Title

Konnektor: A Framework for Using Graph Theory to Plan Networks for Free Energy Calculations.

Journal

Journal of chemical information and computer sciences, 64(22)

Authors

Ries, Benjamin
Gowers, Richard
Baumann, Hannah
et al.

Publication Date

2024-11-25

DOI

10.1021/acs.jcim.4c01710

Peer reviewed

HHS Public Access

Author manuscript

J Chem Inf Model. Author manuscript; available in PMC 2025 February 11.

Published in final edited form as:

J Chem Inf Model. 2024 November 25; 64(22): 8396–8403. doi:10.1021/acs.jcim.4c01710.

Konnektor: A framework for using graph theory to plan networks for free energy calculations

Benjamin Ries[†,‡], **Richard J Gowers**[‡], **Hannah M Baumann**[‡], **David WH Swenson**[‡], **Michael M Henry**[‡,¶], **James RB Eastwood**[‡], **Irfan Alibay**[‡], **David Mobley**[§]

[†] Boehringer Ingelheim Pharma GmbH & Co KG, Medicinal Chemistry, Birkendorfer Str 65, 88397 Biberach an der Riss, Germany

[‡] Open Free Energy, Open Molecular Software Foundation, Davis, California 95616, United States

[¶] Computational and Systems Biology Program, Sloan Kettering Institute, Memorial Sloan Kettering Cancer Center, New York, NY, USA

[§] Departments of Pharmaceutical Sciences and Chemistry, University of California, Irvine, California 92617, United States

Abstract

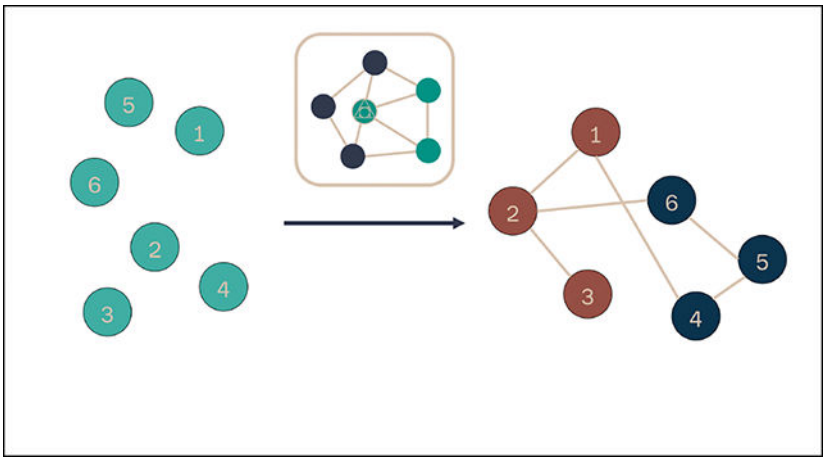
Alchemical free energy campaigns can be planned using graph theory by building up networks that contain nodes representing molecules that are connected by possible transformations as edges. We introduce Konnektor, an open-source Python package, for systematically planning, modifying, and analyzing free energy calculation networks. Konnektor is designed to aid in the drug discovery process by enabling users to easily setup free energy campaigns using complex graph manipulation methods.

The package contains functions for network operations including concatenation of networks, deletion of transformations, and clustering of molecules, along with a framework for combining these tools with existing network generation algorithms to enable the development of more complex methods for network generation. A comparison of the various network layout features offered is carried out using toy datasets. Additionally, Konnektor contains visualization and analysis tools, making the investigation of network features much simpler.

Besides the content of the package, the paper also offers application examples, demonstrating how Konnektor can be used and how the different networks perform from a graph theory perspective. Konnektor is freely available via GitHub at https://github.com/OpenFreeEnergy/konnektor under the permissive MIT License.

Graphical Abstract

benjamin.ries@boehringer-ingelheim.com .



Introduction

In silico binding free energy (FE) calculations using alchemical molecular dynamics (MD) simulations[1,2] are the current state-of-the-art in computer-aided drug design (CADD), yielding potency estimates that help drive key decisions such as the prioritization of synthesis and testing of candidates in drug discovery.[3,4] Free energy methods can help save materials and time, potentially leading to more sustainable and efficient drug discovery pipelines.[5] The vast number of successful applications and developments that have been described in the literature back up the use of FE calculations.[6–23]

Many different methods fall within the broader category of free energy calculations. Binding free energies between a protein and ligand are the paradigmatic example of the free energies that guide CADD efforts. Generally, binding free energy methods can be divided into relative binding free energies (RBFE) and absolute binding free energies (ABFE) methods (see Figure 1 A).[24] ABFE calculations directly yield the desired binding affinity of one molecule to a target $\Delta G_{\mathrm{A_{bind}}}$, but they are also significantly more expensive to calculate compared to the widely used RBFEs.[10,25–27] RBFE calculations yield the difference between two candidates in different environments ($\Delta G_{\mathrm{BA_{water}}}$ and $\Delta G_{\mathrm{BA_{complex}}}$) which then can be used to recover their binding free energy difference $\Delta\Delta G_{\mathrm{BA_{bind}}}$.[5,9,13,16,28,29] Both methods can be used to generate an FE network based on multiple $\Delta\Delta G_{\mathrm{BA_{bind}}}$ (see Figure 1 B). The FE network can then be translated into a ranking of all candidates, provided the candidates are connected by at least one relative comparison (see Figure 1 C).[30] The resulting ranking can be used, for example, as a priority list, helping to steer drug discovery efforts in the direction of improving ligand potency.

Despite these benefits, it is still challenging to run FE campaigns due to a lack of easily accessible automatized setup tools. The lack of such tools makes executing campaigns challenging for non-experts, and also inhibits development of larger-scale approaches to FE calculations.[4,31,32] Additionally, RBFE calculations, even though cheaper than ABFE calculations, are still notably time-consuming, often taking several hours even with state-of-the-art GPUs. This computational cost underlines the significance of well-planned FE

network calculation strategies, which can dramatically enhance the efficiency of building a candidate ranking.[24] Generally, there is a trade-off to be made between, on the one hand, a minimal number of required calculations ($\Delta\Delta G_{BA_{bind}}$s to be calculated) and, on the other, added redundancy in order to increase robustness and gain enhanced insights into the statistics of the final results via thermodynamic cycle analysis.[29,33]

Several approaches have been described to find good calculation strategies/networks. Some examples were directly borrowed from graph theory, such as the minimal spanning tree (MST) and star networks.[13,30,34] More sophisticated approaches have also been suggested. A widely used example being **LOMAP**[5,29,35] a rule-based algorithm developed around the requirements that, first, each node needs to be present in at least one cycle, and second, a network should require only a relatively small number of calculations connecting each ligand with each other. Recently **HIMAP**[36] was proposed, which is an algorithm that uses a density-based clustering step and an additional optimization phase for building up a network around these design concepts: an $n\ log(n)$ desired number of edges, and a certain number of graph cycles. These more sophisticated methods demonstrate the interest in approaches that plan free energy calculation networks, while optimizing multiple features of networks.

Here we present Konnektor, a package which assists users in formulating their FE calculation strategies through the provision of network generation tools. Konnektor is contained in the Open Free Energy (**OpenFE**)[37] environment, allowing for importing Konnektor via **openfe** and seamlessly integrating Konnektor code with OpenFE code. Konnektor can also be used as a standalone lightweight package.

In applications of FE calculations, desirable network characteristics maybe highly dependent on the features of the system being studied and on the FE calculation method. Konnektor's network tools give users the ability to develop new methods for network generation, or easily create bespoke networks to optimize for a specific criteria.

Implementation

Konnektor is an open-source Python 3[38] package under the MIT license. The source code can be retrieved from GitHub: https://github.com/OpenFreeEnergy/Konnektor. Alternatively, the package can be installed via conda-forge.[39]

The package is built into the OpenFE environment by making use of the reusable abstract base classes and tools under the **gufe** package (https://github.com/OpenFreeEnergy/gufe/). As the base, or translation, layer of OpenFE, **gufe** provides a common set of base types for representing components of alchemical simulations, allowing seamless development of new features and interaction with different packages in the **OpenFE** environment. For further information about the architecture of the OpenFE environment, see https://docs.openfree.energy/en/stable/cookbook/index.html.

The implementation of Konnektor relies on Python packages like **NetworkX**,[40] **NumPy**,[41] **scikit-learn**[42] and **scikit-mol**,[43] and **RDKit**[44] (a full list of dependencies can be found in the GitHub repository).

Package Content

Konnektor is structured into two functional compartments: network planners and tools for basic network operations, visualization, and analysis. The network planner module is the core module of Konnektor and allows the generation of a network from a set of molecules. The network tools enable further processing and investigation of transformation networks.

Problem statement

The theoretical model for Konnektor's algorithms is strongly influenced by graph theory. This approach allows us to represent for example each drug candidate as a node n in a graph G (see Figure 1 B). Next, each potential FE calculation comparing two molecules is represented by an edge e in G. The estimated difficulty of calculating a well converged and accurate $\Delta\Delta G_{\text{bind}}$ is represented by a score between 0 and 1 that is assigned to an e as a weight. By convention, 1 is a good score (expressing a likely accurate and well converged result) and 0 is a bad score. Algorithms for producing these estimates are outside the scope of this work, but are an active area of investigation in the field.[5,45] Finally, the task of planning a free energy calculation network to rank all candidates can be converted into a weighted graph construction problem, with the minimal requirement that the resulting graph must be connected.

Network Planning

The weighted graph construction problem can be solved as a Maximal Network (or fully connected graph, with each node connected to every other node) (see Figure 2-A). This approach provides a maximal number of edges N_e $(N_e = N_n * (N_n - 1))$,[46] which is a very inefficient approach for calculating a candidate ranking (see Figure 2). To build a comprehensive candidate ranking, only a connected graph is necessary, and from a thermodynamic standpoint (with perfect calculations leading to exact $\Delta\Delta G_{\text{bind}}$) no additional information is obtained by incorporating more edges into such a graph. Still, the Maximal Network approach is very important, acting as a precursor in network generation, with all subsequent network types pruning relevant edges to achieve their desired network layout. The generation of the Maximal Network is a time-critical step in the network planning phase, as all possible edges are generated and scored. Therefore, we implemented a parallelization scheme for this algorithm to speed up the calculations for larger numbers of molecules, which will affect all other network planning algorithms. However, for very large-scale approaches, this parallelization might not be sufficient.

Given the inefficiency of calculating the Maximal Networks, the purpose of Konnektor is to allow the user to explore different trade-offs between efficiency and other desirable properties. The approach we have used is to implement algorithms from graph theory to generate several distinct network topologies that may be of interest to the user, in addition to providing simple tools for modifying networks to allow more user control.

Reduced Network Topologies—The Heuristic Maximal Network is a special form of the Maximal Network (see Figure 2-B), allowing a drastic reduction of the cost of the network generation by picking randomly a subset of $N_e = N_e^n * N_n$ if $N_e^n \gg N_n$ (N_e^n with as

the number of edges per node, as default $N_e^n = 100$), leading to a highly connected network, instead of a maximally connected network.

Opposite to the Maximal Network, the most edge-efficient network layouts are the Star Network and the Minimal Spanning Tree (MST) Network (see Figure 2-C), for which Konnektor uses the Kruskal Algorithm,[34] with $N_e = N_n - 1$. From a network cost perspective, the MST Network is more efficient than the Star Network (see Figure 2-D), which has to find a compromise for the central molecule. The Star Network is more robust in retrieving a ranking of most molecules when some calculations fail. Calculation failures might occur from non-optimally chosen edge weights, very large molecule perturbations, or other unforeseen failures, like outages of computing nodes. Therefore, redundant edges can safeguard the network against calculation failures and the resulting disconnectivity. The Redundant MST Network (see Figure 2-F) executes the Kruskal Algorithm N_r times, always removing already selected edges from the input. Afterward, the overlay of all generated networks yields the Redundant MST Network with $N_e = N_r * (N_n - 1)$, with N_r: redundancy number, default: 3. The N-Edges Network (see Figure 2-G) builds a network with a MST basis followed by a greedy algorithm selecting the best scoring $N_r - 1$ edges per n leading to $N_e \leq N_r * (N_n - 1)$,with N_r: redundancy number, default: 3.

An additional benefit of redundancy is the possibility of spotting outliers and improving the overall estimates with thermodynamic cycles.[29,33] The Twin Star Network (see Figure 2-E) follows the design principle of building up a significant number of cycles by generating a special form of the Star Network. In the Konnektor implementation, the two nodes that perform best in average over all edges and edge weights are used as centers, building up the number of cycles of size three as $N_{c(3)} = n - 2$, leading to $N_e <= N_c * (N_n - 1)$, with N_c: minimal node cycle number. The Cyclic Network (see Figure 2-H) is generated by a greedy algorithm that builds up all possible cyclic paths around each node of a given size range (default: [3]), with the sum of the contained edge scores as path score. Next the N_c^x best score performing cyclic paths are selected for each node and merged into the final network. The number of generated cycles is more complex to determine as the edge cycles might overlap, therefore the number of cycles can be estimated as: $N_{c(3)} \leq n * N_c^x$.

Network Tools

In addition to network planning algorithms, Konnektor offers features for network operations, visualization, and analysis.

Network Operations—Network operations include common tasks like addition or removal of edges or nodes, provided by the following functions: The *merge_networks* function is the most efficient means of combining two networks into a new one, provided they share a node, by leaving the network edges unmodified. The *concatenate_networks* and *append_component* functions can be utilized when no node is shared, employing specific types of network planners solving a bipartite graph matching and returning a chosen number of edges in order to concatenate two networks. Removing elements from a network is possible using *delete_transformation* or *delete_component*.

Molecule Clustering—The *Clusterer* classes in Konnektor assist in separating the molecules based on their properties. In the following we will provide three examples.

Konnektor contains functionality for separating molecules by their fingerprint diversity using **scikit-mol** as a direct interface into **scikit-learn** with the *ComponentsDiversityClusterer*. This allows users to use a plethora of different fingerprint generators implemented in **scikit-mol** and clustering algorithms from **scikit-learn**.

The *ChargeClusterer* simply separates a set of molecules into groups by formal charge, while the *ScaffoldClusterer* clusters molecules by common structure scaffolds using the scaffold network module in **RDKit** by Kruger *et al.*.[47]

Network Visualization—The visualization module offers two methods for network visualization. The *draw_network* function produces a **Matplotlib**[48] figure of the network, which can be easily converted into an image file (.png) (see 3). Alternatively, an interactive visualization variant is available for use in **IPython**[49] environments using **IpyCytoscape**[50] in the function *draw_network_widget*.

Network Analysis—The Analysis module provides users with functions to determine graph properties of a network, like the *network_cost*, which gives the sum of all edge weights as a cost. More in-depth insights can be obtained with the *component_connectivity* functions, calculating how many edges are connected to each individual node, or the *component_score*, returning the sum of all edge weights the node is involved in. The *component_cycle* functionality calculates the number of cycles in which the node is present. The *robustness_measure* is a function that samples the graph to test if it remains connected after a certain proportion of edges are randomly removed (see Figure 2-1-6).

Advanced Network Algorithms

Finally, Konnektor allows for more advanced network layouts and contains an advanced network algorithm example, the Starry Sky Network, that combines multiple steps in order to build up a more complex network layout. The Starry Sky Network Planner (see Figure 2-I) was inspired by HIMAP,[36] combining the following steps with the tools from the section Network Tools:

1. HDBSCAN clustering[51] using Morgan fingerprints[52] as features for the molecules
2. Constructing *Star Network*s for each cluster.
3. Concatenating (explained below) the networks to one *Starry Sky Network* after adding the n *best performing* edges between the clusters.

The idea of this algorithm is to build an efficient network layout while optimizing the number of edges and graph score. In the clusters, similar molecules are expected to group, allowing the usage of low edge redundancy inside the clusters. Outside the clusters, additional redundant edges are used to increase the network robustness in case of calculation failure, ensuring graph connectivity.

Applications

In the following section, we demonstrate how Konnektor can be used for generating FE simulation campaigns.

Generating Networks

A very simple starting point is the generation of a Cyclic Network from a randomized dataset and visualizing the result (Figure 3). The input values are generated with Konnektor's randomized toy dataset functionality, leading to a random set of molecules, and additionally returns a specific mapper and scorer for these artificial molecules. Next, the generated input values are used to construct the network planner class for cyclic networks called *CyclicNetworkGenerator*. The network is generated by calling the *generate_ligand_network* function and passing the list of molecules. Finally, the network is visualized using the *draw_ligand_network* function (see Figure 3).

The presented example can be tested in the provided Jupyter Notebook on GitHub, alongside more advanced examples, a real-world case, and building a charge separating network, using a wider range of tools in Konnektor and the interactive widget visualization (see https://github.com/OpenFreeEnergy/konnektor/blob/main/examples/konnektor_example.ipynb).

Systematic Network Comparison

In the following systematic comparison of network layouts, we applied graph analysis tools (**1**-**6**) to different-sized random data sets arranged in each of the network layouts (**A**-**I**) defined above (see Figure 2). As a test range, we selected 10 molecule sets starting with sizes from 5 to 200 molecules. The data sets were generated three times randomly and each possible edge was assigned a weight from a random normal distribution.

Comparing Network Cost, Number of Edges, and Number of Cycles—Network cost aggregates all edge weights of a network, indicating the overall difficulty of all calculations finishing successfully in a network. The smaller the cost, the more likely a network and its *Transformation*s will converge well and yield reasonable FE estimates.

The MST Network shows the lowest network cost in this comparison (2-**1**.**C**), a result expected from graph theory. Next follows the Redundant MST and N-Node Edges Networks: both optimize the edge cost but use significantly more edges than the MST Network, increasing the required computational cost (2–1**,2**.**F**-**G**).

The Star Network and the Twin Star Network perform less well on network cost, as they use the arithmetic mean of cost for choosing the central molecule(2-**1**.**D**-**E**). However, the Star Network is the only approach that uses the same number of edges as the MST Network. For the Twin Star Network, a key feature is the number of generated cycles in the graph, which is rivaled only by the Cyclic Network (2-**2**.**E**,**H**).

The Cyclic Network shows an improved network cost over the Twin Star Network that is similar to the Star Network; however, it uses a similar number of edges to the Twin Star Network to generate cycles (2-**1**,**2**.**D**,**E**,**H**).

The Starry Sky network performs slightly better on network cost than the Star Network, since it can define multiple centers depending on the clustering, but uses slightly more edges (2-**2**.**D**,**I**). The network layout generates a notable number of cycles too, but the number of cycles and edges vary strongly depending on the dataset (2-**3**.**I**).

For completion, we present the Maximal Network and the Heuristic Maximal Network as extreme-case examples of a maximal number of Edges, Cycles, and network cost (2-**1**-**3**.**A,B**).

Comparing Network Robustness—Next, we tested graph robustness, measuring when the loss of an edge leads to a disconnected graph, since disconnection translates to the loss of molecules in a final ranking Random edge failure rates of 10% and 25% were used and repeated for each network 50 times (2-**4**,**5**).

As expected, the most efficient network layouts like the MST Network and the Star Network form disconnected graphs if even one edge is removed (2-**4**,**5**.**C**,**F**). For the MST Network case, it is hard to predict how many *Component*s would be lost in a final ranking, as the loss highly depends on where in the structure the *Transformation* fails. The Star Network will lose as many molecules as failing edges (a 10% loss of edges, leads to 10% loss of molecules in the final ranking). As a practical note, a good transformation score function is one under which higher network cost directly correlates with more edge failures in the network. This meaning of cost has not been considered in this toy system.

The Twin Star Network is more robust than the most edge efficient approaches but loses robustness quickly with an increased number of molecules: since the condition for disconnection is two edges being removed from a single molecule, the probability of this condition being met increases with the total number of edges removed (2-**4**,**5**.**E**). The N-Node Edges Network, the Redundant MST Network, and the Cyclic Network are more robust network approaches which rarely show disconnected graphs at an edge failure rate of 10% (2-**4**.**F-H**). However, the networks also start to break down with a 25% failure rate and increasing numbers of molecules(2-**5**.**F-H**). The two variants of the Maximal Network approaches were the most robust approaches in both failure rate cases (2-**4**,**5**.**A**,**B**). The Heuristic Maximal Network approach however gets disconnected with small numbers of molecules, which is expected as the method is only proposed for a very large number of molecules.

Network Generation Cost—The final parameter we tested was the time cost of Network generation (2-**6**). In our assessment, we focused on network planning and want to note, that with molecules from drug design projects and real-world edge generation methods, e.g. using LOMAP atom mapper and scorer,[5] the calculation time will be increased. Benchmarks were carried out using a single core of an AMD Ryzen 9 7900X CPU.

The Maximal Network approaches are the baseline of time consumption in our approaches, as each other network layout uses them as a basic solution that is filtered down to the desired network layout (2-**6**.**A**).

In the benchmark, the MST Network and Star Network are the fastest algorithms (2-**6**.**C**,**D**). The Twin Star-, Redundant MST, and N-Node Edges Networks are slightly slower, which might only matter for very large numbers of molecules (2-**6**.**E**-**G**). The slower algorithms are the Starry Sky and Cyclic Network generating algorithms, which show a clear increase in calculation time (2-**6**.**H**,**I**). Interestingly, the Starry Sky Network shows noisy generation cost averages. This observation is a dataset dependency caused by the number of clusters contained in the randomized toy dataset.

Conclusion

We have introduced Konnektor, a tool for systematically planning, modifying, and analyzing FE calculation networks. Konnektor enables users to plan their free energy calculation networks and to augment these networks to adjust the calculations according to the needs of drug design projects, like adding new molecule designs or handling multiple crystal structure dependent networks. Method developers can rapidly construct more complex network generation algorithms relying on the tools in Konnektor, which could address applications including large-scale networks with many ligands, networks involving mixed FE calculation methods, or even multistate FE methods. Our application examples show how Konnektor can be used and how the different networks perform based on graph analysis metrics. We hope that Konnektor will encourage a more creative application of graph theory to the problems of Free Energy calculations. We exhort practitioners to use the tools presented here to develop even more sophisticated network generation algorithms and unleash the Free Energy methods of a new era, freed from the constraints of star networks.

Acknowledgement

The authors thank Jenke Scheen, Enrico Ruijsenaars, Josh Horton, and David Dotson for fruitful discussions and testing the package, and Josh Mitchell for support with documentation. DLM appreciates financial support from the National Institutes of Health (R35GM148236).

Special thanks are extended to Aniket Magarkar for his support throughout the course of this study. We thank the partners in the Open Free Energy Fund, and Boehringer Ingelheim Pharma GmbH & Co. KG for funding.

Data and Software Availability

The full workflow code is available on GitHub https://github.com/OpenFreeEnergy/konnektor and installation is described in the *README.md* file. Used sub-packages can be installed with the provided conda *environment.yml* file as described in the *README.md* file.

References

(1). Kirkwood JG Statistical Mechanics of Fluid Mixtures. J. Chem. Phys. 1935, 3, 300–313.

(2). Zwanzig RW High-Temperature Equation of State by a Pertubation Method. I. Nonpolar Gases. J. Chem. Phys. 1954, 22, 1420–1426.

(3). Cournia Z; Allen BK; Beuming T; Pearlman DA; Radak BK; Sherman W Rigorous Free Energy Simulations in Virtual Screening. J. Chem. Inf. Model. 2020, 60, 4153–4169. [PubMed: 32539386]

(4). Chipot C Free Energy Methods for the Description of Molecular Processes. Annual Review of Biophysics 2023, 52, 113–138.

(5). Liu S; Wu Y; Lin T; Abel R; Redmann JP; Summa CM; Jaber VR; Lim NM; Mobley DL Lead Optimization Mapper: Automating Free Energy Calculations for Lead Optimization. J. Comput. Aided Mol. Des. 2013, 27, 755–770. [PubMed: 24072356]

(6). Christ CD; Mark AE; van Gunsteren WF Basic Ingredients of Free Energy Calculations: A Review. J. Comput. Chem. 2009, 31, 1569–1582.

(7). Riniker S; Christ CD; Hansen N; Mark AE; Nair PC; Van Gunsteren WF Comparison of Enveloping Distribution Sampling and Thermodynamic Integration to Calculate Binding Free Energies of Phenylethanolamine N-Methyltransferase Inhibitors. J. Chem. Phys. 2011, 135, 24105.

(8). Wang L; Wu Y; Deng Y; Kim B; Pierce L; Krilov G; Lupyan D; Robinson S; Dahlgren MK; Greenwood J; Romero DL; Masse C; Knight JL; Steinbrecher T; Beuming T; Damm W; Harder E; Sherman W; Brewer M; Wester R; Murcko M; Frye L; Farid R; Lin T; Mobley DL; Jorgensen WL; Berne BJ; Friesner RA; Abel R Accurate and Reliable Prediction of Relative Ligand Binding Potency in Prospective Drug Discovery by Way of a Modern Free-Energy Calculation Protocol and Force Field. J. Am. Chem. Soc. 2015, 137, 2695–2703. [PubMed: 25625324]

(9). Wang L; Deng Y; Wu Y; Kim B; LeBard DN; Wandschneider D; Beachy M; Friesner RA; Abel R Accurate Modeling of Scaffold Hopping Transformations in Drug Discovery. J. Chem. Theory Comput. 2017, 13, 42–54. [PubMed: 27933808]

(10). Aldeghi M; Heifetz A; Bodkin MJ; Knapp S; Biggin PC Accurate Calculation of the Absolute Free Energy of Binding for Drug Molecules. Chem. Sci. 2016, 7, 207–218. [PubMed: 26798447]

(11). Sidler D; Schwaninger A; Riniker S Replica Exchange Enveloping Distribution Sampling (RE-EDS): A Robust Method to Estimate Multiple Free-Energy Differences From a Single Simulation. J. Chem. Phys. 2016, 145, 154114. [PubMed: 27782485]

(12). Yu HS; Deng Y; Wu Y; Sindhikara D; Rask AR; Kimura T; Abel R; Wang L Accurate and Reliable Prediction of the Binding Affinities of Macrocycles to Their Protein Targets. J. Chem. Theory Comput. 2017, 13, 6290–6300. [PubMed: 29120625]

(13). Jespers W; Esguerra M; Åqvist J; Gutiérrez-De-Terán H QligFEP: An Automated Workflow for Small Molecule Free Energy Calculations in Q. J. Cheminf. 2019, 11, 26.

(14). Wei J; Chipot C; Roux B Computing Relative Binding Affinity of Ligands to Receptor: An Effective Hybrid Single-Dual-Topology Free-Energy Perturbation Approach in NAMD. J. Chem. Inf. Model. 2019, 59, 3794–3802. [PubMed: 31411473]

(15). Paulsen JL; Yu HS; Sindhikara D; Wang L; Appleby T; Villasenor AG; Schmitz U; Shivakumar D Evaluation of Free Energy Calculations for the Prioritization of Macrocycle Synthesis. J. Chem. Inf. Model. 2020, 60, 3489–3498. [PubMed: 32539379]

(16). Meier K; Bluck JP; Christ CD Use of Free Energy Methods in the Drug Discovery Industry; ACS Symp. Ser.; J. Am. Chem. Soc, 2021; Vol. 1397; pp 39–66.

(17). Tielker N; Eberlein L; Beckstein O; Güssregen S; Iorga BI; Kast SM; Liu S Free Energy Methods in Drug Discovery: Current State and Future Directions; 2021; Chapter 3, pp 67–107.

(18). Heinzelmann G; Gilson M Automation of Absolute Protein-Ligand Binding Free Energy Calculations for Docking Refinement and Compound Evaluation. Sci. Rep. 2021, 11, 1116. [PubMed: 33441879]

(19). Gapsys V; Pérez-Benito L; Aldeghi M; Seeliger D; van Vlijmen H; Tresadern G; de Groot BL Large Scale Relative Protein Ligand Binding Affinities Using Non-Equilibrium Alchemy. Chem. Sci. 2020, 11, 1140–1152.

(20). Armacost KA; Riniker S; Cournia Z Novel Directions in Free Energy Methods and Applications. J. Chem. Inf. Model. 2020, 60, 1–5. [PubMed: 31983210]

(21). Sun S; Fushimi M; Rossetti T; Kaur N; Ferreira J; Miller M; Quast J; van den Heuvel J; Steegborn C; Levin LR; Buck J; Myers RW; Kargman S; Liverton N; Meinke PT; Huggins DJ Scaffold Hopping and Optimization of Small Molecule Soluble Adenyl Cyclase Inhibitors Led by Free Energy Perturbation. Journal of Chemical Information and Modeling 2023, 63, 2828–2841. [PubMed: 37060320]

(22). Chen W; Cui D; Jerome SV; Michino M; Lenselink EB; Huggins DJ; Beautrait A; Vendome J; Abel R; Friesner RA; Wang L Enhancing Hit Discovery in Virtual Screening through

Absolute Protein–Ligand Binding Free-Energy Calculations. Journal of Chemical Information and Modeling 2023, 63, 3171–3185. [PubMed: 37167486]

(23). Sampson JM; Cannon DA; Duan J; Epstein JC; Sergeeva AP; Katsamba PS; Mannepalli SM; Bahna FA; Adihou H; Guéret SM; Gopalakrishnan R; Geschwindner S; Rees DG; Sigurdardottir A; Wilkinson T; Dodd RB; De Maria L; Mobarec JC; Shapiro L; Honig B; Buchanan A; Friesner RA; Wang L Robust Prediction of Relative Binding Energies for Protein–Protein Complex Mutations Using Free Energy Perturbation Calculations. Journal of Molecular Biology 2024, 436, 168640. [PubMed: 38844044]

(24). Cournia Z; Allen B; Sherman W Relative Binding Free Energy Calculations in Drug Discovery: Recent Advances and Practical Considerations. J. Chem. Inf. Model. 2017, 57, 2911–2937. [PubMed: 29243483]

(25). Boresch S; Tettinger F; Leitgeb M; Karplus M Absolute Binding Free Energies: A Quantitative Approach for their Calculation. J. Phys. Chem. B 2003, 107, 9535–9551.

(26). Alibay I; Magarkar A; Seeliger D; Biggin PC Evaluating the Use of Absolute Binding Free Energy in the Fragment Optimisation Process. Commun. Chem. 2022, 5, 105. [PubMed: 36697714]

(27). Fu H; Zhou Y; Jing X; Shao X; Cai W Meta-Analysis Reveals That Absolute Binding Free-Energy Calculations Approach Chemical Accuracy. J. Med. Chem. 2022, 65, 12970–12978. [PubMed: 36179112]

(28). Jorgensen WL; Buckner JK; Boudon S; Tirado-Rives J Efficient Computation of Absolute Free Energies of Binding by Computer Simulations. Application to the Methane Dimer in Water. J. Chem. Phys. 1988, 89, 3742–3746.

(29). Wang L; Wu Y; Deng Y; Kim B; Pierce L; Krilov G; Lupyan D; Robinson S; Dahlgren MK; Greenwood J; Romero DL; Masse C; Knight JL; Steinbrecher T; Beuming T; Damm W; Harder E; Sherman W; Brewer M; Wester R; Murcko M; Frye L; Farid R; Lin T; Mobley DL; Jorgensen WL; Berne BJ; Friesner RA; Abel R Accurate and Reliable Prediction of Relative Ligand Binding Potency in Prospective Drug Discovery by Way of a Modern Free-Energy Calculation Protocol and Force Field. J. Am. Chem. Soc. 2015, 137, 2695–2703. [PubMed: 25625324]

(30). Yang Q; Burchett W; Steeno GS; Liu S; Yang M; Mobley DL; Hou X Optimal Designs for Pairwise Calculation: An Application to Free Energy Perturbation in Minimizing Prediction Variability. J. Comput. Chem. 2020, 41, 247–257. [PubMed: 31721260]

(31). Loeffler HH; Michel J; Woods C FESetup: Automating Setup for Alchemical Free Energy Simulations. J. Chem. Inf. Model 2015, 55, 2485–2490. [PubMed: 26544598]

(32). Zavitsanou S; Tsengenes A; Papadourakis M; Amendola G; Chatzigoulas A; Dellis D; Cosconati S; Cournia Z FEPrepare: A Web-Based Tool for Automating the Setup of Relative Binding Free Energy Calculations. J. Chem. Inf. Model. 2021, 61, 4131–4138. [PubMed: 34519200]

(33). Wang L; Deng Y; Knight JL; Wu Y; Kim B; Sherman W; Shelley JC; Lin T; Abel R Modeling Local Structural Rearrangements Using FEP/REST: Application to Relative Binding Affinity Predictions of CDK2 Inhibitors. Journal of Chemical Theory and Computation 2013, 9, 1282–1293. [PubMed: 26588769]

(34). Kruskal JB On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. Proc. Amer. Math. Soc 1956, 7, 48–48.

(35). Scheen J; Mackey M; Michel J Data-driven generation of perturbation networks for relative binding free energy calculations. Digital Discovery 2022, 1, 870–885.

(36). Pitman M; Hahn DF; Tresadern G; Mobley DL To Design Scalable Free Energy Perturbation Networks, Optimal Is Not Enough. J. Chem. Inf. Model 2023, 63, 1776–1793. [PubMed: 36878475]

(37). Gowers RJ; Alibay I; Swenson DW; Henry MM; Ries B; Baumann HM; Eastwood JRB The Open Free Energy library. 2023; 10.5281/zenodo.8344248, If you use this software, please cite it as below.

(38). van Rossum G Python Tutorial; Centrum voor Wiskunde en Informatica (CWI): Amsterdam, 1995.

(39). conda-forge community The conda-forge Project: Community-Based Software Distribution Built on the conda Package Format and Ecosystem. 2015; 10.5281/zenodo.4774216.

(40). Hagberg AA; Schult DA; Swart PJ Exploring Network Structure, Dynamics, and Function using NetworkX. Proceedings of the 7th Python in Science Conference. Pasadena, CA USA, 2008; pp 11–15.

(41). van Der Walt S; Colbert SC; Varoquaux G The NumPy Array: A Structure for Efficient Numerical Computation. Comput. Sci. Eng. 2011, 13, 22–30.

(42). Pedregosa F; Varoquaux G; Gramfort A; Michel V; Thirion B; Grisel O; Blondel M; Prettenhofer P; Weiss R; Dubourg V; Vanderplas J; Passos A; Cournapeau D; Brucher M; Perrot M; Duchesnay E Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research 2011, 12, 2825–2830.

(43). Bjerrum EJ; Bachorz RA; Bitton A; Choung O.-h.; Chen Y; Esposito C; Ha SV; Poehlmann A Scikit-Mol brings cheminformatics to Scikit-Learn. ChemRxiv 2023, preprint.

(44). Landrum G; Tosco P; Kelley B; Riniker S; Ric gedeck; Vianello R; Schneider N; Dalke A; N D; Eisuke K; Cole B; Turk S; Swain M; Alexander S; Cosgrove D; Vaucher A; Wójcikowski M; Jones G; Probst D; Godin G; Scalfani VF; Pahl A; Francois B; JLVarjo strets123; DoliathGavin JP; Sforna G; Jensen JH rdkit/rdkit: 2021_03_2 (Q1 2021) Release. 2023.

(45). Scheen J; Wu W; Mey ASJS; Tosco P; Mackey M; Michel J Hybrid Alchemical Free Energy/ Machine-Learning Methodology for the Computation of Hydration Free Energies. J. Chem. Inf. Model 2020, 60, 5331–5339. [PubMed: 32639733]

(46). Cormen TH; Leiserson CE; Rivest RL; Stein C Introduction to algorithms; MIT press, 2022.

(47). Kruger F; Stiefl N; Landrum GA rdScaffoldNetwork: The Scaffold Network Implementation in RDKit. Journal of Chemical Information and Modeling 2020, 60, 3331–3335. [PubMed: 32584031]

(48). Hunter JD Matplotlib: A 2D graphics environment. Computing in Science & Engineering 2007, 9, 90–95.

(49). Pérez F; Granger BE IPython: a System for Interactive Scientific Computing. Computing in Science and Engineering 2007, 9, 21–29.

(50). Shannon P; Markiel A; Ozier O; Baliga NS; Wang JT; Ramage D; Amin N; Schwikowski B; Ideker T Cytoscape: a software environment for integrated models of biomolecular interaction networks. Genome research 2003, 13, 2498–2504. [PubMed: 14597658]

(51). Campello RJGB; Moulavi D; Zimek A; Sander J Hierarchical Density Estimates for Data Clustering, Visualization, and Outlier Detection. ACM Trans. Knowl. Discov. Data 2015, 10.

(52). Morgan HL The Generation of a Unique Machine Description for Chemical Structures-a Technique Developed at Chemical Abstracts Service. J. chem. doc. 1965, 5, 107–113.

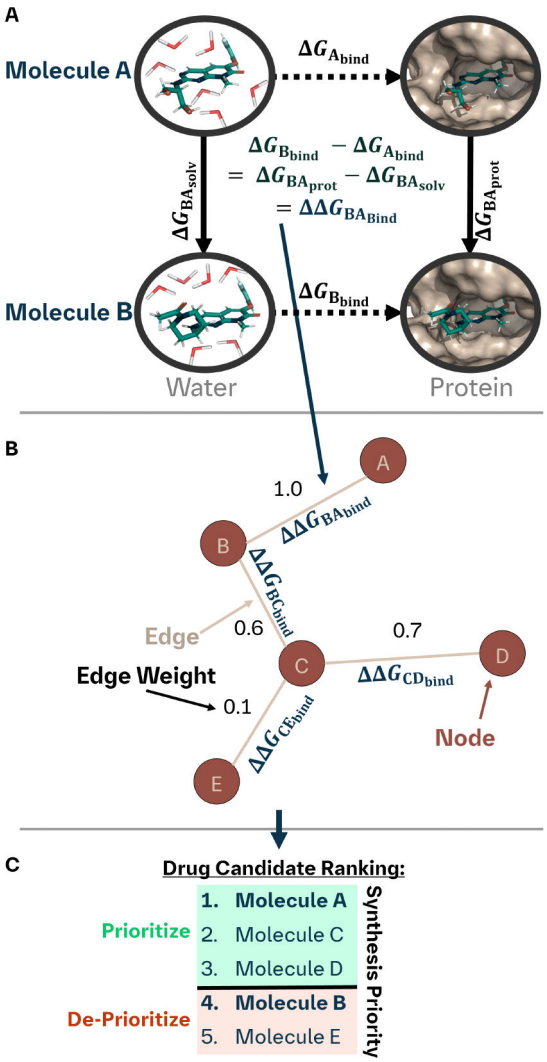


Figure 1:

The thermodynamic cycle describes how to calculate a comparison of the binding affinities of two molecules $\Delta\Delta G_{BA_{bind}}$ (**A**). These comparisons can be used to build up a FE network of thermodynamic cycles or edges. The difficulty to calculate one FE calculation can be described as an edge weight prior to calculating the $\Delta\Delta G_{BA_{bind}}$ (**B**). The edge weights can be used to build up efficient connected networks to calculate a set of $\Delta\Delta G_{BA_{bind}}$ that can be translated into a ranking. The final ranking can be used to support drug discovery, in order to find the most promising candidates due to their binding affinity, helping to prioritize synthesis (**C**).

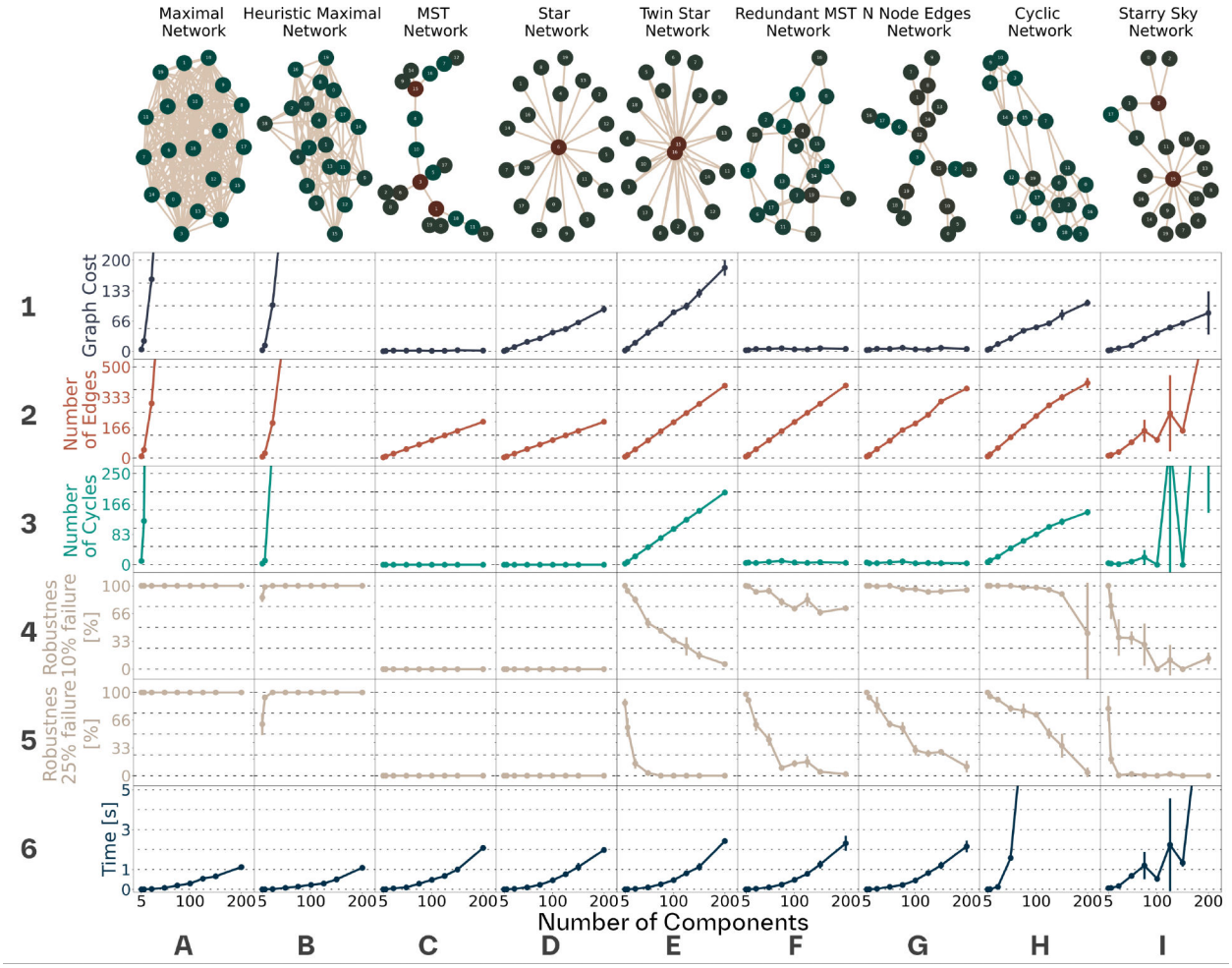


Figure 2:

Comparison of network layouts (**A-I**) using randomized toy datasets and graph analysis functions (**1-6**).

Code:

```
# Here we generate some random input data
from konnektor.utils import toy_data
components, mapper, scorer = toy_data.build_random_dataset(n_compounds=20)

# Generate network
from konnektor.network_planners import CyclicNetworkGenerator
networker = CyclicNetworkGenerator(mapper=mapper, scorer=scorer)
network = networker.generate_ligand_network(components)

# Visualize the generated network
from konnektor.visualization import draw_ligand_network
fig = draw_ligand_network(network=network, title="Cyclic Network")
```

Output:

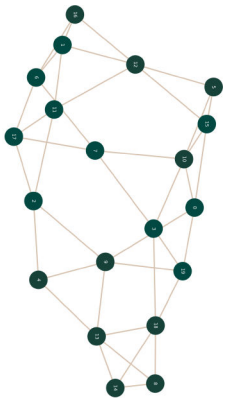


Figure 3:

Network visualization of the output of the Cyclic Network generation code example. Each node represents a molecule and each edge is a potential FE calculation