence and collapse of these two eigenvectors onto a pseudo-1D manifold. This suggests that these two components characterize the same collective dynamical mode and enable us to eliminate $\boldsymbol{\psi}_4$ without significant loss of information to construct a 2D embedding into the two leading non-trivial eigenvectors $\{\boldsymbol{\psi}_2, \boldsymbol{\psi}_3\}$ (Fig. S8c).[75]

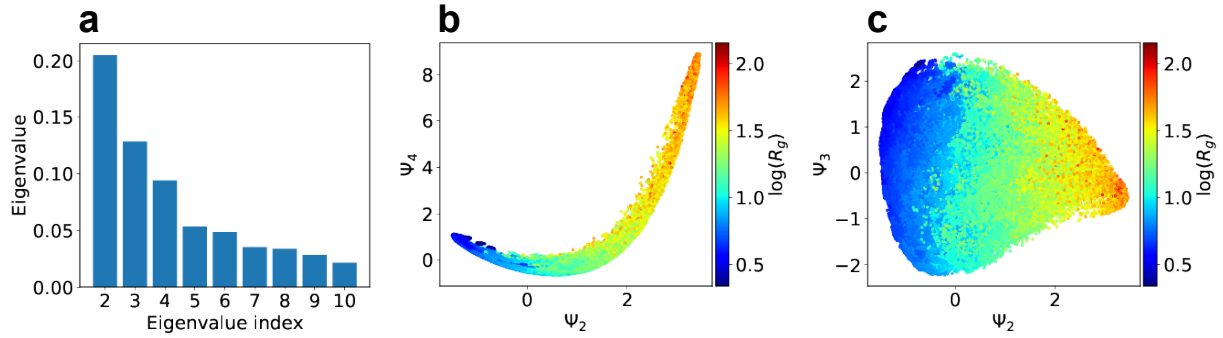

Figure S8: Diffusion map analysis of molecular assembly pathways. (a) Eigenvalue spectrum from the application of diffusion maps to the ensemble of 1181 molecular simulation trajectories conducted over the course of the computational screen. Two-dimensional embeddings of the 118,100 simulation snapshots harvested from the 1181 simulation trajectories into (b) $\boldsymbol{\psi}_2$ and $\boldsymbol{\psi}_4$ and (c) $\boldsymbol{\psi}_2$ and $\boldsymbol{\psi}_3$ where each frame is colored by the log of the radius of gyration $\log(R_g)$. Although the eigenvalue spectrum gap observed between λ_4 and λ_5 suggests a 3D embedding into $\{\boldsymbol{\psi}_2, \boldsymbol{\psi}_3, \boldsymbol{\psi}_4\}$, an observed functional dependence between $\boldsymbol{\psi}_2$ and $\boldsymbol{\psi}_4$ permits us to eliminate $\boldsymbol{\psi}_4$ and construct a 2D intrinsic manifold in $\{\boldsymbol{\psi}_2, \boldsymbol{\psi}_3\}$.

Analysis of the projection of the ensemble of X_n-4T-X_n simulation trajectories into the intrinsic manifold provides a wealth of interpretable information about the dynamical evolution of the self-assembly processes. For example, we can identify which molecules tend to follow similar assembly courses, resolve important collective motions governing the assembly dynamics, and extract structural mechanisms of assembly and correlate these with the rank ordering of the molecular candidates furnished by our sequence-property surrogate model.

1.8 Peptide Synthesis

1.8.1 Round 0

HO-DG-4T-GD-OH: Solid-supported Wang-DG-NH_2 peptide N-acylated with 5-bromothiophene-2-carboxylic acid was prepared (0.2 mmol). The peptide was coupled with 5,5′-bis-trimethylstannyl-[2,2′]-bithiophene (0.010 mmol, 0.049 g) in the presence of $Pd(PPh_3)_4$ (0.0080 mmol, 0.0093 g) using the general on-resin Stille coupling procedure for 20 hours. Resin was then subjected to the general cleavage procedure and crude peptide was obtained as a yellow powder. Following HPLC purification, 0.0146 mmol, 0.0111 g, 15% yield. MS (ESI) m/z 1524.2 $(2M-H)^-$ (calc. 1523.1), m/z 1142.1 $(3M-2H)^{2-}$ (calc. 1142.1), m/z 760.98 $(M-H)^-$ (calc. 761.04), m/z 380.03 $(M-2H)^{2-}$ (calc. 380.01).

HO-VD-4T-DV-OH: Solid-supported Wang-VD-NH_2 peptide N-acylated with 5-bromothiophene-2-carboxylic acid was prepared (0.2 mmol). The peptide was coupled with 5,5′-bis-trimethylstannyl-[2,2′]-bithiophene (0.010 mmol, 0.049 g) in the presence of $Pd(PPh_3)_4$ (0.0080 mmol, 0.0093 g) using the general on-resin Stille coupling procedure for 20 hours. Resin was then subjected to the general cleavage procedure and crude peptide was obtained as a yellow powder. Following HPLC purification, 0.0130 mmol, 0.0110 g, 13% yield. MS (ESI) m/z 1692.9 $(2M-H)^-$ (calc. 1691.3), m/z 1410.2 $(5M-3H)^{3-}$ (calc. 1409.2), m/z 1268.9 $(3M-2H)^{2-}$ (calc. 1268.2), m/z 845.06 $(M-H)^-$ (calc. 845.14), m/z 421.96 $(M-2H)^{2-}$ (calc. 422.07).

HO-EV-4T-VE-OH: Solid-supported Wang-EV-NH_2 peptide N-acylated with 5-bromothiophene-2-carboxylic acid was prepared (0.2 mmol). The peptide was coupled with 5,5′-bis-trimethylstannyl-[2,2′]-bithiophene (0.010 mmol, 0.049 g) in the presence of $Pd(PPh_3)_4$ (0.0080 mmol, 0.0093 g) using the general on-resin Stille coupling procedure for 20 hours. Resin was then subjected to the general cleavage procedure and crude peptide was obtained as a yellow powder. Following HPLC purification, 0.0106 mmol, 0.0093 g, 11% yield. MS (ESI) m/z 1748.1 $(2M-H)^-$ (calc. 1747.3), m/z 1456.8 $(5M-3H)^{3-}$ (calc. 1455.9), m/z 1311.1 $(3M-2H)^{2-}$ (calc. 1310.2), m/z 873.18 $(M-H)^-$ (calc. 873.16), m/z 436.18 $(M-2H)^{2-}$ (calc. 436.08).

HO-EGG-4T-GGE-OH: Prepared according to the protocol reported before.[29]

HO-VEF-4T-FEV-OH: Solid-supported Wang-VEF-NH_2 peptide N-acylated with 5-bromothiophene-2-carboxylic acid was prepared (0.2 mmol). The peptide was coupled with 5,5′-bis-trimethylstannyl-[2,2′]-bithiophene (0.010 mmol, 0.049 g) in the presence of $Pd(PPh_3)_4$ (0.0080 mmol, 0.0093 g) using the general on-resin Stille coupling procedure for 20 hours. Resin was then subjected to the general cleavage procedure and crude peptide was obtained as a yellow powder. Following HPLC purification, 0.0098 mmol, 0.0114 g, 10% yield. MS (ESI) m/z 1947.4 $(5M\text{-}3H)^{3-}$ (calc. 1946.2), m/z 1752.4 $(3M\text{-}2H)^{2-}$ (calc. 1751.5), m/z 1557.6 $(4M\text{-}3H)^{3-}$ (calc. 1556.7), m/z 1167.6 $(M\text{-}H)^{-}$ (calc. 1167.3), m/z 583.28 $(M\text{-}2H)^{2-}$ (calc. 583.15).

HO-AAD-4T-DAA-OH: Solid-supported Wang-AAD-NH_2 peptide N-acylated with 5-bromothiophene-2-carboxylic acid was prepared (0.2 mmol). The peptide was coupled with 5,5′-bis-trimethylstannyl-[2,2′]-bithiophene (0.010 mmol, 0.049 g) in the presence of $Pd(PPh_3)_4$ (0.0080 mmol, 0.0093 g) using the general on-resin Stille coupling procedure for 20 hours. Resin was then subjected to the general cleavage procedure and crude peptide was obtained as a yellow powder. Following HPLC purification, 0.0208 mmol, 0.0194 g, 21% yield. MS (ESI) m/z 465.03 $(M\text{-}2H)^{2-}$ (calc. 465.08), m/z 311.03 $(M\text{-}2H)^{3-}$ (calc. 309.72).

HO-DVAA-4T-AAVD-OH: Solid-supported Wang-DVAA-NH_2 peptide N-acylated with 5-bromothiophene-2-carboxylic acid was prepared (0.2 mmol). The peptide was coupled with 5,5′-bis-trimethylstannyl-[2,2′]-bithiophene (0.010 mmol, 0.049 g) in the presence of $Pd(PPh_3)_4$ (0.0080 mmol, 0.0093 g) using the general on-resin Stille coupling procedure for 20 hours. Resin was then subjected to the general cleavage procedure and crude peptide was obtained as a yellow powder. Following HPLC purification, 0.0264 mmol, 0.0298 g, 26% yield. MS (ESI) m/z 1129.2 $(M\text{-}H)^{-}$ (calc. 1129.3), m/z 564.28 $(M\text{-}2H)^{2-}$ (calc. 564.15).

HO-DVAG-4T-GAVD-OH: Solid-supported Wang-DVAG-NH_2 peptide N-acylated with 5-bromothiophene-2-carboxylic acid was prepared (0.2 mmol). The peptide was coupled with 5,5′-bis-trimethylstannyl-[2,2′]-bithiophene (0.010 mmol, 0.049 g) in the presence of

$Pd(PPh_3)_4$ (0.0080 mmol, 0.0093 g) using the general on-resin Stille coupling procedure for 20 hours. Resin was then subjected to the general cleavage procedure and crude peptide was obtained as a yellow powder. Following HPLC purification, 0.0258 mmol, 0.0284 g, 26% yield. MS (ESI) m/z 1101.2 $(M\text{-}H)^-$ (calc. 1101.3), m/z 550.19 $(M\text{-}2H)^{2-}$ (calc. 550.63).

HO-AAED-4T-DEAA-OH: Solid-supported Wang-AAED-NH_2 peptide N-acylated with 5-bromothiophene-2-carboxylic acid was prepared (0.2 mmol). The peptide was coupled with 5,5′-bis-trimethylstannyl-[2,2′]-bithiophene (0.010 mmol, 0.049 g) in the presence of $Pd(PPh_3)_4$ (0.0080 mmol, 0.0093 g) using the general on-resin Stille coupling procedure for 20 hours. Resin was then subjected to the general cleavage procedure and crude peptide was obtained as a yellow powder. Following HPLC purification, 0.0196 mmol, 0.0233 g, 20% yield. MS (ESI) m/z 594.08 $(M\text{-}2H)^{2-}$ (calc. 594.12), m/z 395.83 $(M\text{-}3H)^{3-}$ (calc. 395.74).

HO-VEFAG-4T-GAFEV-OH: Solid-supported Wang-VEFAG-NH_2 peptide N-acylated with 5-bromothiophene-2-carboxylic acid was prepared (0.2 mmol). The peptide was coupled with 5,5′-bis-trimethylstannyl-[2,2′]-bithiophene (0.010 mmol, 0.049 g) in the presence of $Pd(PPh_3)_4$ (0.0080 mmol, 0.0093 g) using the general on-resin Stille coupling procedure for 20 hours. Resin was then subjected to the general cleavage procedure and crude peptide was obtained as a yellow powder. Following HPLC purification, 0.0078 mmol, 0.0111 g, 7.8% yield. MS (ESI) m/z 1424.3 $(M\text{-}H)^-$ (calc. 1423.4), m/z 711.29 $(M\text{-}2H)^{2-}$ (calc. 711.21).

HO-VEVEV-4T-VEVEV-OH: Solid-supported Wang-VEVEV-NH_2 peptide N-acylated with 5-bromothiophene-2-carboxylic acid was prepared (0.2 mmol). The peptide was coupled with 5,5′-bis-trimethylstannyl-[2,2′]-bithiophene (0.010 mmol, 0.049 g) in the presence of $Pd(PPh_3)_4$ (0.0080 mmol, 0.0093 g) using the general on-resin Stille coupling procedure for 20 hours. Resin was then subjected to the general cleavage procedure and crude peptide was obtained as a yellow powder. Following HPLC purification, 0.0086 mmol, 0.0131 g, 8.6% yield. MS (ESI) m/z 1527.9 $(M\text{-}H)^-$ (calc. 1527.5), m/z 763.29 $(M\text{-}2H)^{2-}$ (calc. 763.25).

1.8.2 Round 1

HO-DT-4T-TD-OH: Solid-supported Wang-DT-NH_2 peptide N-acylated with 5-bromothiophene-2-carboxylic acid was prepared (0.2 mmol). The peptide was coupled with 5,5′-bis-trimethylstannyl-[2,2′]-bithiophene (0.010 mmol, 0.049 g) in the presence of $Pd(PPh_3)_4$ (0.0080 mmol, 0.0093 g) using the general on-resin Stille coupling procedure for 20 hours. Resin was then subjected to the general cleavage procedure and crude peptide was obtained as a yellow powder. Following HPLC purification, 0.0092 mmol, 0.0078 g, 9.2% yield. MS (ESI) m/z 424.01 $(M\text{-}2H)^{2-}$ (calc. 424.05).

HO-EN-4T-NE-OH: Solid-supported Wang-EN-NH_2 peptide N-acylated with 5-bromothiophene-2-carboxylic acid was prepared (0.2 mmol). The peptide was coupled with 5,5′-bis-trimethylstannyl-[2,2′]-bithiophene (0.010 mmol, 0.049 g) in the presence of $Pd(PPh_3)_4$ (0.0080 mmol, 0.0093 g) using the general on-resin Stille coupling procedure for 20 hours. Resin was then subjected to the general cleavage procedure and crude peptide was obtained as a yellow powder. Following HPLC purification, 0.0072 mmol, 0.0065 g, 7.2% yield. MS (ESI) m/z 451.22 $(M\text{-}2H)^{2-}$ (calc. 451.06).

HO-DSG-4T-GSD-OH: Solid-supported Wang-DSG-NH_2 peptide N-acylated with 5-bromothiophene-2-carboxylic acid was prepared (0.2 mmol). The peptide was coupled with 5,5′-bis-trimethylstannyl-[2,2′]-bithiophene (0.010 mmol, 0.049 g) in the presence of $Pd(PPh_3)_4$ (0.0080 mmol, 0.0093 g) using the general on-resin Stille coupling procedure for 20 hours. Resin was then subjected to the general cleavage procedure and crude peptide was obtained as a yellow powder. Following HPLC purification, 0.0104 mmol, 0.0097 g, 10% yield. MS (ESI) m/z 935.05 $(M\text{-}H)^{-}$ (calc. 935.11), m/z 467.06 $(M\text{-}2H)^{2-}$ (calc. 467.05).

HO-SSD-4T-DSS-OH: Solid-supported Wang-SSD-NH_2 peptide N-acylated with 5-bromothiophene-2-carboxylic acid was prepared (0.2 mmol). The peptide was coupled with 5,5′-bis-trimethylstannyl-[2,2′]-bithiophene (0.010 mmol, 0.049 g) in the presence of $Pd(PPh_3)_4$ (0.0080 mmol, 0.0093 g) using the general on-resin Stille coupling procedure for 20 hours. Resin was then subjected to the general cleavage procedure and crude peptide was obtained

as a yellow powder. Following HPLC purification, 0.0112 mmol, 0.0112 g, 11% yield. MS (ESI) m/z 497.02 $(M\text{-}2H)^{2-}$ (calc. 497.07).

HO-DGL-4T-LGD-OH: Solid-supported Wang-DGL-NH_2 peptide N-acylated with 5-bromothiophene-2-carboxylic acid was prepared (0.2 mmol). The peptide was coupled with 5,5′-bis-trimethylstannyl-[2,2′]-bithiophene (0.010 mmol, 0.049 g) in the presence of $Pd(PPh_3)_4$ (0.0080 mmol, 0.0093 g) using the general on-resin Stille coupling procedure for 20 hours. Resin was then subjected to the general cleavage procedure and crude peptide was obtained as a yellow powder. Following HPLC purification, 0.0142 mmol, 0.0141 g, 14% yield. MS (ESI) m/z 493.11 $(M\text{-}2H)^{2-}$ (calc. 493.12).

HO-DNDN-4T-NDND-OH: Solid-supported Wang-DNDN-NH_2 peptide N-acylated with 5-bromothiophene-2-carboxylic acid was prepared (0.2 mmol). The peptide was coupled with 5,5′-bis-trimethylstannyl-[2,2′]-bithiophene (0.010 mmol, 0.049 g) in the presence of $Pd(PPh_3)_4$ (0.0080 mmol, 0.0093 g) using the general on-resin Stille coupling procedure for 20 hours. Resin was then subjected to the general cleavage procedure and crude peptide was obtained as a yellow powder. Following HPLC purification, 0.0072 mmol, 0.0096 g, 7.2% yield. MS (ESI) m/z 1410.3 (M+2K-3H) - (calc. 1409.2), m/z 1334.06 $(M\text{-}H)^{-}$ (calc. 1333.23), m/z 666.22 $(M\text{-}2H)^{2-}$ (calc. 666.11).

HO-IDSV-4T-VSDI-OH: Solid-supported Wang-IDSV-NH_2 peptide N-acylated with 5-bromothiophene-2-carboxylic acid was prepared (0.2 mmol). The peptide was coupled with 5,5′-bis-trimethylstannyl-[2,2′]-bithiophene (0.010 mmol, 0.049 g) in the presence of $Pd(PPh_3)_4$ (0.0080 mmol, 0.0093 g) using the general on-resin Stille coupling procedure for 20 hours. Resin was then subjected to the general cleavage procedure and crude peptide was obtained as a yellow powder. Following HPLC purification, 0.0094 mmol, 0.0117 g, 9.4% yield. MS (ESI) m/z 1284.3 (M+K-2H) - (calc. 1283.4), m/z 1246.3 $(M\text{-}H)^{-}$ (calc. 1245.4), m/z 622.24 $(M\text{-}2H)^{2-}$ (calc. 622.22).

HO-EYIQG-4T-GQIYE-OH: Solid-supported Wang-EYIQG-NH_2 peptide N-acylated with 5-bromothiophene-2-carboxylic acid was prepared (0.2 mmol). The peptide was cou-

pled with 5,5′-bis-trimethylstannyl-[2,2′]-bithiophene (0.010 mmol, 0.049 g) in the presence of $Pd(PPh_3)_4$ (0.0080 mmol, 0.0093 g) using the general on-resin Stille coupling procedure for 20 hours. Resin was then subjected to the general cleavage procedure and crude peptide was obtained as a yellow powder. Following HPLC purification, 0.0064 mmol, 0.0102 g, 6.4% yield. MS (ESI) m/z 1598.9 $(M\text{-}H)^-$ (calc. 1597.5), m/z 798.52 $(M\text{-}2H)^{2-}$ (calc. 798.25), m/z 532.23 $(M\text{-}3H)^{3-}$ (calc. 531.83).

HO-GFGFD-4T-DFGFG-OH: Solid-supported Wang-GFGFD-NH_2 peptide N-acylated with 5-bromothiophene-2-carboxylic acid was prepared (0.2 mmol). The peptide was coupled with 5,5′-bis-trimethylstannyl-[2,2′]-bithiophene (0.010 mmol, 0.049 g) in the presence of $Pd(PPh_3)_4$ (0.0080 mmol, 0.0093 g) using the general on-resin Stille coupling procedure for 20 hours. Resin was then subjected to the general cleavage procedure and crude peptide was obtained as a yellow powder. Following HPLC purification, 0.0078 mmol, 0.0114 g, 7.8% yield. MS (ESI) m/z 1464.7 $(M\text{-}H)^-$ (calc. 1463.4), m/z 731.54 $(M\text{-}2H)^{2-}$ (calc. 731.18).

1.8.3 Round 2

HO-DGG-4T-GGD-OH: Prepared according to the protocol reported before.[28]

HO-ESA-4T-ASE-OH: Solid-supported Wang-ESA-NH_2 peptide N-acylated with 5-bromothiophene-2-carboxylic acid was prepared (0.2 mmol). The peptide was coupled with 5,5′-bis-trimethylstannyl-[2,2′]-bithiophene (0.010 mmol, 0.049 g) in the presence of $Pd(PPh_3)_4$ (0.0080 mmol, 0.0093 g) using the general on-resin Stille coupling procedure for 20 hours. Resin was then subjected to the general cleavage procedure and crude peptide was obtained as a yellow powder. Following HPLC purification, 0.0092 mmol, 0.0091 g, 9.2% yield. MS (ESI) m/z 991.29 $(M\text{-}H)^-$ (calc. 991.17), m/z 495.20 $(M\text{-}2H)^{2-}$ (calc. 495.09).

HO-DGA-4T-AGD-OH: Solid-supported Wang-DGA-NH_2 peptide N-acylated with 5-bromothiophene-2-carboxylic acid was prepared (0.2 mmol). The peptide was coupled with 5,5′-bis-trimethylstannyl-[2,2′]-bithiophene (0.010 mmol, 0.049 g) in the presence of $Pd(PPh_3)_4$ (0.0080 mmol, 0.0093 g) using the general on-resin Stille coupling procedure for

20 hours. Resin was then subjected to the general cleavage procedure and crude peptide was obtained as a yellow powder. Following HPLC purification, 0.0176 mmol, 0.0159 g, 18% yield. MS (ESI) m/z 903.09 $(M\text{-}H)^-$ (calc. 903.12), m/z 451.10 $(M\text{-}2H)^{2-}$ (calc. 451.06).

HO-ETGG-4T-GGTE-OH: Solid-supported Wang-ETGG-NH_2 peptide N-acylated with 5-bromothiophene-2-carboxylic acid was prepared (0.2 mmol). The peptide was coupled with 5,5′-bis-trimethylstannyl-[2,2′]-bithiophene (0.010 mmol, 0.049 g) in the presence of $Pd(PPh_3)_4$ (0.0080 mmol, 0.0093 g) using the general on-resin Stille coupling procedure for 20 hours. Resin was then subjected to the general cleavage procedure and crude peptide was obtained as a yellow powder. Following HPLC purification, 0.0112 mmol, 0.0124 g, 11% yield. MS (ESI) m/z 552.32 $(M\text{-}2H)^{2-}$ (calc. 552.11).

HO-DANN-4T-NNAD-OH: Solid-supported Wang-DANN-NH_2 peptide N-acylated with 5-bromothiophene-2-carboxylic acid was prepared (0.2 mmol). The peptide was coupled with 5,5′-bis-trimethylstannyl-[2,2′]-bithiophene (0.010 mmol, 0.049 g) in the presence of $Pd(PPh_3)_4$ (0.0080 mmol, 0.0093 g) using the general on-resin Stille coupling procedure for 20 hours. Resin was then subjected to the general cleavage procedure and crude peptide was obtained as a yellow powder. Following HPLC purification, 0.0118 mmol, 0.0147 g, 12% yield. MS (ESI) m/z 622.14 $(M\text{-}2H)^{2-}$ (calc. 622.17).

HO-DLAG-4T-GALD-OH: Solid-supported Wang-DLAG-NH_2 peptide N-acylated with 5-bromothiophene-2-carboxylic acid was prepared (0.2 mmol). The peptide was coupled with 5,5′-bis-trimethylstannyl-[2,2′]-bithiophene (0.010 mmol, 0.049 g) in the presence of $Pd(PPh_3)_4$ (0.0080 mmol, 0.0093 g) using the general on-resin Stille coupling procedure for 20 hours. Resin was then subjected to the general cleavage procedure and crude peptide was obtained as a yellow powder. Following HPLC purification, 0.0274 mmol, 0.0311 g, 27% yield. MS (ESI) m/z 1695.6 $(3M\text{-}2H)^{2-}$ (calc. 1694.4), m/z 1129.5 $(M\text{-}H)^-$ (calc. 1129.3), m/z 564.36 $(M\text{-}2H)^{2-}$ (calc. 564.14).

HO-DDDAA-4T-AADDD-OH: Solid-supported Wang-DDDAA-NH_2 peptide N-acylated with 5-bromothiophene-2-carboxylic acid was prepared (0.2 mmol). The peptide was cou-

pled with 5,5′-bis-trimethylstannyl-[2,2′]-bithiophene (0.010 mmol, 0.049 g) in the presence of $Pd(PPh_3)_4$ (0.0080 mmol, 0.0093 g) using the general on-resin Stille coupling procedure for 20 hours. Resin was then subjected to the general cleavage procedure and crude peptide was obtained as a yellow powder. Following HPLC purification, 0.0124 mmol, 0.0172 g, 12% yield. MS (ESI) m/z 695.34 $(M\text{-}2H)^{2-}$ (calc. 695.13), m/z 463.24 $(M\text{-}3H)^{3-}$ (calc. 463.08).

HO-AEVGA-4T-AGVEA-OH: Solid-supported Wang-AEVGA-NH_2 peptide N-acylated with 5-bromothiophene-2-carboxylic acid was prepared (0.2 mmol). The peptide was coupled with 5,5′-bis-trimethylstannyl-[2,2′]-bithiophene (0.010 mmol, 0.049 g) in the presence of $Pd(PPh_3)_4$ (0.0080 mmol, 0.0093 g) using the general on-resin Stille coupling procedure for 20 hours. Resin was then subjected to the general cleavage procedure and crude peptide was obtained as a yellow powder. Following HPLC purification, 0.0282 mmol, 0.0359 g, 28% yield. MS (ESI) m/z 1271.4 $(M\text{-}H)^{-}$ (calc. 1271.7), m/z 635.65 $(M\text{-}2H)^{2-}$ (calc. 635.18).

1.9 Safety comment

There were no unexpected or unusual safety hazards encountered.

1.10 ESI Spectra

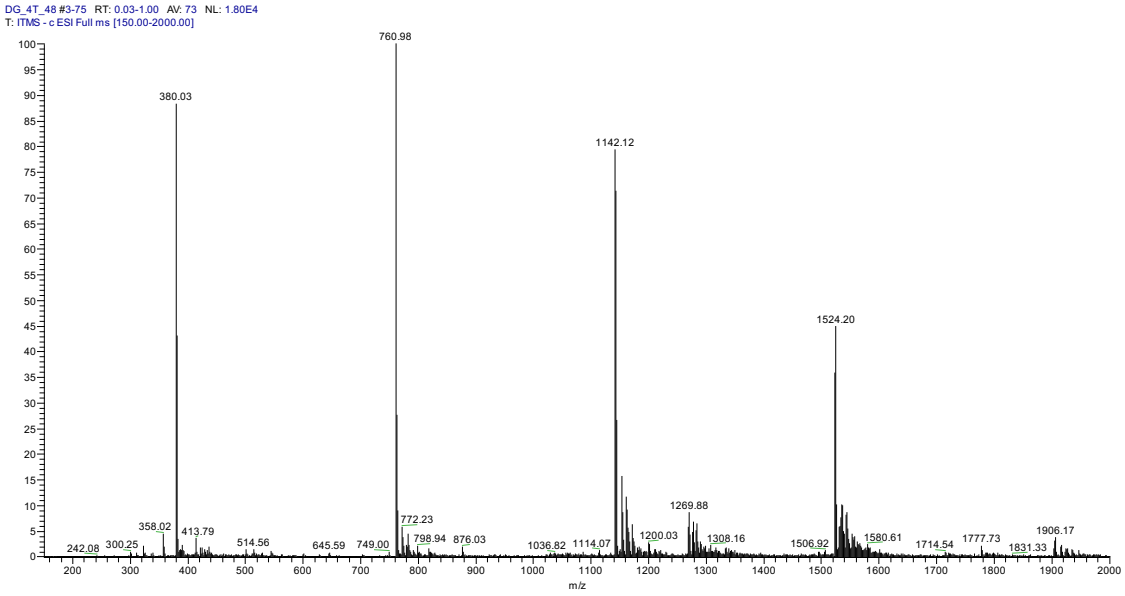

Figure S9: ESI- of HO-DG-4T-GD-OH.

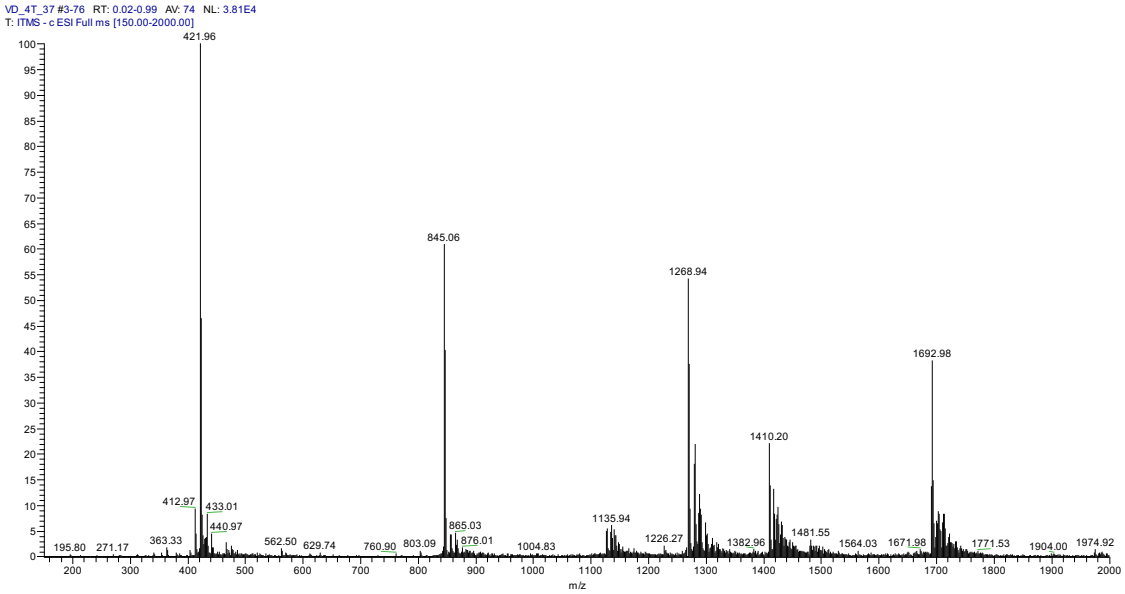

Figure S10: ESI- of HO-VD-4T-DV-OH.

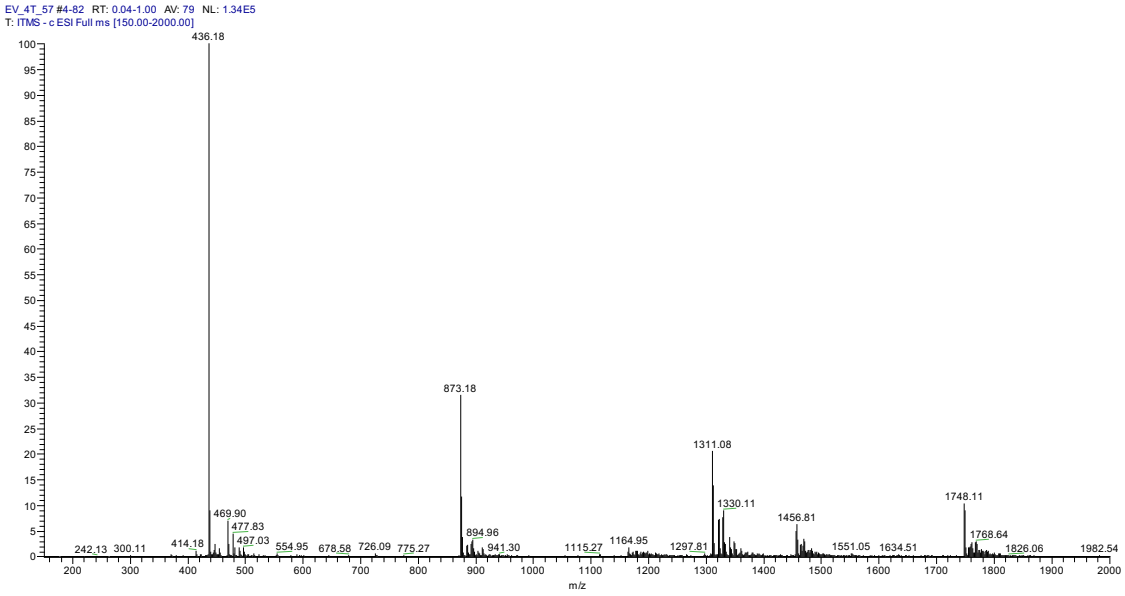

Figure S11: ESI- of HO-EV-4T-VE-OH.

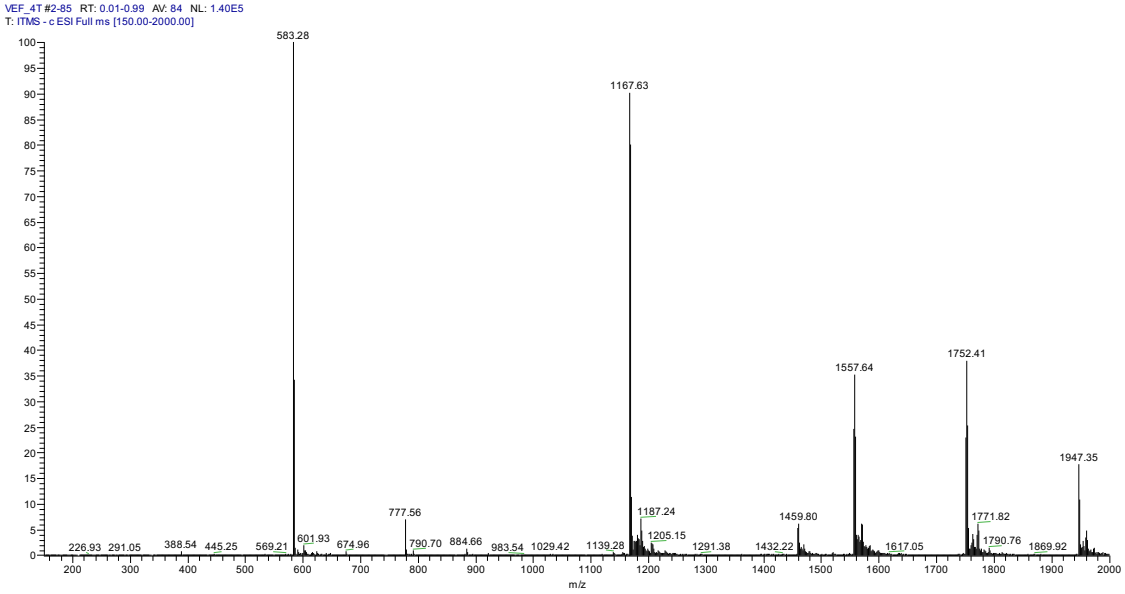

Figure S12: ESI- of HO-VEF-4T-FEV-OH.

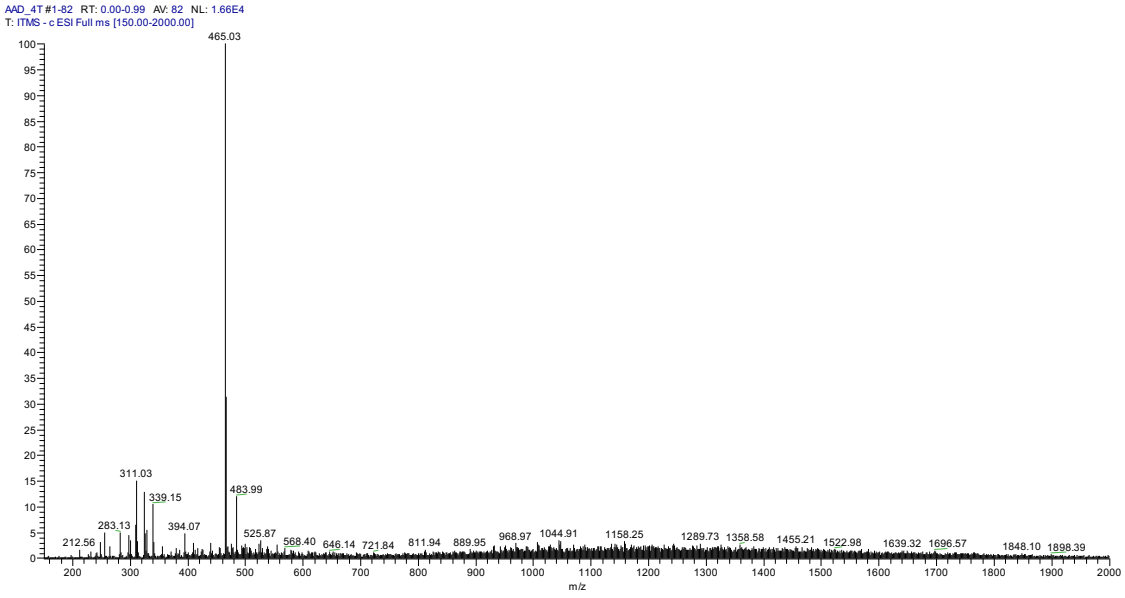

Figure S13: ESI- of HO-AAD-4T-DAA-OH.

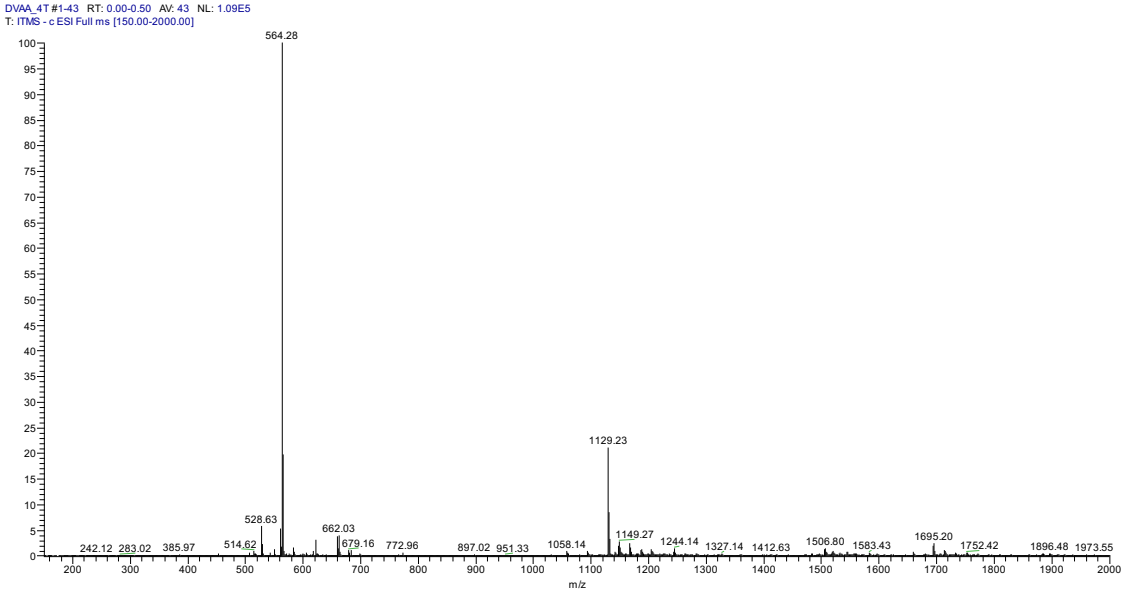

Figure S14: ESI- of HO-DVAA-4T-AAVD-OH.

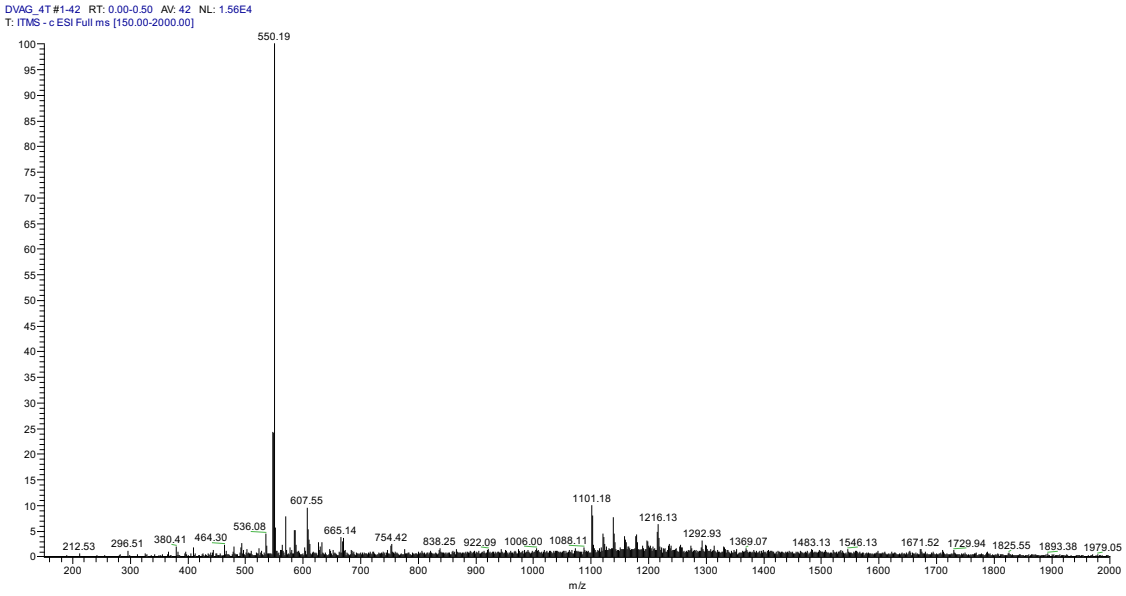

Figure S15: ESI- of HO-DVAG-4T-GAVD-OH.

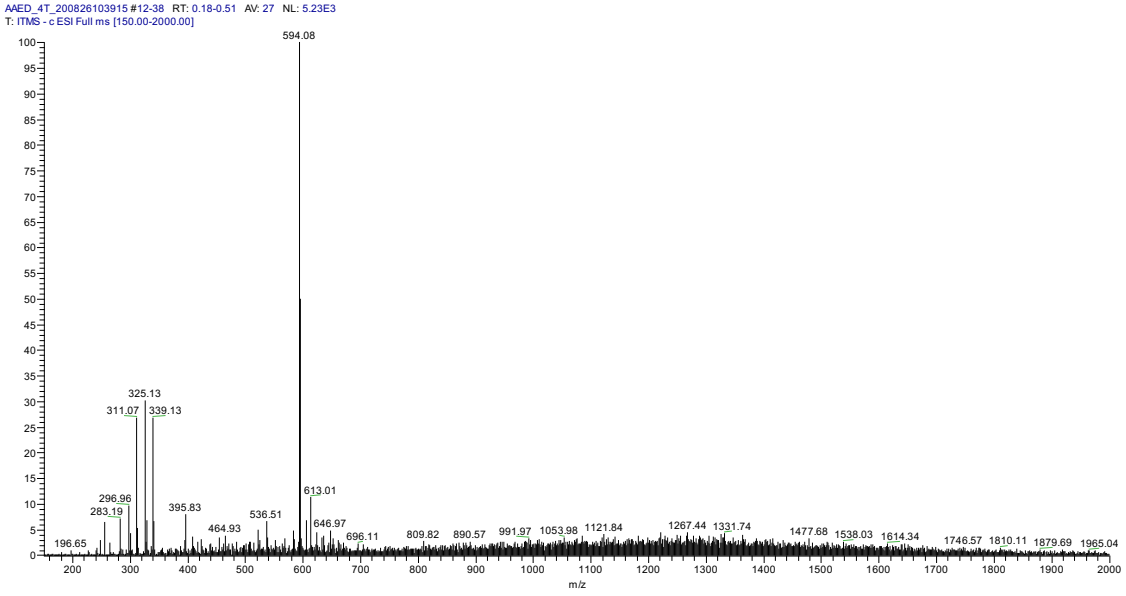

Figure S16: ESI- of HO-AAED-4T-DEAA-OH.

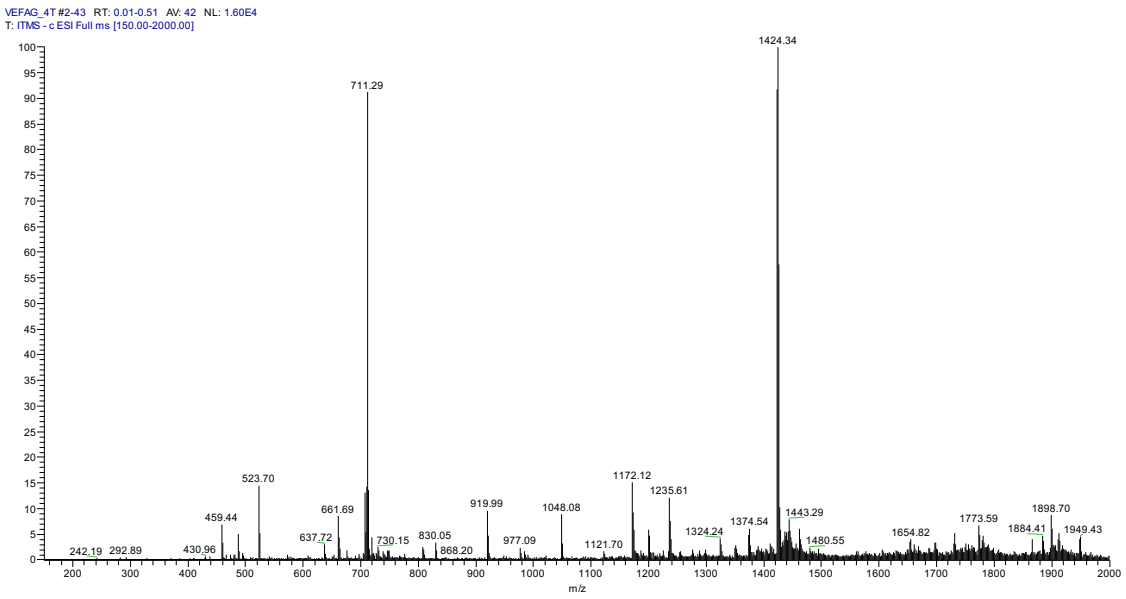

Figure S17: ESI- of HO-VEFAG-4T-GAFEV-OH.

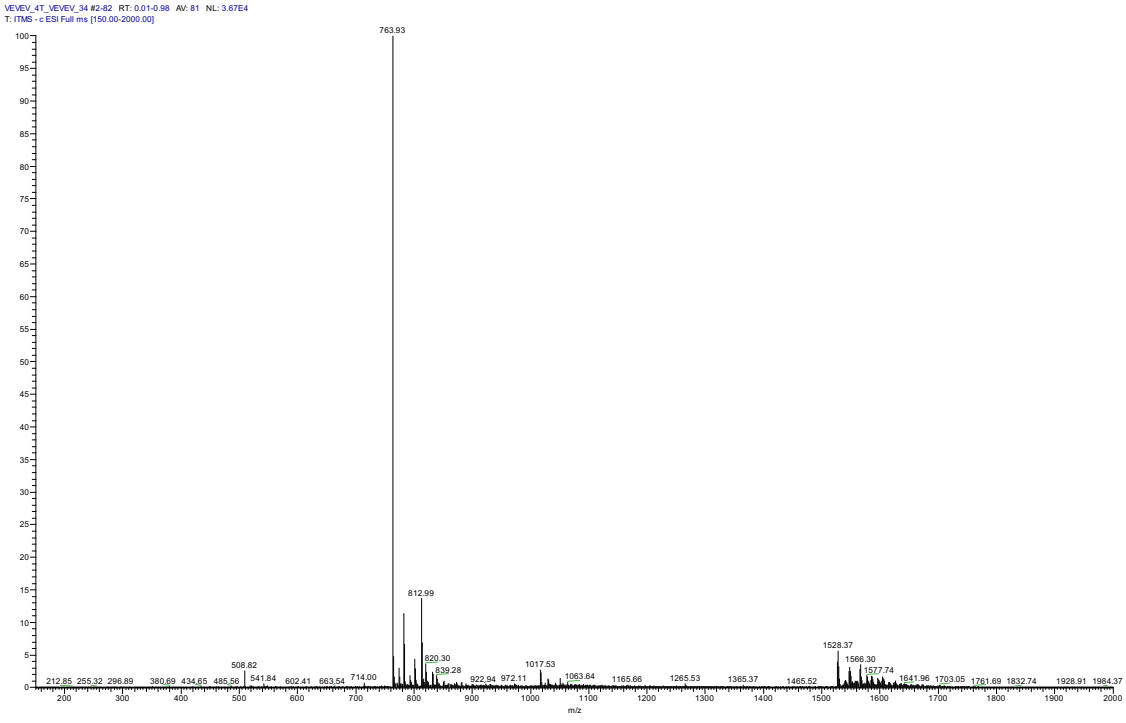

Figure S18: ESI- of HO-VEVEV-4T-VEVEV-OH.

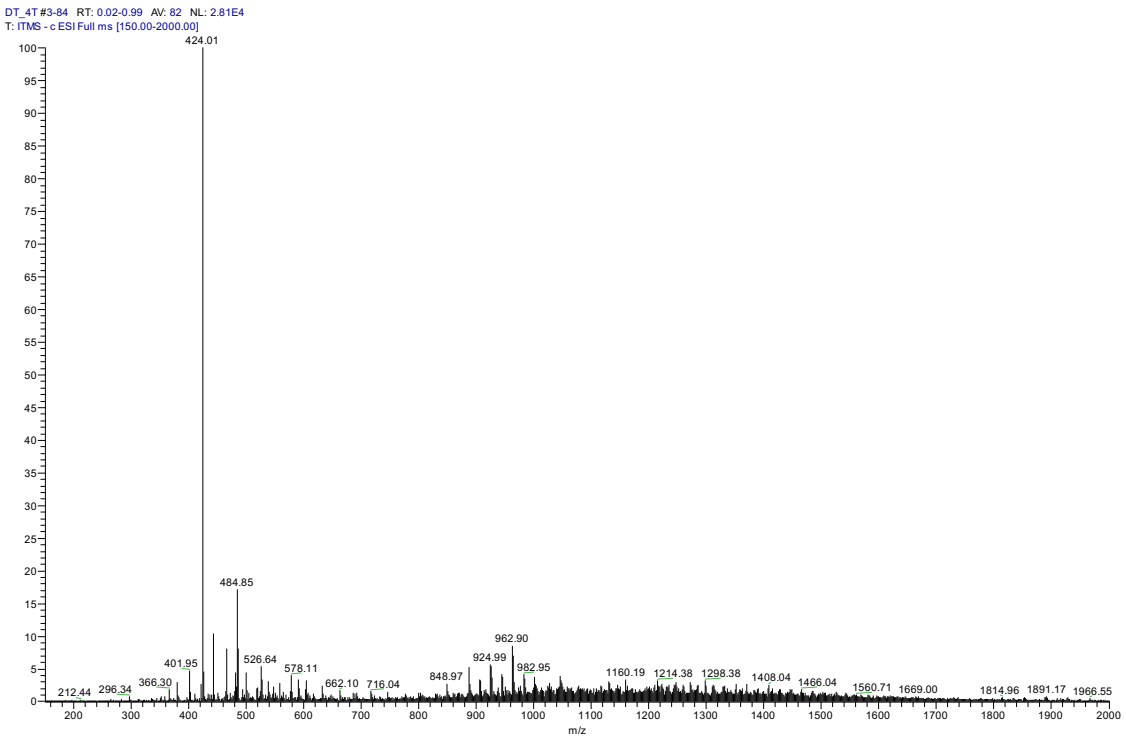

Figure S19: ESI- of HO-DT-4T-TD-OH.

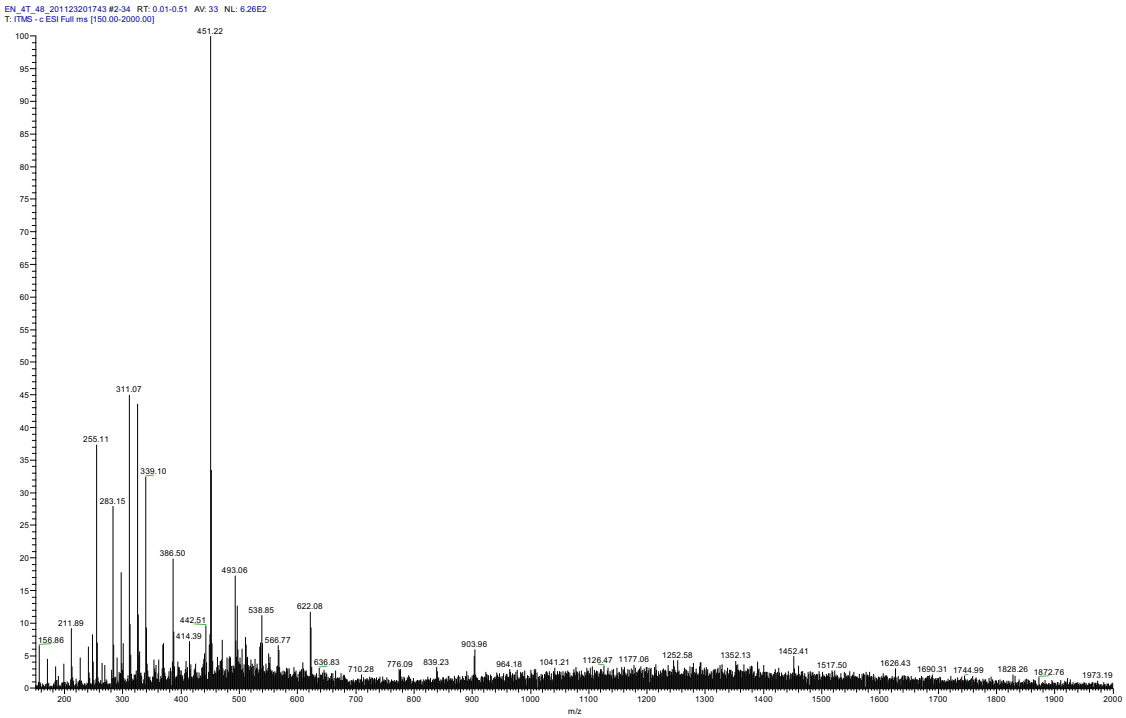

Figure S20: ESI- of HO-EN-4T-NE-OH.

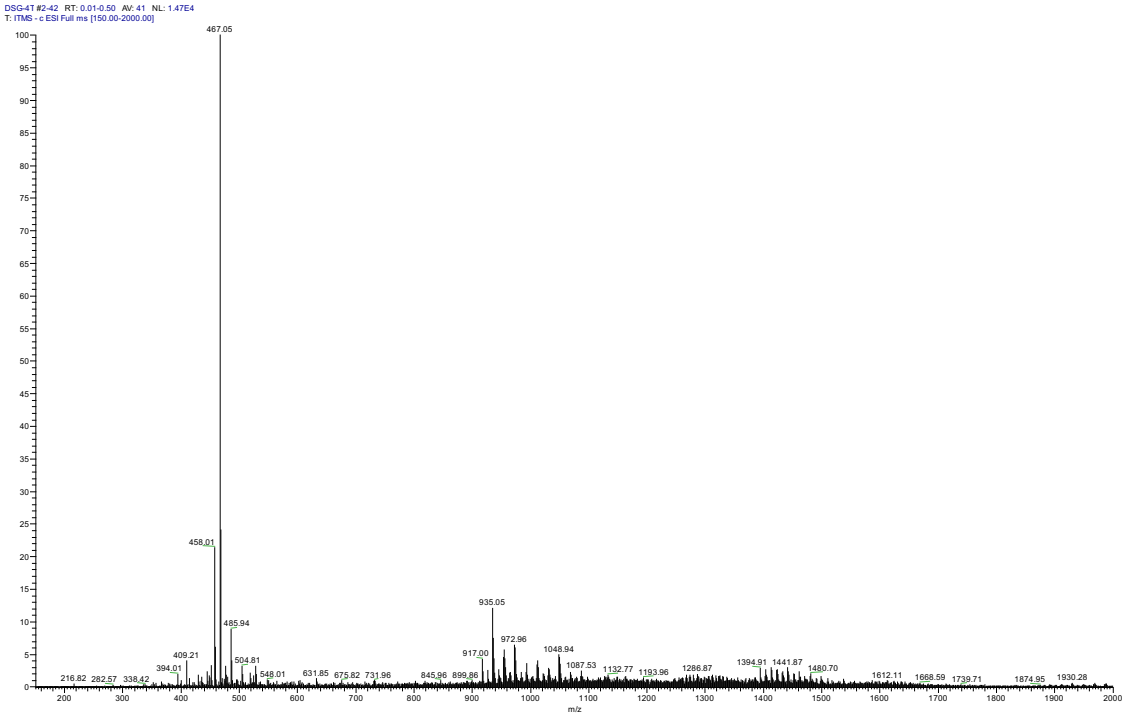

Figure S21: ESI- of HO-DSG-4T-GSD-OH.

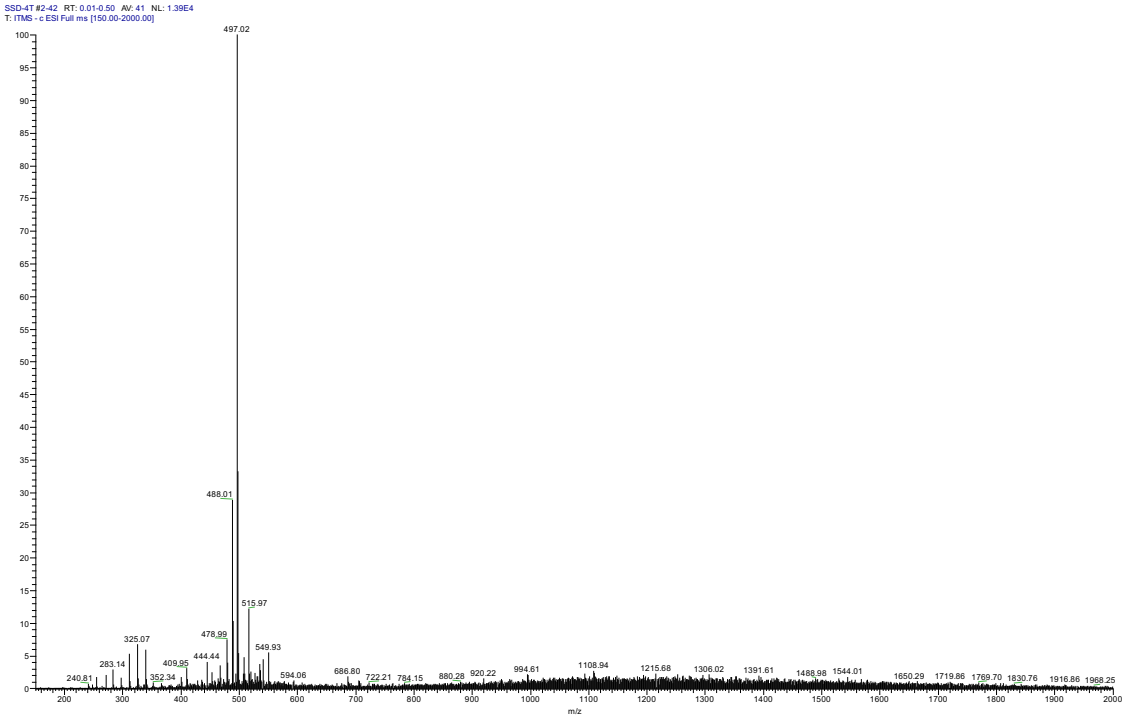

Figure S22: ESI- of HO-SSD-4T-DSS-OH.

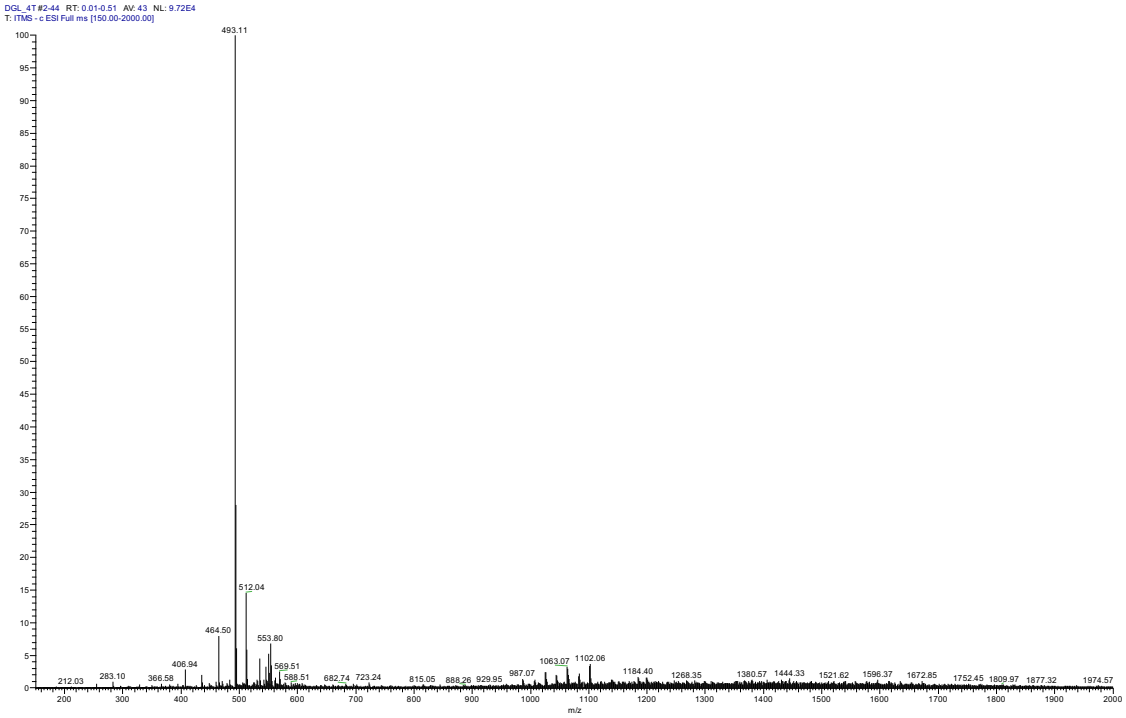

Figure S23: ESI- of HO-DGL-4T-LGD-OH.

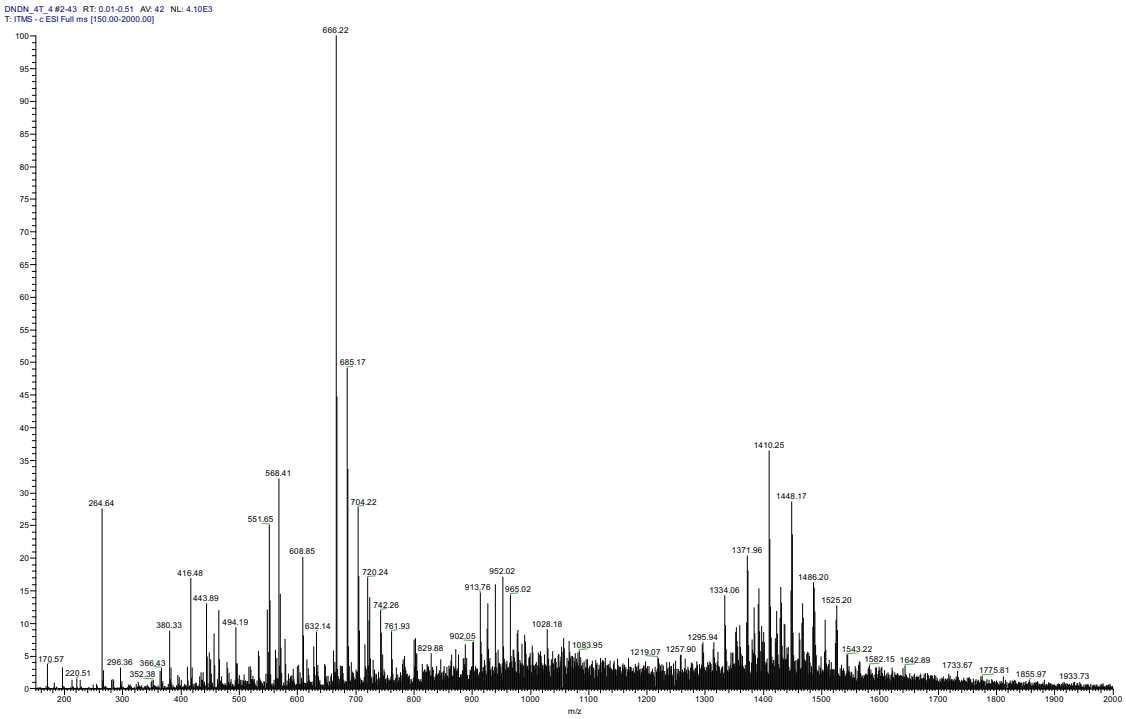

Figure S24: ESI- of HO-DNDN-4T-NDND-OH.

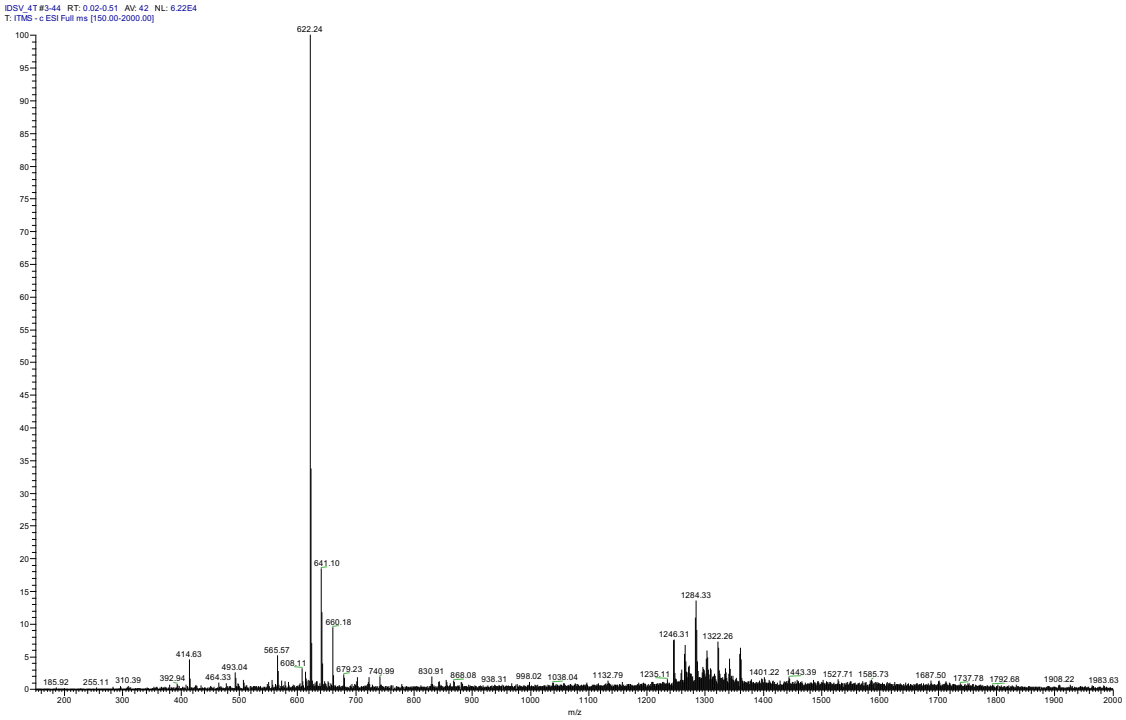

Figure S25: ESI- of HO-IDSV-4T-VSDI-OH.

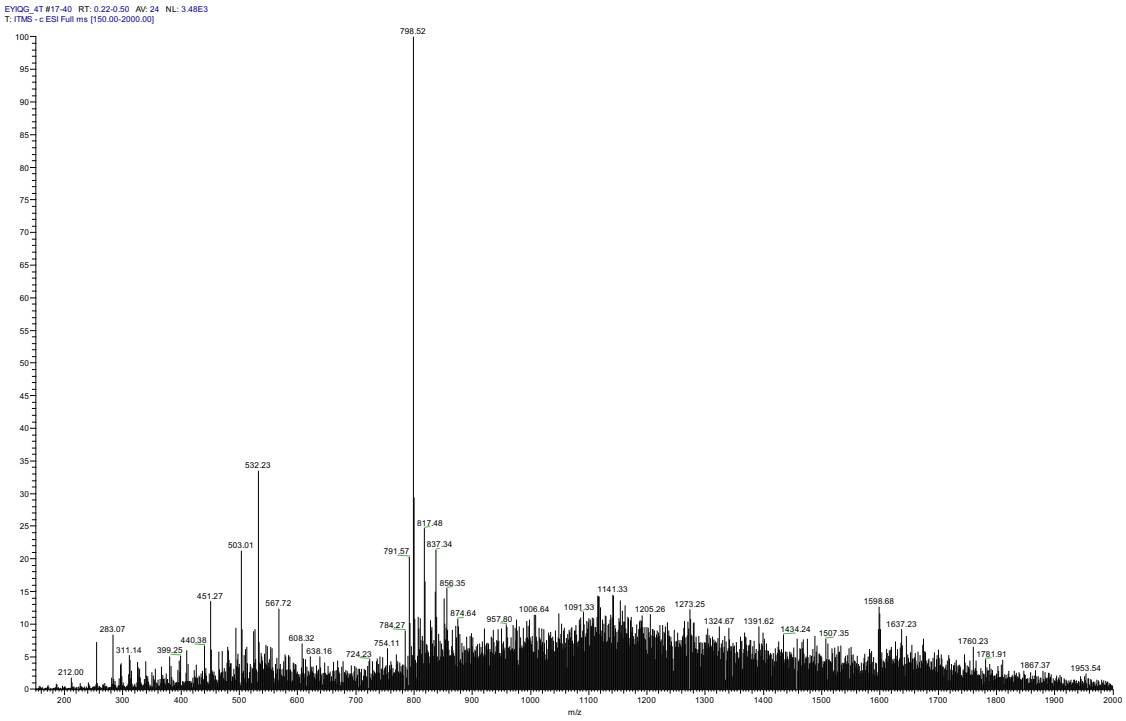

Figure S26: ESI- of HO-EYIQG-4T-GQIYE-OH.

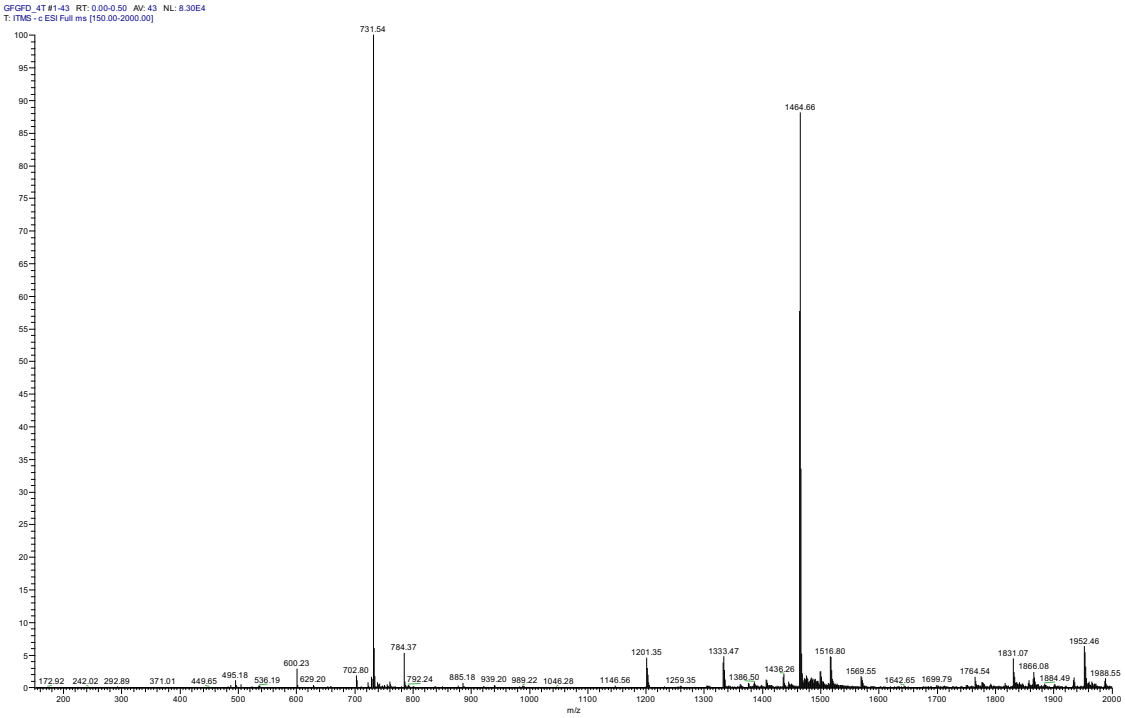

Figure S27: ESI- of HO-GFGFD-4T-DFGFG-OH.

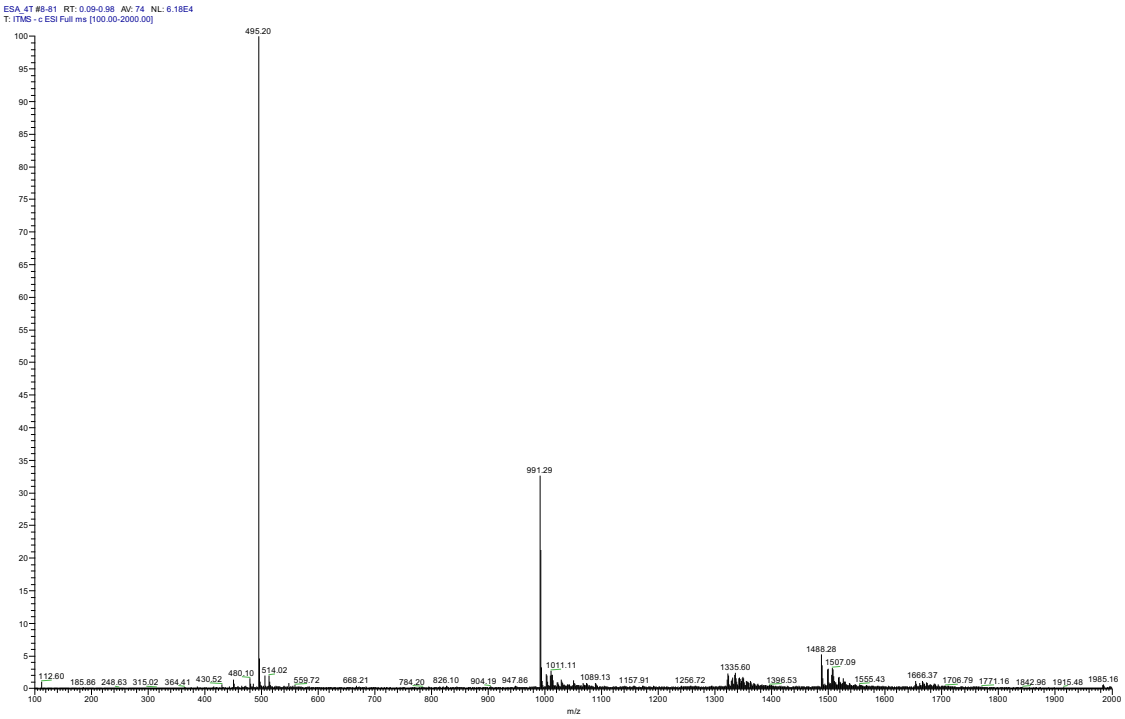

Figure S28: ESI- of HO-ESA-4T-ASE-OH.

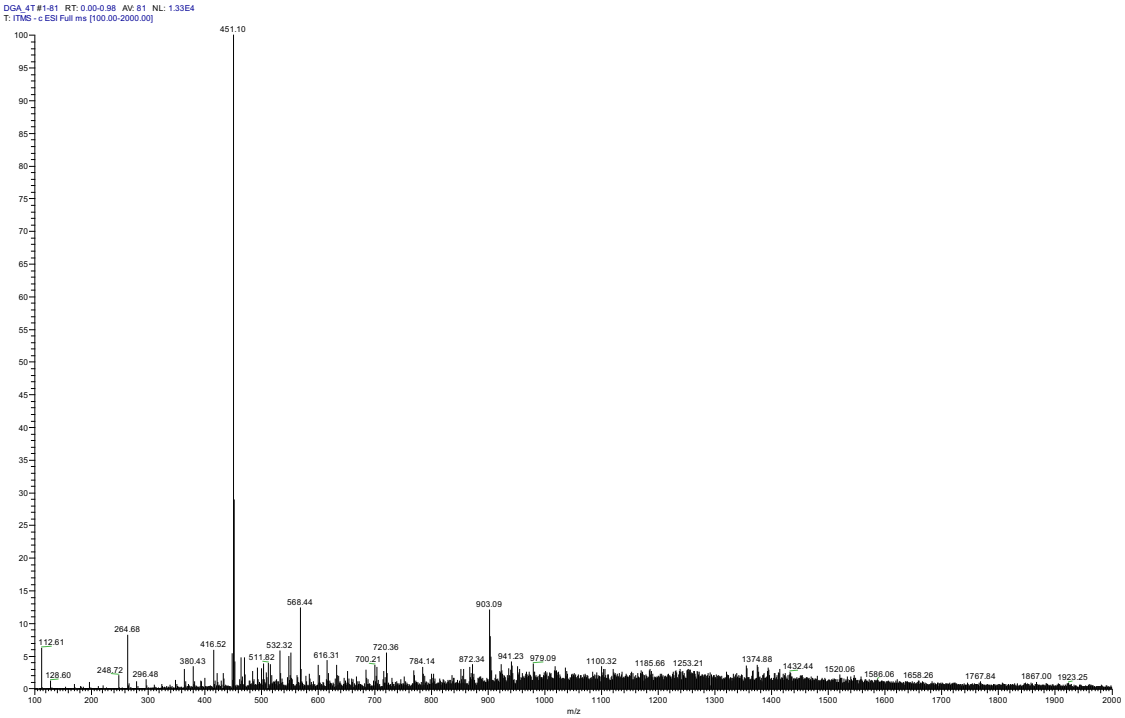

Figure S29: ESI- of HO-DGA-4T-AGD-OH.

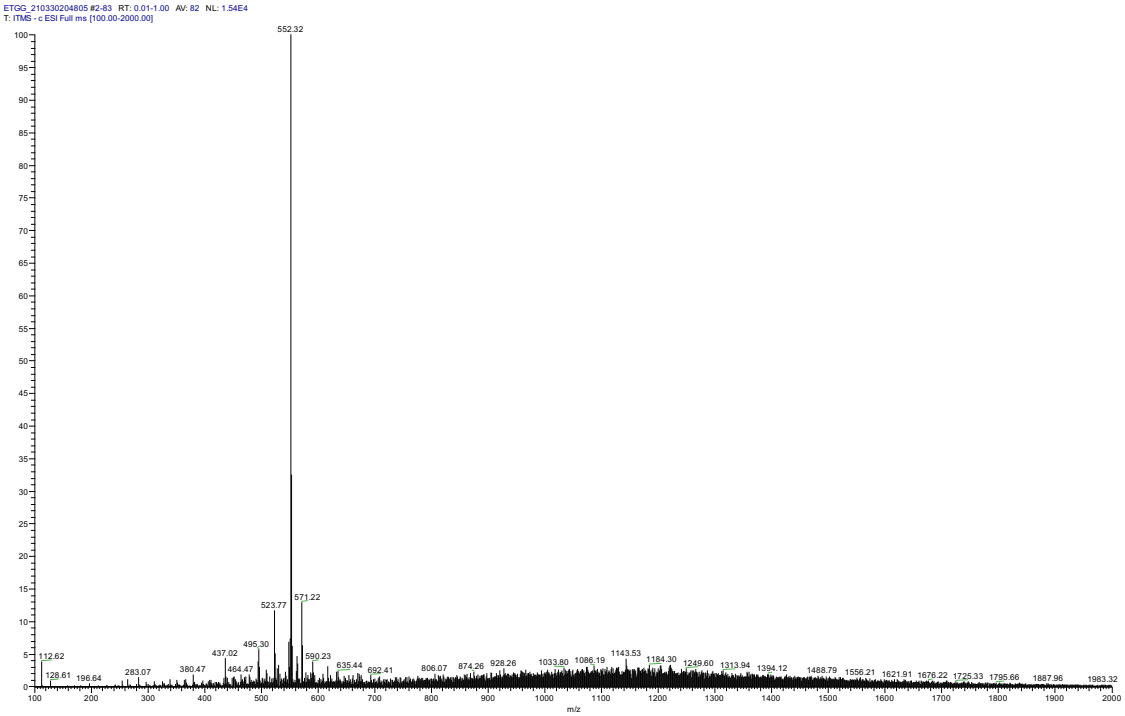

Figure S30: ESI- of HO-ETGG-4T-GGTE-OH.

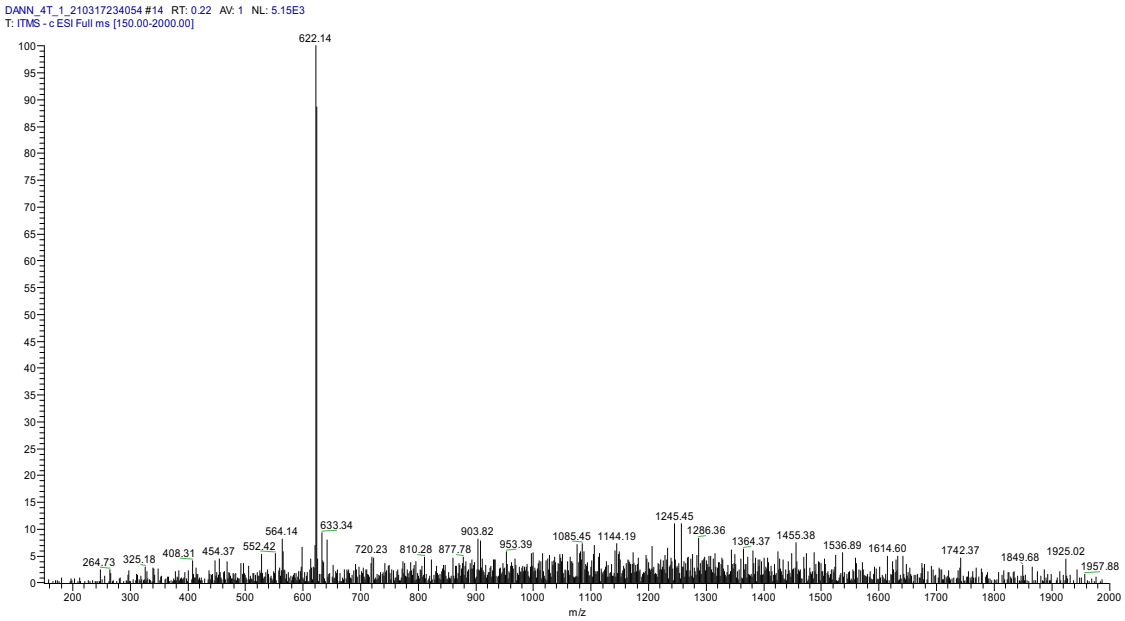

Figure S31: ESI- of HO-DANN-4T-NNAD-OH.

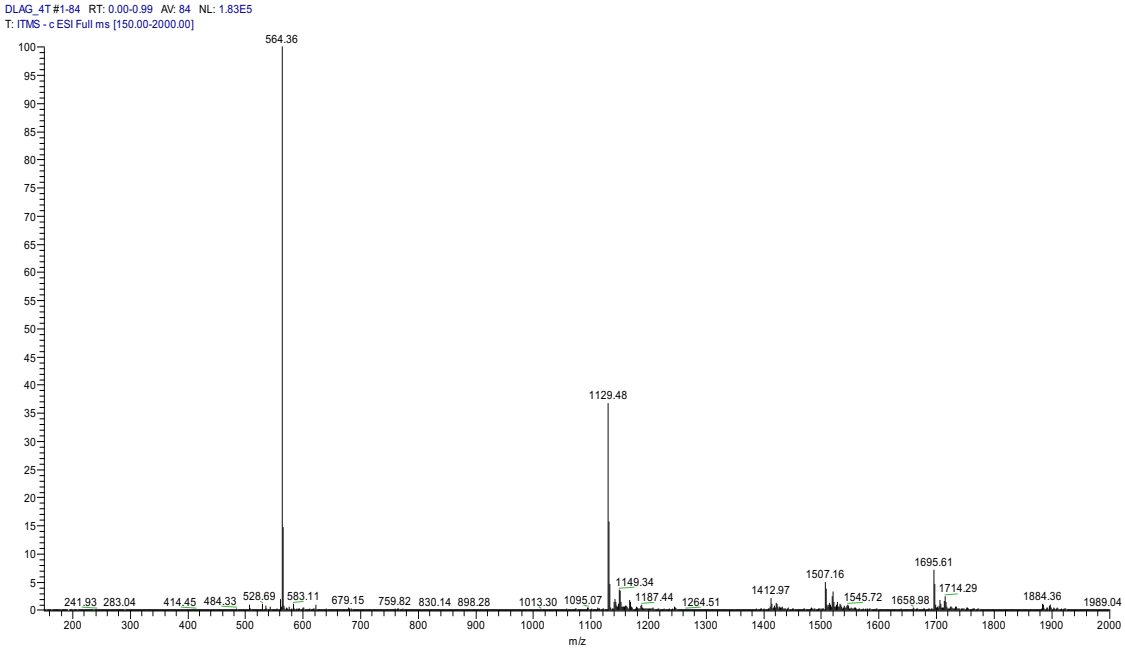

Figure S32: ESI- of HO-DLAG-4T-GALD-OH.

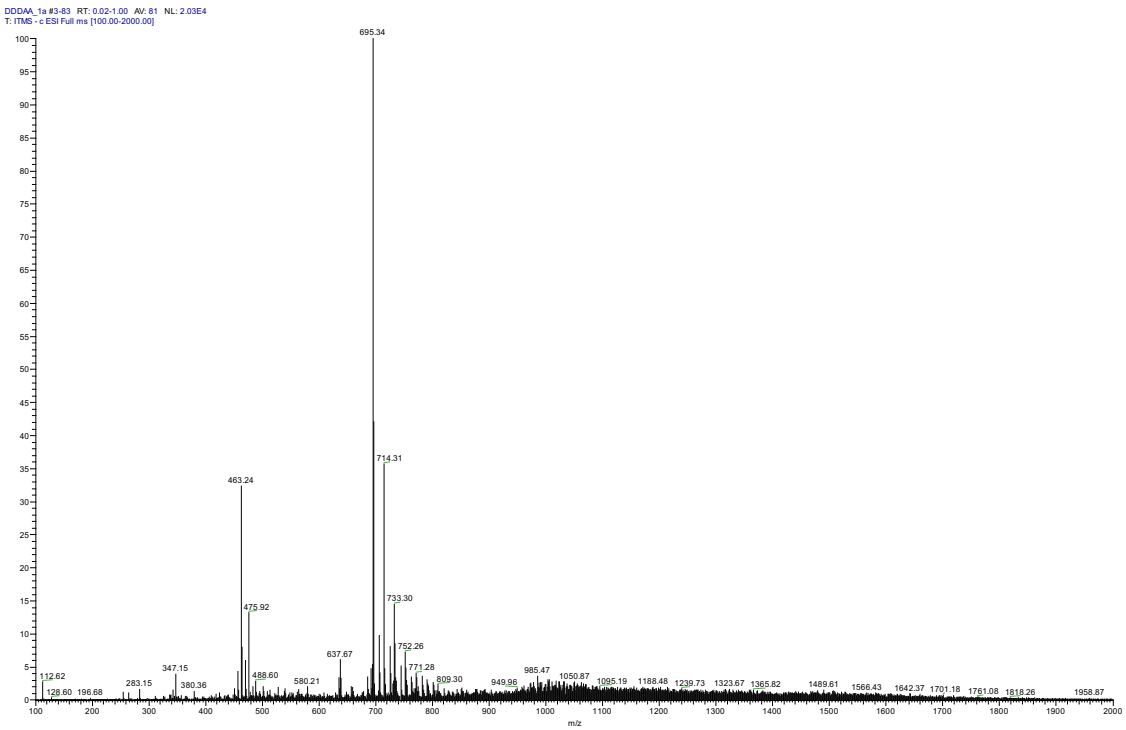

Figure S33: ESI- of HO-DDDAA-4T-AADDD-OH.

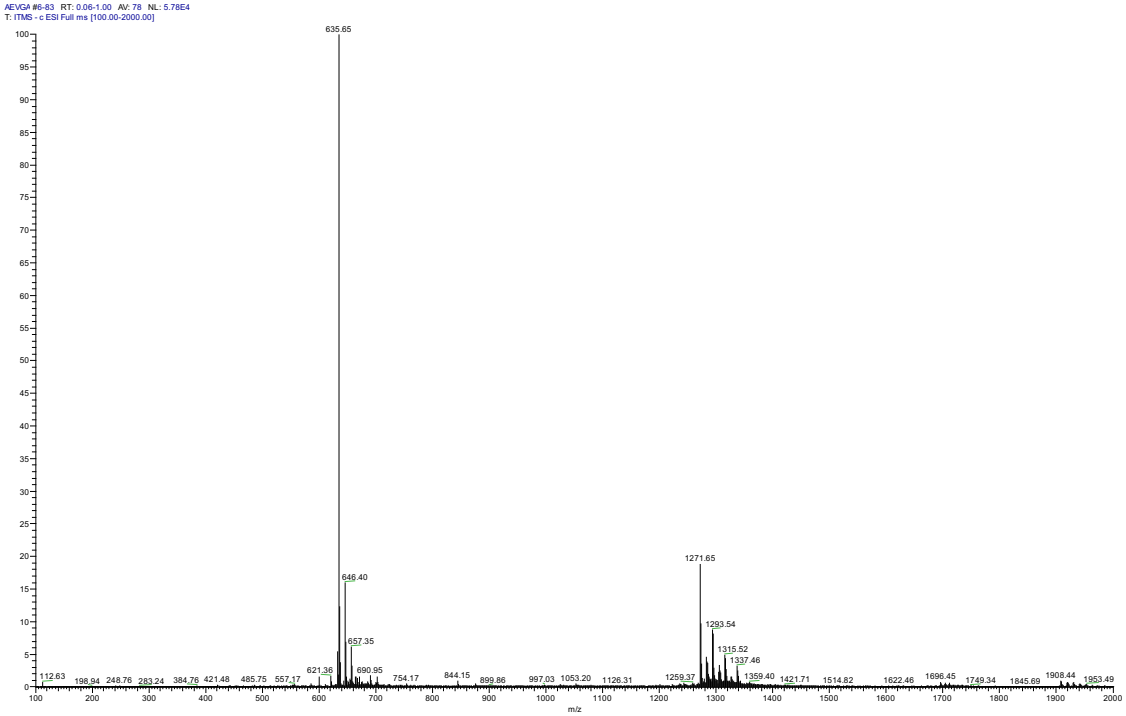

Figure S34: ESI- of HO-AEVGA-4T-AGVEA-OH.

1.11 Analytical HPLC Traces

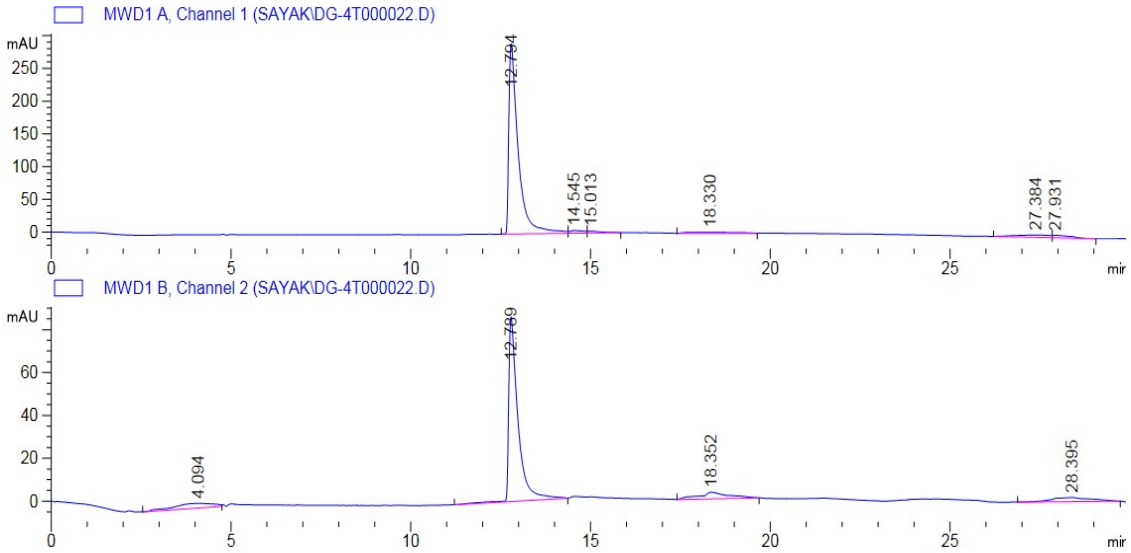

Figure S35: Analytical HPLC trace of HO-DG-4T-GD-OH monitoring 420 nm (top) and 260 nm (bottom).

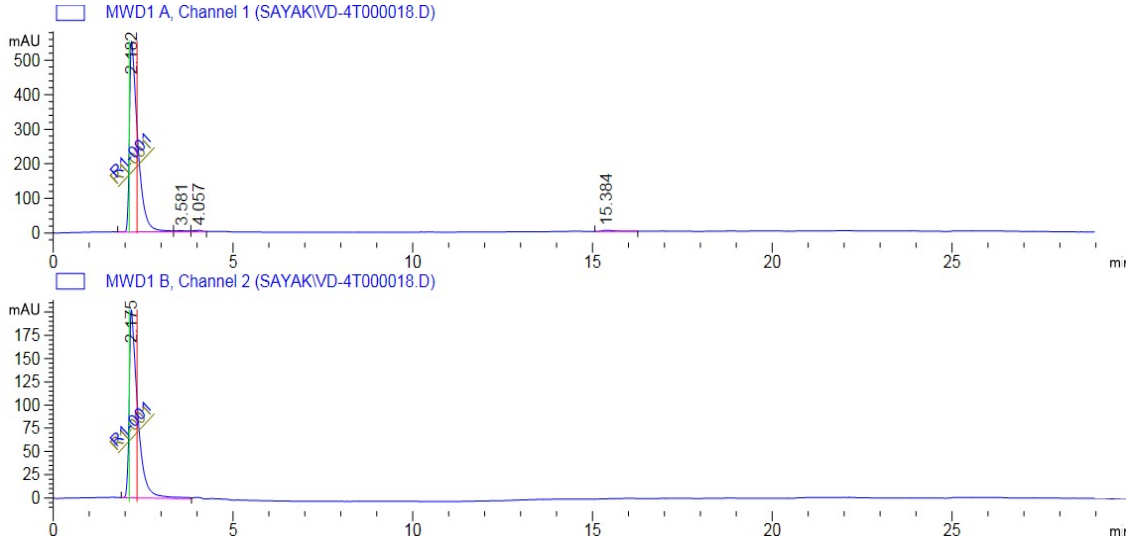

Figure S36: Analytical HPLC trace of HO-VD-4T-DV-OH monitoring 420 nm (top) and 260 nm (bottom).

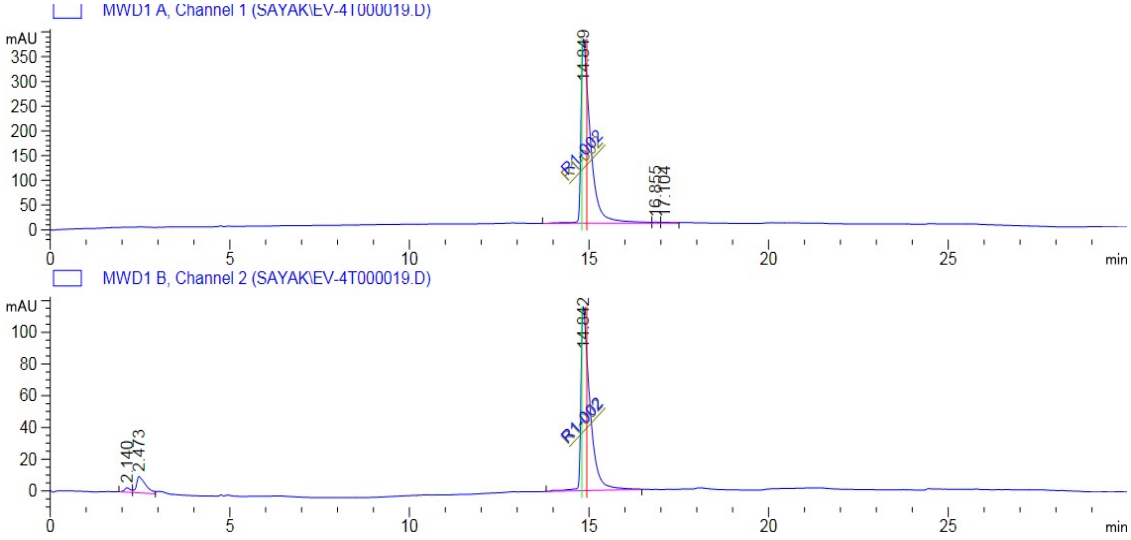

Figure S37: Analytical HPLC trace of HO-EV-4T-VE-OH monitoring 420 nm (top) and 260 nm (bottom).

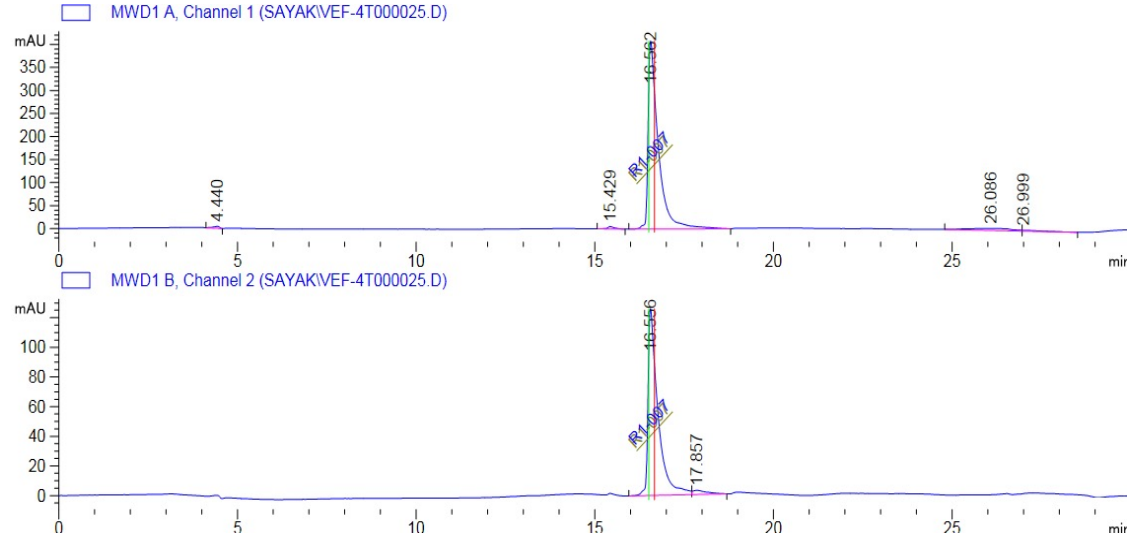

Figure S38: Analytical HPLC trace of HO-VEF-4T-FEV-OH monitoring 420 nm (top) and 260 nm (bottom).

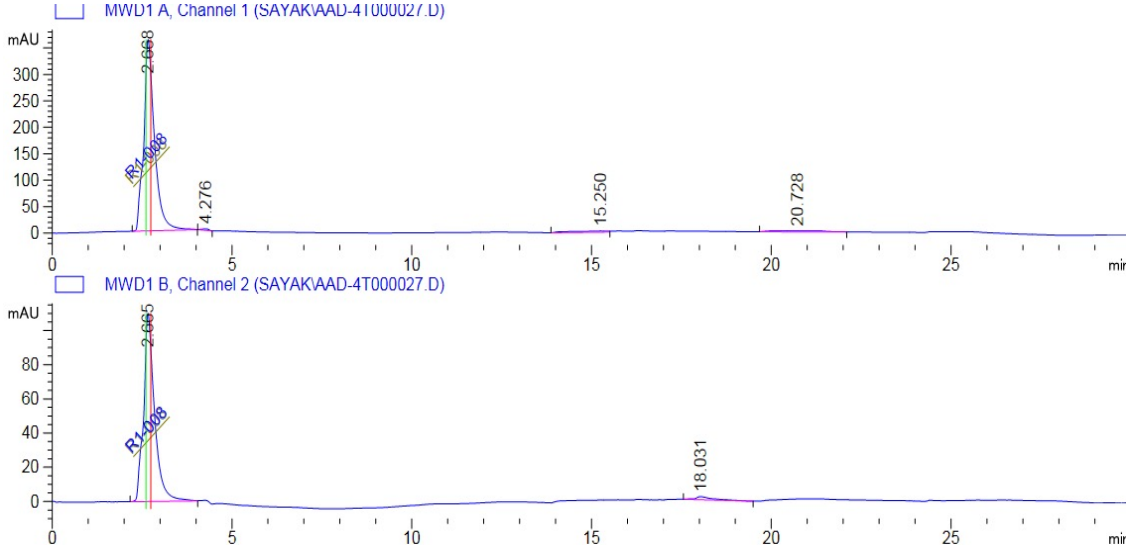

Figure S39: Analytical HPLC trace of HO-AAD-4T-DAA-OH monitoring 420 nm (top) and 260 nm (bottom).

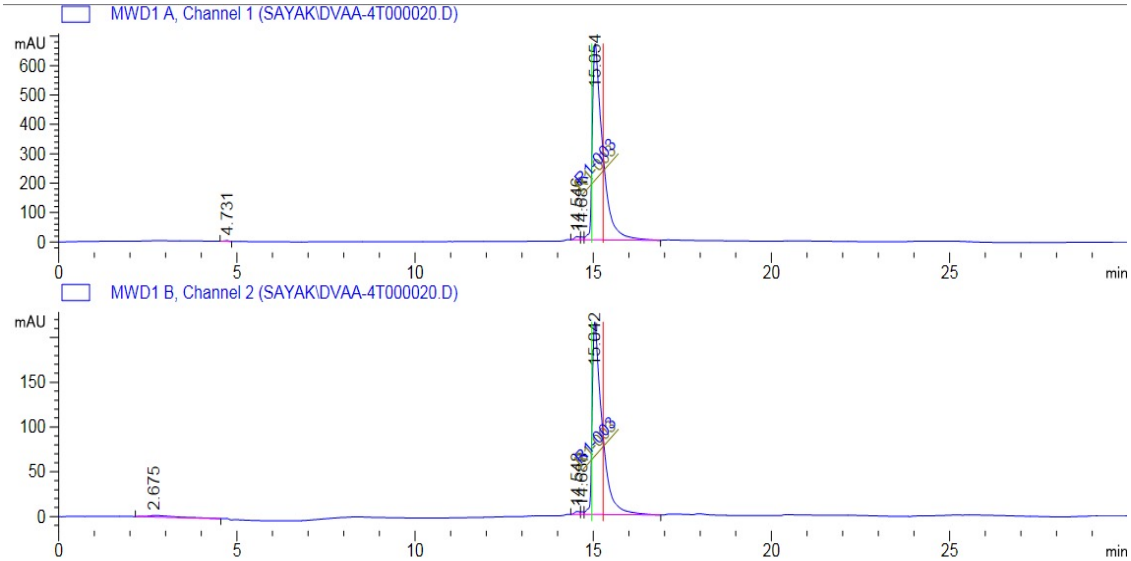

Figure S40: Analytical HPLC trace of HO-DVAA-4T-AAVD-OH monitoring 420 nm (top) and 260 nm (bottom).

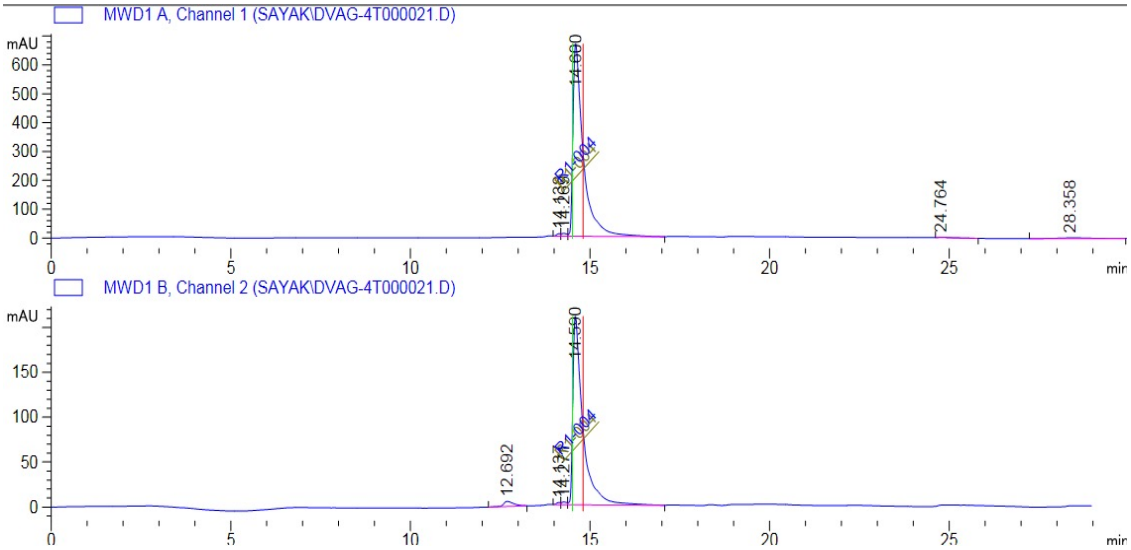

Figure S41: Analytical HPLC trace of HO-DVAG-4T-GAVD-OH monitoring 420 nm (top) and 260 nm (bottom).

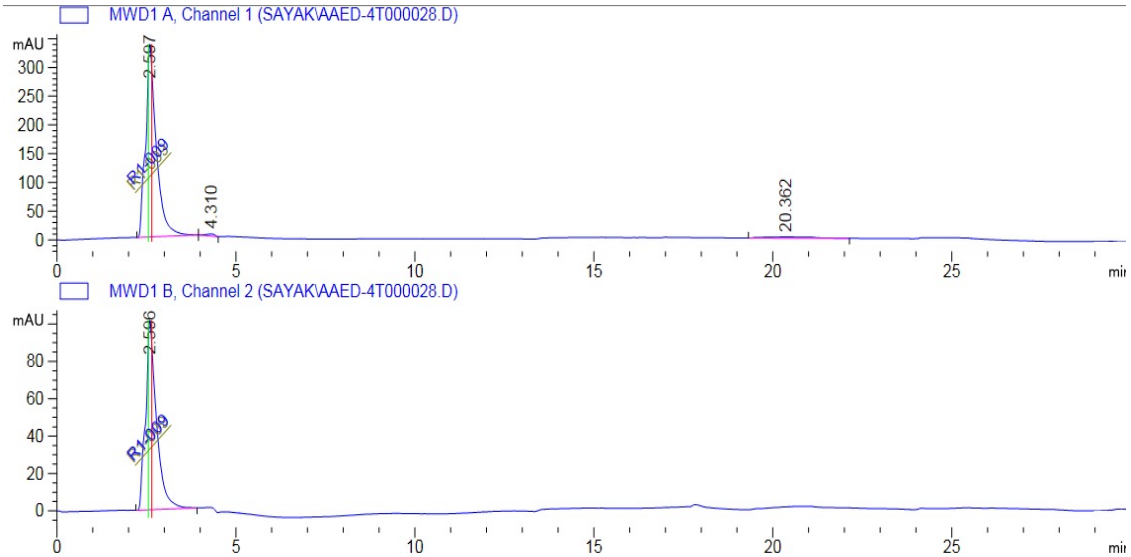

Figure S42: Analytical HPLC trace of HO-AAED-4T-DEAA-OH monitoring 420 nm (top) and 260 nm (bottom).

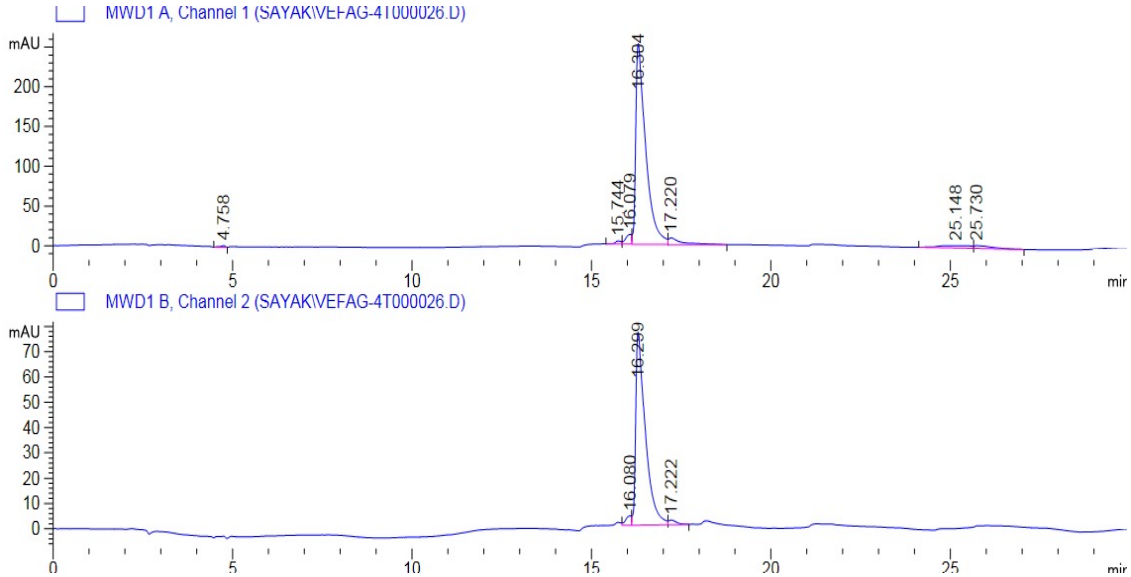

Figure S43: Analytical HPLC trace of HO-VEFAG-4T-GAFEV-OH monitoring 420 nm (top) and 260 nm (bottom).

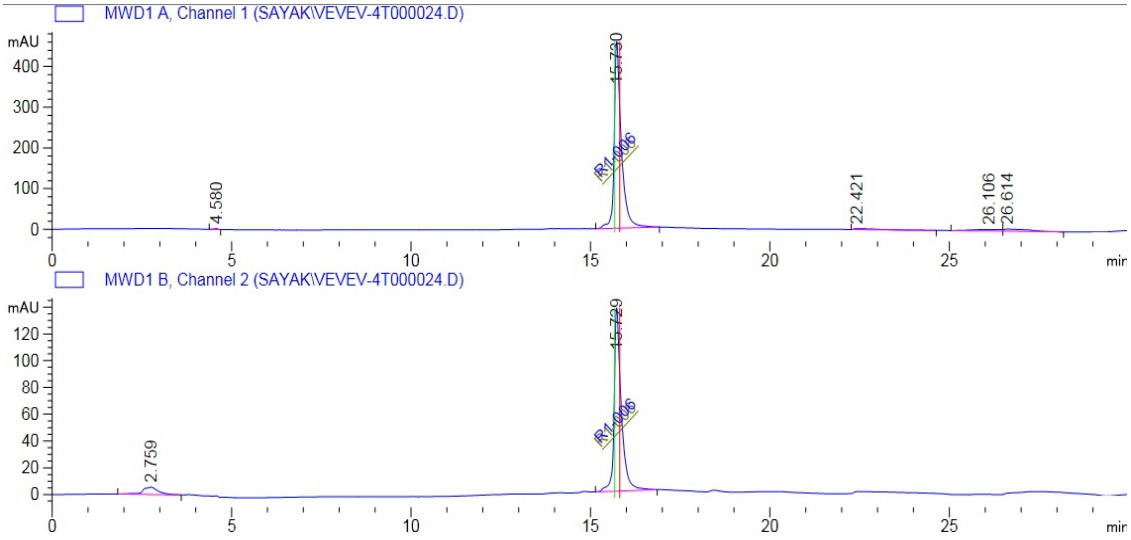

Figure S44: Analytical HPLC trace of HO-VEVEV-4T-VEVEV-OH monitoring 420 nm (top) and 260 nm (bottom).

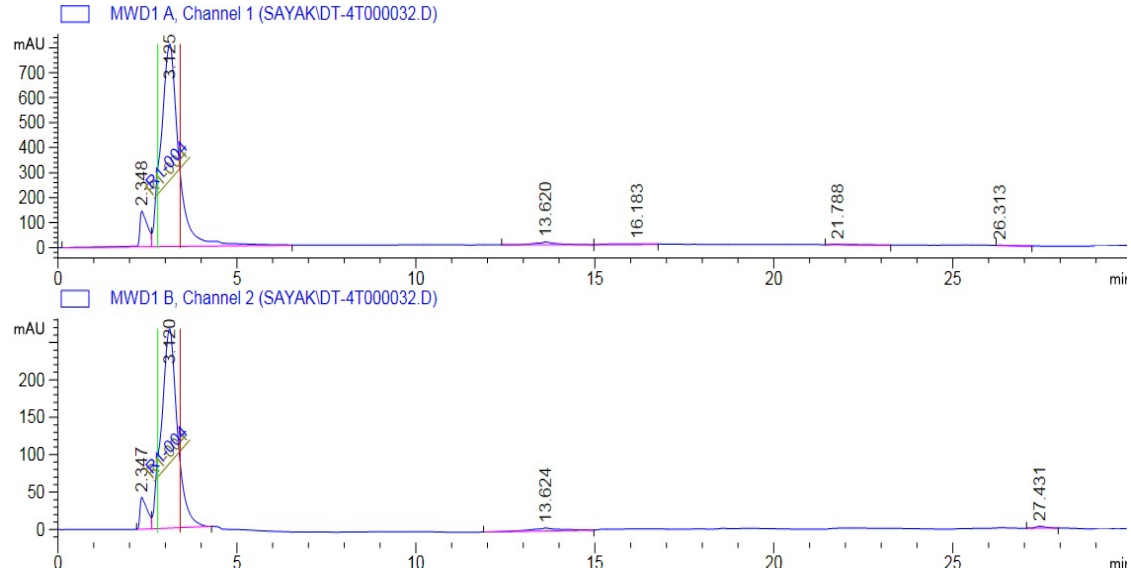

Figure S45: Analytical HPLC trace of HO-DT-4T-TD-OH monitoring 420 nm (top) and 260 nm (bottom).

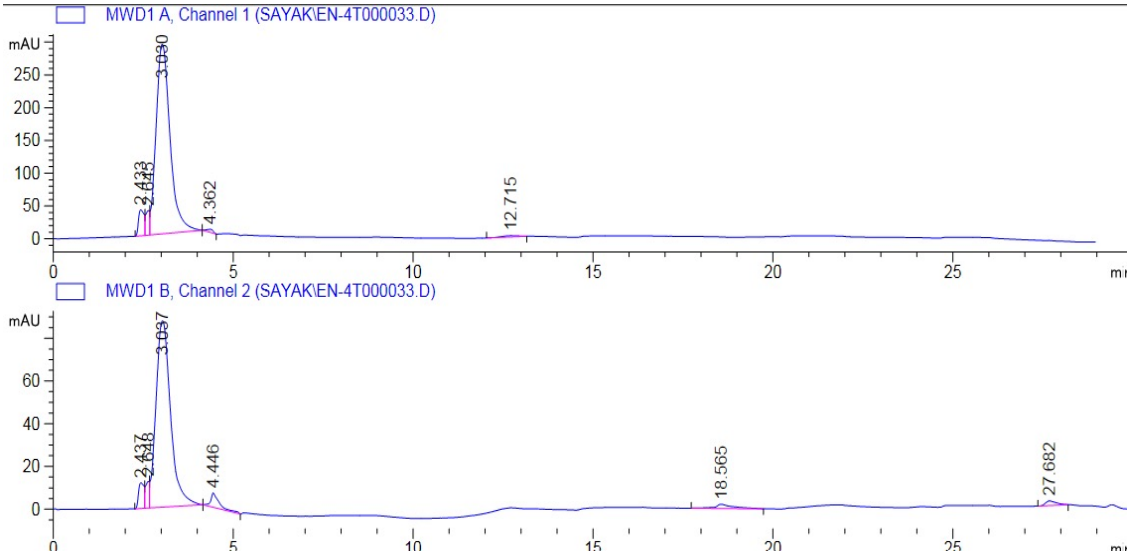

Figure S46: Analytical HPLC trace of HO-EN-4T-NE-OH monitoring 420 nm (top) and 260 nm (bottom).

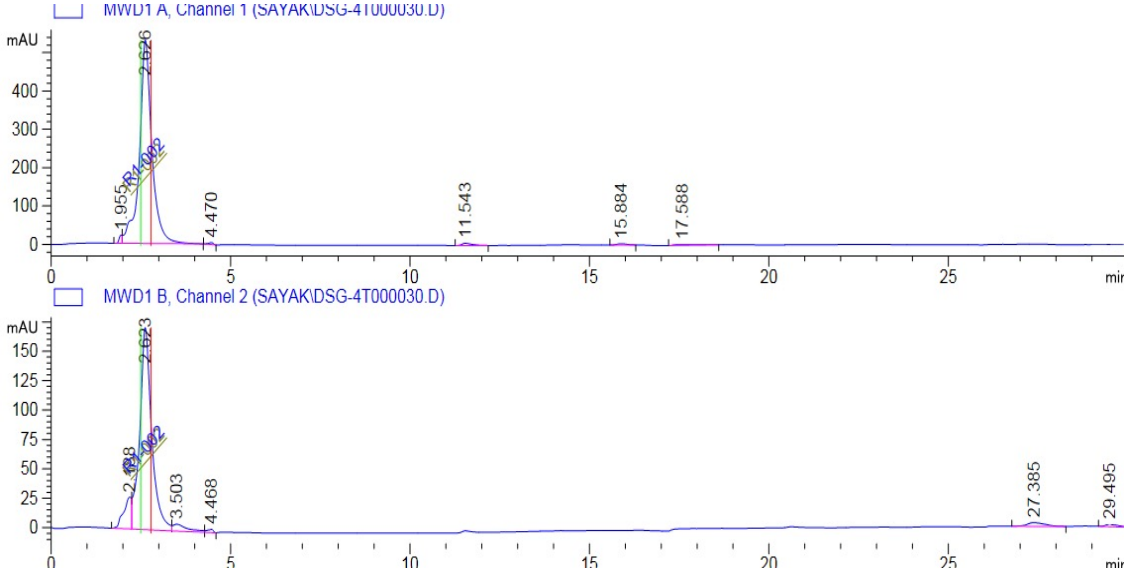

Figure S47: Analytical HPLC trace of HO-DSG-4T-GSD-OH monitoring 420 nm (top) and 260 nm (bottom).

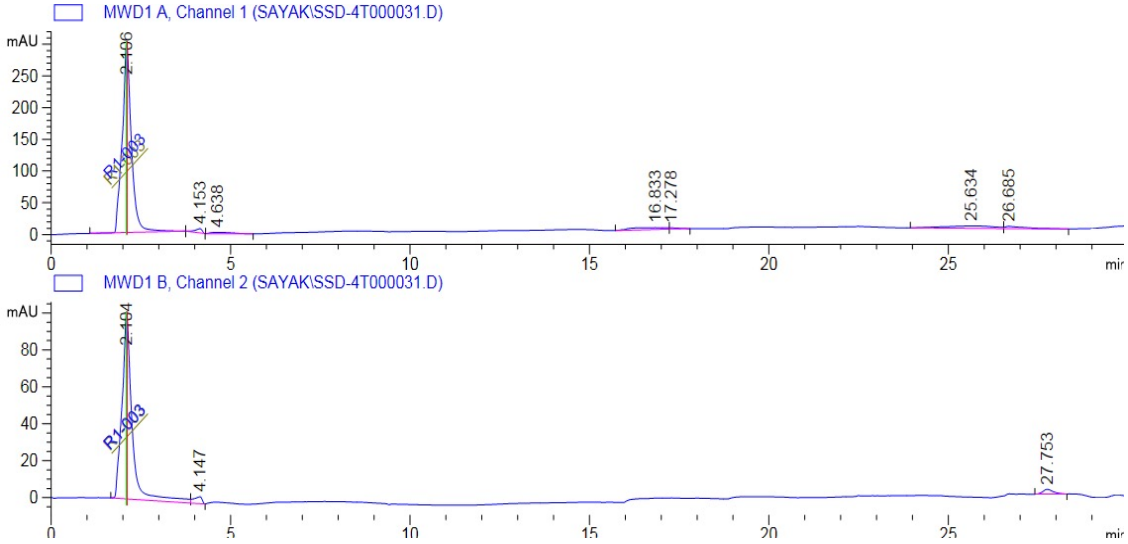

Figure S48: Analytical HPLC trace of HO-SSD-4T-DSS-OH monitoring 420 nm (top) and 260 nm (bottom).

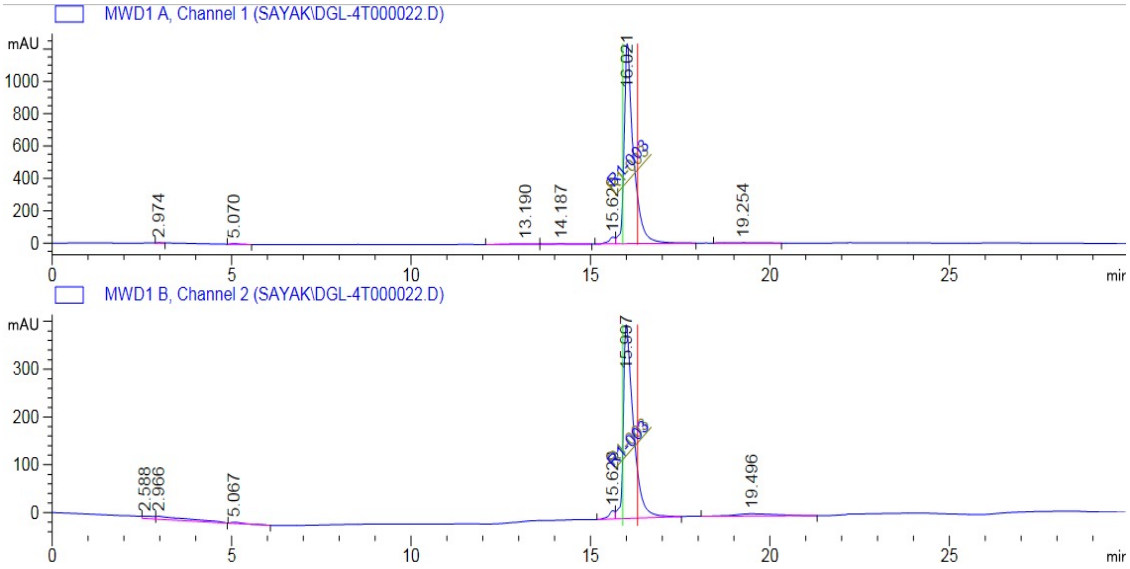

Figure S49: Analytical HPLC trace of HO-DGL-4T-LGD-OH monitoring 420 nm (top) and 260 nm (bottom).

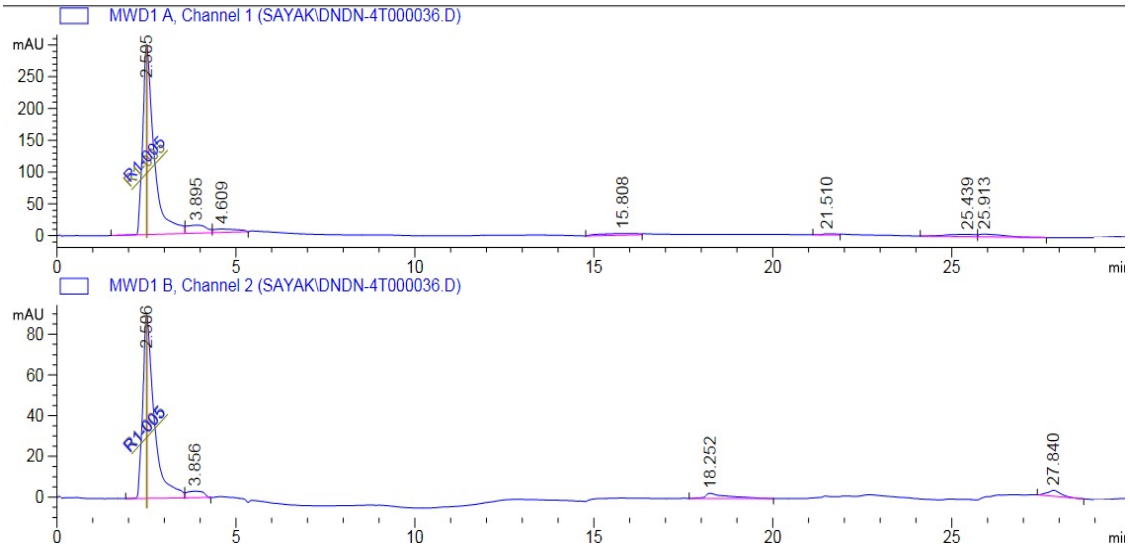

Figure S50: Analytical HPLC trace of HO-DNDN-4T-NDND-OH monitoring 420 nm (top) and 260 nm (bottom).

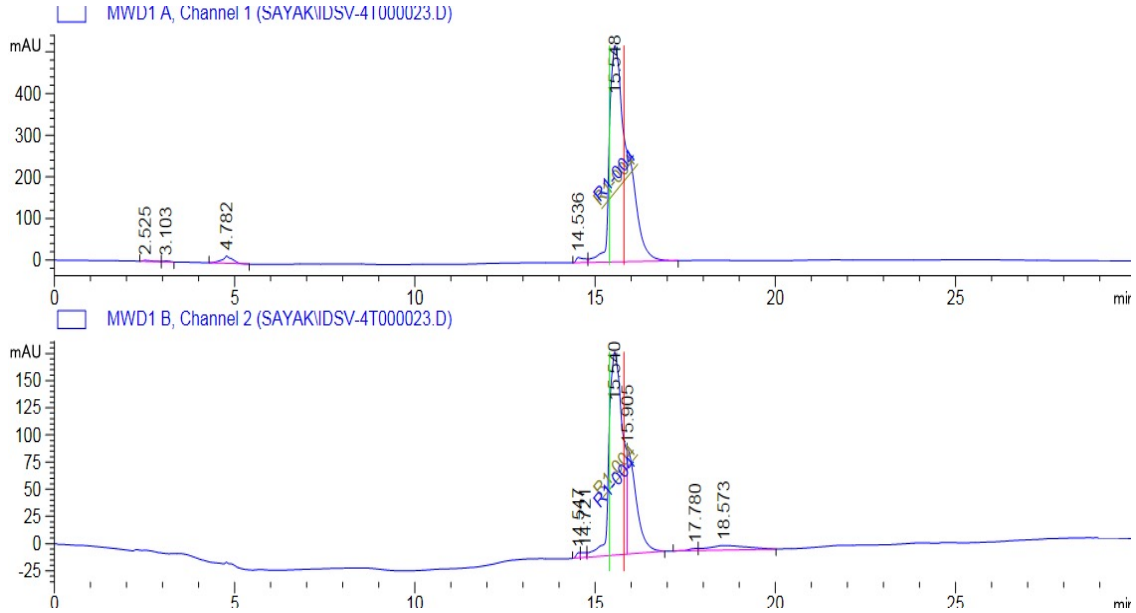

Figure S51: Analytical HPLC trace of HO-IDSV-4T-VSDI-OH monitoring 420 nm (top) and 260 nm (bottom).

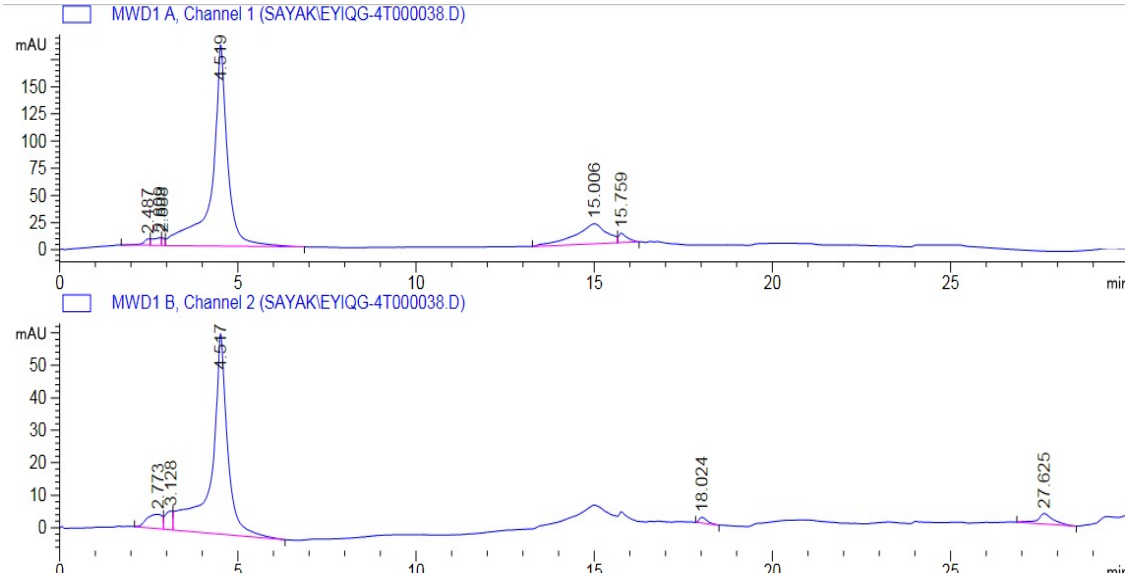

Figure S52: Analytical HPLC trace of HO-EYIQG-4T-GQIYE-OH monitoring 420 nm (top) and 260 nm (bottom).

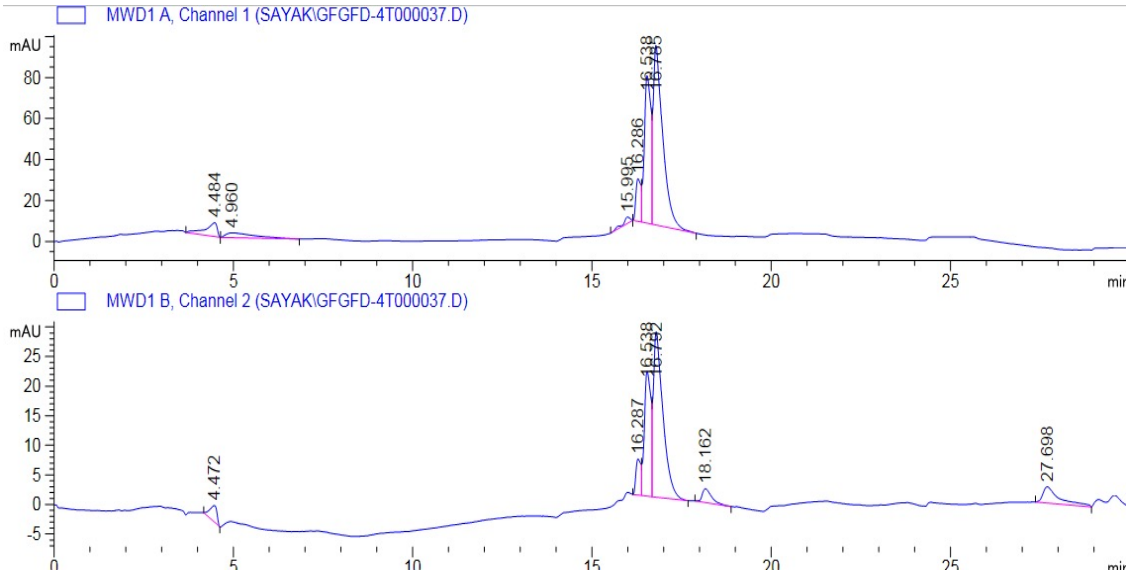

Figure S53: Analytical HPLC trace of HO-GFGFD-4T-DFGFG-OH monitoring 420 nm (top) and 260 nm (bottom).

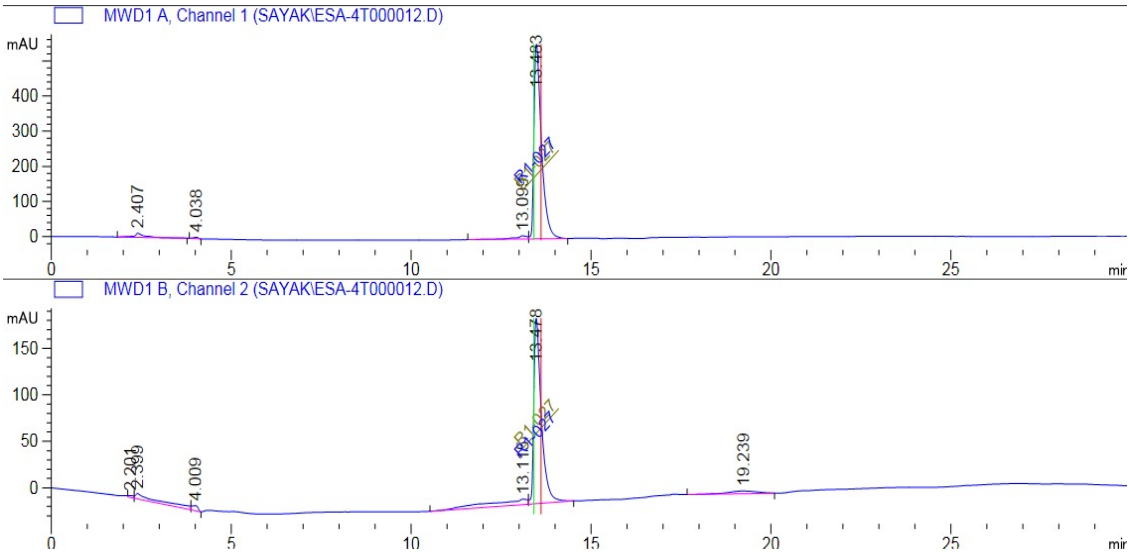

Figure S54: Analytical HPLC trace of HO-ESA-4T-ASE-OH monitoring 420 nm (top) and 260 nm (bottom).

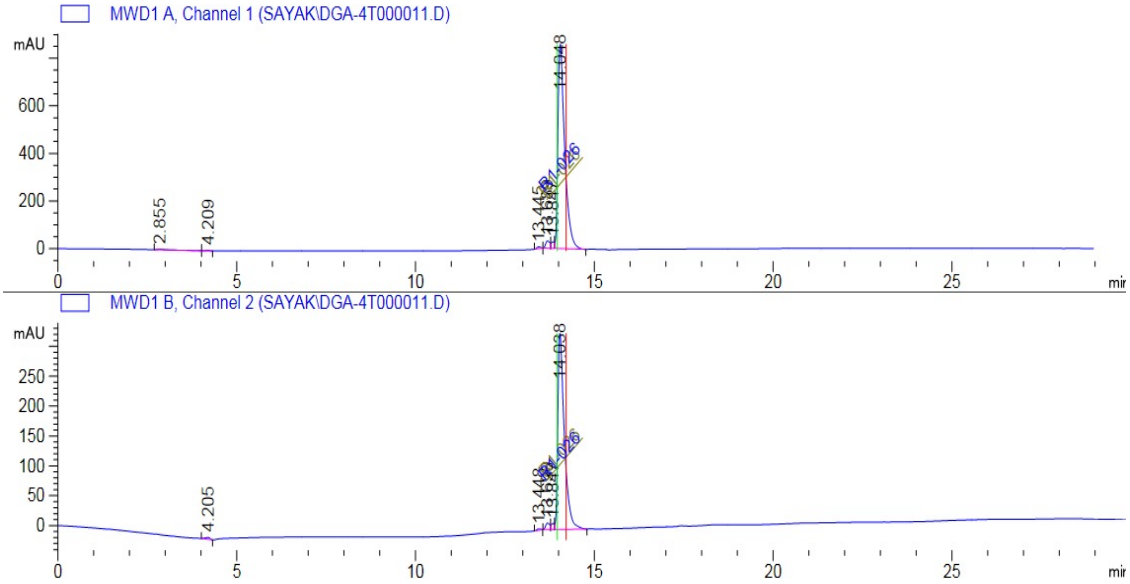

Figure S55: Analytical HPLC trace of HO-DGA-4T-AGD-OH monitoring 420 nm (top) and 260 nm (bottom).

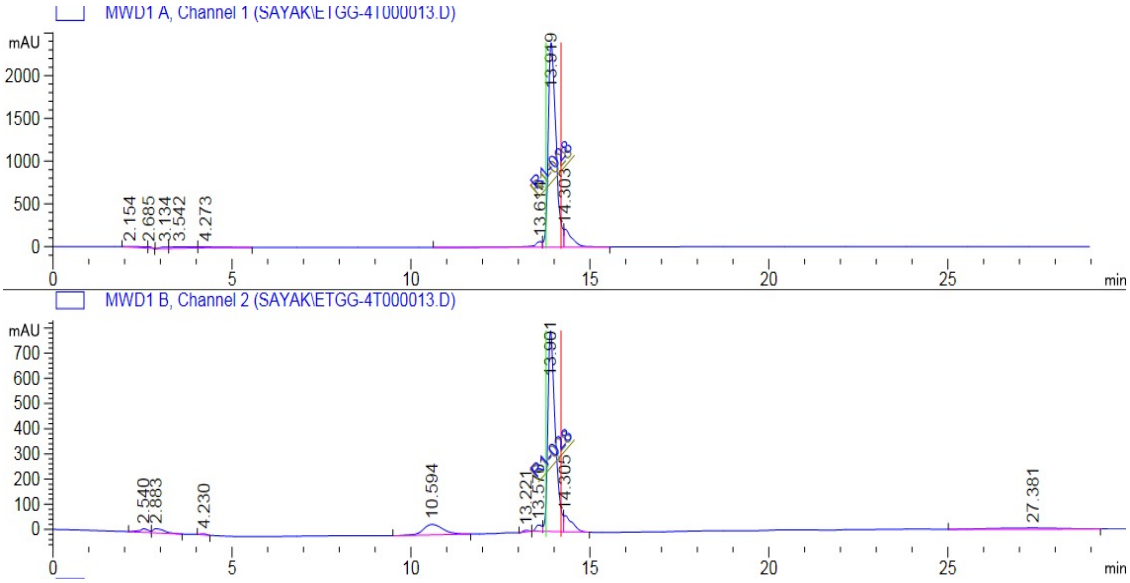

Figure S56: Analytical HPLC trace of HO-ETGG-4T-GGTE-OH monitoring 420 nm (top) and 260 nm (bottom).

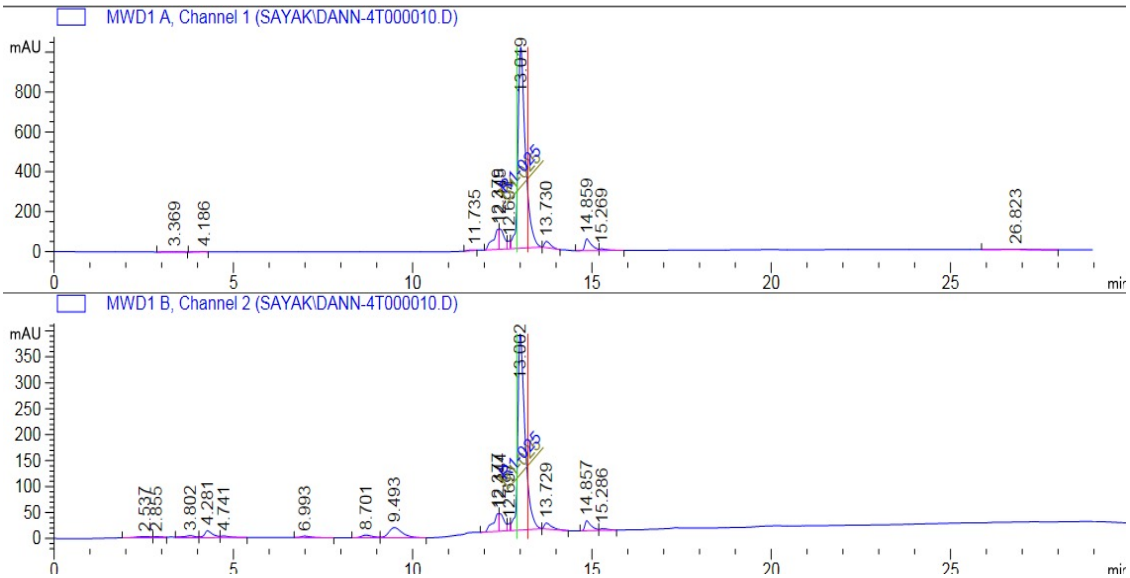

Figure S57: Analytical HPLC trace of HO-DANN-4T-NNAD-OH monitoring 420 nm (top) and 260 nm (bottom).

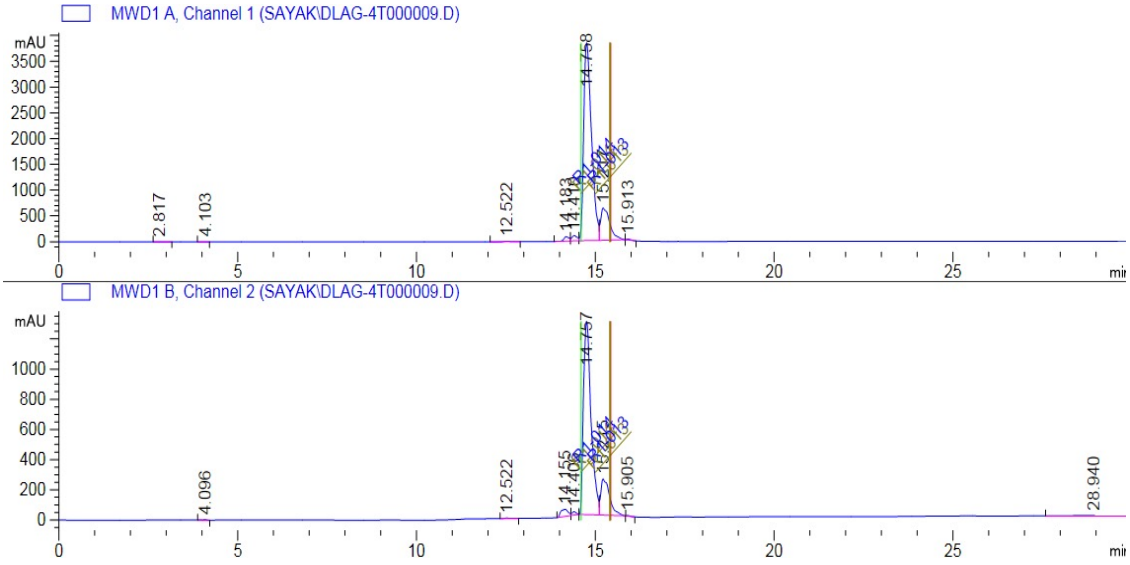

Figure S58: Analytical HPLC trace of HO-DLAG-4T-GALD-OH monitoring 420 nm (top) and 260 nm (bottom).

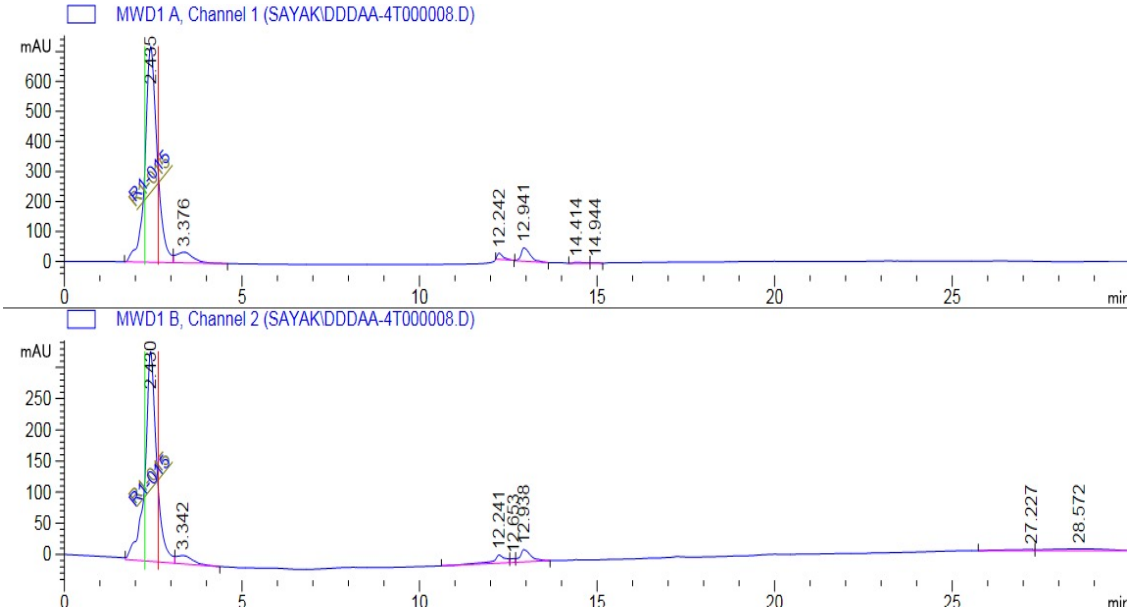

Figure S59: Analytical HPLC trace of HO-DDDAA-4T-AADDD-OH monitoring 420 nm (top) and 260 nm (bottom).

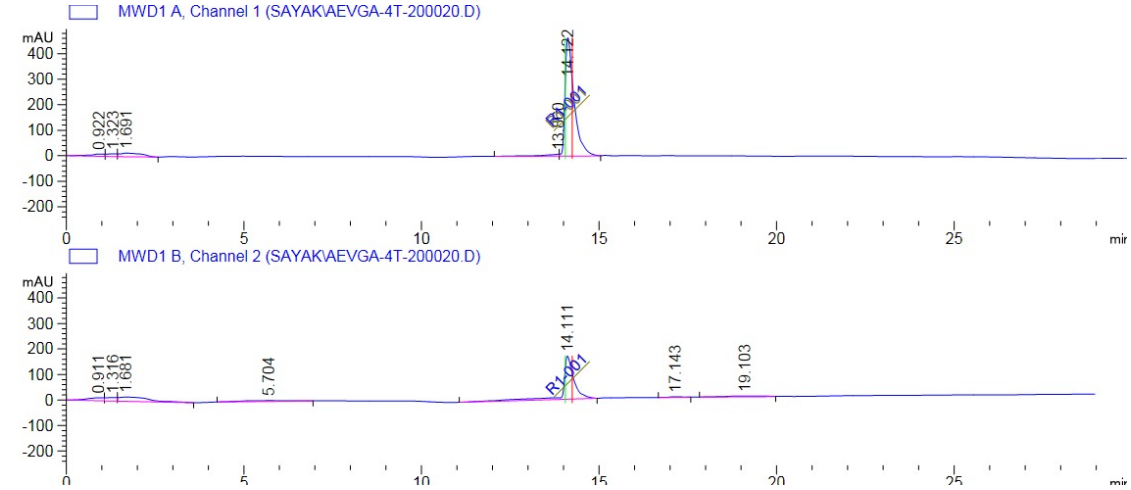

Figure S60: Analytical HPLC trace of HO-AEVGA-4T-AGVEA-OH monitoring 420 nm (top) and 260 nm (bottom).

References

(1) Abraham, M. J.; Murtola, T.; Schulz, R.; Páll, S.; Smith, J. C.; Hess, B.; Lindahl, E. GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. *SoftwareX* **2015**, *1*, 19–25.

(2) Lindorff-Larsen, K.; Piana, S.; Palmo, K.; Maragakis, P.; Klepeis, J. L.; Dror, R. O.; Shaw, D. E. Improved side-chain torsion potentials for the Amber ff99SB protein force field. *Proteins: Structure, Function, and Bioinformatics* **2010**, *78*, 1950–1958.

(3) Reynolds, C. A.; Essex, J. W.; Richards, W. G. Atomic charges for variable molecular conformations. *Journal of the American Chemical Society* **1992**, *114*, 9075–9079.

(4) Cornell, W. D.; Cieplak, P.; Bayly, C. I.; Kollman, P. A. Application of RESP charges to calculate conformational energies, hydrogen bond energies, and free energies of solvation. *Journal of the American Chemical Society* **2002**, *115*, 9620–9631.

(5) Frisch, M. J.; Trucks, G. W.; Schlegel, H. B.; Scuseria, G. E.; Robb, M. A.; Cheeseman, J. R.; Scalmani, G.; Barone, V.; Petersson, G. A.; Nakatsuji, H. et al. Gaussian16 Revision C.01. 2016; Gaussian Inc. Wallingford CT.

(6) Da Silva, A. W. S.; Vranken, W. F. ACPYPE-Antechamber python parser interface. *BMC Research Notes* **2012**, *5*, 1–8.

(7) Case, D.; Belfon, K.; Ben-Shalom, I.; Brozell, S.; Cerutti, D.; III, T. C.; Cruzeiro, V.; Darden, T.; Duke, R.; Giambasu, G. et al. AMBER 2020. 2020; University of California, San Francisco.

(8) Mark, P.; Nilsson, L. Structure and dynamics of the TIP3P, SPC, and SPC/E water models at 298 K. *The Journal of Physical Chemistry A* **2001**, *105*, 9954–9960.

(9) Bussi, G.; Donadio, D.; Parrinello, M. Canonical sampling through velocity rescaling. *The Journal of Chemical Physics* **2007**, *126*, 014101.

(10) Parrinello, M.; Rahman, A. Polymorphic transitions in single crystals: A new molecular dynamics method. *Journal of Applied Physics* **1981**, *52*, 7182–7190.

(11) Hockney, R. W.; Eastwood, J. W. *Computer Simulation Using Particles*; CRC Press, 1988.

(12) Hess, B.; Bekker, H.; Berendsen, H. J.; Fraaije, J. G. LINCS: A linear constraint solver for molecular simulations. *Journal of Computational Chemistry* **1997**, *18*, 1463–1472.

(13) Essman, U.; Perera, L.; Berkowitz, M.; Darden, T.; Lee, H.; Pedersen, L. A smooth particle mesh ewald potential. *J. Chem. Phys* **1995**, *103*, 8577–8592.

(14) Shmilovich, K.; Panda, S. S.; Stouffer, A.; Tovar, J. D.; Ferguson, A. L. Supporting data for: "Hybrid computational-experimental data-driven design of self-assembling π-conjugated peptides". 2021; `https://doi.org/10.5281/zenodo.5048397`.

(15) Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L. et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems* **2019**, *32*, 8026–8037.

(16) Fey, M.; Lenssen, J. E. Fast graph representation learning with PyTorch Geometric. *arXiv preprint arXiv:1903.02428* **2019**,

(17) Hagberg, A.; Swart, P.; S Chult, D. *Exploring network structure, dynamics, and function using NetworkX*; 2008.

(18) GPy, GPy: A Gaussian process framework in python. `http://github.com/SheffieldML/GPy`, since 2012.

(19) Harris, C. R.; Millman, K. J.; van der Walt, S. J.; Gommers, R.; Virtanen, P.; Cournapeau, D.; Wieser, E.; Taylor, J.; Berg, S.; Smith, N. J. et al. Array programming with NumPy. *Nature* **2020**, *585*, 357–362.

(20) Shmilovich, K.; Mansbach, R. A.; Sidky, H.; Dunne, O. E.; Panda, S. S.; Tovar, J. D.; Ferguson, A. L. Discovery of self-assembling π-conjugated peptides by active learning-directed coarse-grained molecular simulation. *The Journal of Physical Chemistry B* **2020**, *124*, 3873–3891.

(21) Wang, J.; Ferguson, A. L. Mesoscale simulation of asphaltene aggregation. *The Journal of Physical Chemistry B* **2016**, *120*, 8016–8035.

(22) Mansbach, R. A.; Ferguson, A. L. Coarse-grained molecular simulation of the hierarchical self-assembly of π-conjugated optoelectronic peptides. *Journal of Physical Chemistry B* **2017**, *121*, 1684–1706.

(23) Mansbach, R. A.; Ferguson, A. L. Control of the hierarchical assembly of π-conjugated optoelectronic peptides by pH and flow. *Organic and Biomolecular Chemistry* **2017**, *15*, 5484–5502.

(24) Mansbach, R. A.; Ferguson, A. L. Patchy particle model of the hierarchical self-assembly of π-conjugated optoelectronic peptides. *The Journal of Physical Chemistry B* **2018**, *122*, 10219–10236.

(25) McGibbon, R. T.; Beauchamp, K. A.; Harrigan, M. P.; Klein, C.; Swails, J. M.; Hernández, C. X.; Schwantes, C. R.; Wang, L.-P.; Lane, T. J.; Pande, V. S. MDTraj: A modern open library for the analysis of molecular dynamics trajectories. *Biophysical Journal* **2015**, *109*, 1528 – 1532.

(26) Pareto, V. *Cours d'économie politique*; Librairie Droz, 1964; Vol. 1.

(27) Kasha, M.; Rawls, H.; El-Bayoumi, M. A. The exciton model in molecular spectroscopy. *Pure and Applied Chemistry* **1965**, *11*, 371–392.

(28) Ardoña, H. A. M.; Besar, K.; Togninalli, M.; Katz, H. E.; Tovar, J. D. Sequence-

dependent mechanical, photophysical and electrical properties of pi-conjugated peptide hydrogelators. *Journal of Materials Chemistry C* **2015**, *3*, 6505–6514.

(29) Besar, K.; Ardona, H. A. M.; Tovar, J. D.; Katz, H. E. Demonstration of hole transport and voltage equilibration in self-assembled π-conjugated peptide nanostructures using field-effect transistor architectures. *ACS Nano* **2015**, *9*, 12401–12409.

(30) Ghosh, P.; Sajjadi, M. S.; Vergari, A.; Black, M.; Schölkopf, B. From variational to deterministic autoencoders. *arXiv preprint arXiv:1903.12436* **2019**,

(31) Kingma, D. P.; Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* **2013**,

(32) Shi, C.; Xu, M.; Guo, H.; Zhang, M.; Tang, J. A graph to graphs framework for retrosynthesis prediction. International Conference on Machine Learning. 2020; pp 8818–8827.

(33) Qian, W. W.; Russell, N. T.; Simons, C. L. W.; Luo, Y.; Burke, M. D.; Peng, J. Integrating deep neural networks and symbolic inference for organic reactivity prediction. *ChemRxiv* **2020**, DOI:10.26434/chemrxiv.11659563.v1.

(34) Mercado, R.; Rastemo, T.; Lindelöf, E.; Klambauer, G.; Engkvist, O.; Chen, H.; Bjerrum, E. J. Graph networks for molecular design. *Machine Learning: Science and Technology* **2021**, *2*, 025023.

(35) Battaglia, P. W.; Hamrick, J. B.; Bapst, V.; Sanchez-Gonzalez, A.; Zambaldi, V.; Malinowski, M.; Tacchetti, A.; Raposo, D.; Santoro, A.; Faulkner, R. et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261* **2018**,

(36) Kawashima, S.; Pokarowski, P.; Pokarowska, M.; Kolinski, A.; Katayama, T.; Kanehisa, M. AAindex: amino acid index database, progress report 2008. *Nucleic Acids Research* **2007**, *36*, D202–D205.

(37) Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; Dahl, G. E. Neural message passing for quantum chemistry. International Conference on Machine Learning. 2017; pp 1263–1272.

(38) Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* **2014**,

(39) Li, Y.; Tarlow, D.; Brockschmidt, M.; Zemel, R. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493* **2015**,

(40) Vinyals, O.; Bengio, S.; Kudlur, M. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391* **2015**,

(41) Winter, R.; Noé, F.; Clevert, D.-A. Permutation-Invariant Variational Autoencoder for Graph-Level Representation Learning. *arXiv preprint arXiv:2104.09856* **2021**,

(42) Simonovsky, M.; Komodakis, N. Graphvae: Towards generation of small graphs using variational autoencoders. International Conference on Artificial Neural Networks. 2018; pp 412–422.

(43) Kingma, D. P.; Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* **2014**,

(44) White, T. Sampling generative networks. *arXiv preprint arXiv:1609.04468* **2016**,

(45) Arthur, D.; Vassilvitskii, S. K-Means++: The Advantages of Careful Seeding. Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms. USA, 2007; pp 1027–1035.

(46) Brochu, E.; Cora, V. M.; De Freitas, N. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599* **2010**,

(47) Rasmussen, C. E.; Williams, C. K. I. *Gaussian processes for machine learning (adaptive computation and machine learning)*; The MIT Press, 2005.

(48) Paleyes, A.; Pullin, M.; Mahsereci, M.; Lawrence, N.; González, J. Emulation of physical processes with Emukit. Second Workshop on Machine Learning and the Physical Sciences, NeurIPS. 2019.

(49) Ebden, M. Gaussian processes: A quick introduction. *arXiv preprint arXiv:1505.02965* **2015**,

(50) Mohammadi, H.; Riche, R. L.; Durrande, N.; Touboul, E.; Bay, X. An analytic comparison of regularization methods for Gaussian processes. *arXiv preprint arXiv:1602.00853* **2016**,

(51) Perdikaris, P.; Raissi, M.; Damianou, A.; Lawrence, N. D.; Karniadakis, G. E. Nonlinear information fusion algorithms for data-efficient multi-fidelity modelling. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* **2017**, *473*, 20160751.

(52) Duvenaud, D. Automatic model construction with Gaussian processes. Ph.D. thesis, University of Cambridge, 2014.

(53) Waskom, M. L. seaborn: statistical data visualization. *Journal of Open Source Software* **2021**, *6*, 3021.

(54) Močkus, J. On Bayesian methods for seeking the extremum. Optimization techniques IFIP technical conference. 1975; pp 400–404.

(55) Jones, D. R.; Schonlau, M.; Welch, W. J. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization* **1998**, *13*, 455–492.

(56) Paria, B.; Kandasamy, K.; Póczos, B. A flexible framework for multi-objective Bayesian

optimization using random scalarizations. Proceedings of The 35th Uncertainty in Artificial Intelligence Conference. 2020; pp 766–776.

(57) Knowles, J. ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation* **2006**, *10*, 50–66.

(58) Thompson, W. R. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika* **1933**, *25*, 285–294.

(59) Auer, P. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research* **2002**, *3*, 397–422.

(60) Lizotte, D. J. Practical Bayesian optimization. Ph.D. thesis, 2008.

(61) Ginsbourger, D.; Le Riche, R.; Carraro, L. A multi-points criterion for deterministic parallel global optimization based on Gaussian processes. *HAL preprint hal-00260579* **2008**,

(62) Gómez-Bombarelli, R.; Aguilera-Iparraguirre, J.; Hirzel, T. D.; Duvenaud, D.; Maclaurin, D.; Blood-Forsythe, M. A.; Chae, H. S.; Einzinger, M.; Ha, D.-G.; Wu, T. et al. Design of efficient molecular organic light-emitting diodes by a high-throughput virtual screening and experimental approach. *Nature Materials* **2016**, *15*, 1120–1127.

(63) Coifman, R. R.; Lafon, S. Diffusion maps. *Applied and Computational Harmonic Analysis* **2006**, *21*, 5–30.

(64) Nadler, B.; Lafon, S.; Coifman, R. R.; Kevrekidis, I. G. Diffusion maps, spectral clustering and eigenfunctions of Fokker-Planck operators. *arXiv preprint math/0506090* **2005**,

(65) Long, A. W.; Ferguson, A. L. Rational design of patchy colloids via landscape engineering. *Molecular Systems Design & Engineering* **2018**, *3*, 49–65.

(66) Wang, J.; Ferguson, A. L. Nonlinear machine learning in simulations of soft and biological materials. *Molecular Simulation* **2018**, *44*, 1090–1107.

(67) Ma, Y.; Ferguson, A. L. Inverse design of self-assembling colloidal crystals with omnidirectional photonic bandgaps. *Soft Matter* **2019**, *15*, 8808–8826.

(68) Bartók, A. P.; Kondor, R.; Csányi, G. On representing chemical environments. *Physical Review B* **2013**, *87*, 184115.

(69) De, S.; Bartók, A. P.; Csányi, G.; Ceriotti, M. Comparing molecules and solids across structural and alchemical space. *Physical Chemistry Chemical Physics* **2016**, *18*, 13754–13769.

(70) Ceriotti, M.; Willatt, M. J.; Csányi, G. Machine learning of atomic-scale properties based on physical principles. *Handbook of Materials Modeling: Methods: Theory and Modeling* **2020**, 1911–1937.

(71) Homer, E. R.; Hensley, D. M.; Rosenbrock, C. W.; Nguyen, A. H.; Hart, G. L. Machine-learning informed representations for grain boundary structures. *Frontiers in Materials* **2019**, *6*, 168.

(72) Jäger, M. O.; Morooka, E. V.; Canova, F. F.; Himanen, L.; Foster, A. S. Machine learning hydrogen adsorption on nanoclusters through structural descriptors. *npj Computational Materials* **2018**, *4*, 1–8.

(73) Himanen, L.; Jäger, M. O.; Morooka, E. V.; Canova, F. F.; Ranawat, Y. S.; Gao, D. Z.; Rinke, P.; Foster, A. S. DScribe: Library of descriptors for machine learning in materials science. *Computer Physics Communications* **2020**, *247*, 106949.

(74) Wang, J.; Gayatri, M. A.; Ferguson, A. L. Mesoscale simulation and machine learning of asphaltene aggregation phase behavior and molecular assembly landscapes. *The Journal of Physical Chemistry B* **2017**, *121*, 4923–4944.

(75) Ferguson, A. L.; Panagiotopoulos, A. Z.; Debenedetti, P. G.; Kevrekidis, I. G. Systematic determination of order parameters for chain dynamics using diffusion maps. *Proceedings of the National Academy of Sciences* **2010**, *107*, 13597–13602.