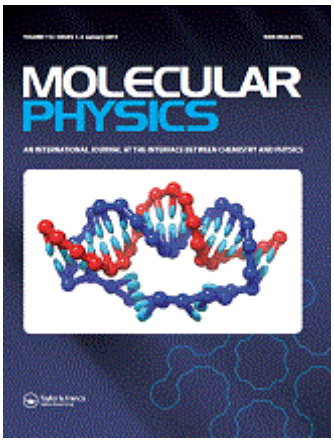


Subsets of adjacent nodes (SOAN): A fast method for computing suboptimal paths in protein dynamic networks

Journal:	*Molecular Physics*
Manuscript ID	Draft
Manuscript Type:	Invited Article
Date Submitted by the Author:	n/a
Complete List of Authors:	Dodd, Thomas ; Georgia State University, Department of Chemistry Yao, Xin-Qiu; Georgia State University, Department of Chemistry Hamelberg, Donald ; Georgia State University, Department of Chemistry Ivanov, Ivaylo; Georgia State University, Department of Chemistry
Keywords:	molecular dynamics, dynamic network analysis, suboptimal paths, graph theory, allosteric communication and regulation

Allosteric regulation is a key functional feature of many proteins and protein complexes, involving communication between distal protein regions. The process is initiated by ligand binding or other structural or dynamic perturbation, which occurs at one site and is subsequently propagated through the protein to influence the activities at a distal site. We present a new method for rapidly computing suboptimal paths between defined sites in protein dynamic networks. The method allows identification of allosteric residue paths in large proteins and macromolecular assemblies. Knowledge of allosteric communication mechanisms has impact on the fields of rational drug discovery and protein design.

Subsets of adjacent nodes (SOAN): A fast method for computing suboptimal paths in protein dynamic networks

Thomas Dodd[1,2†], Xin-Qiu Yao[1,2†], Donald Hamelberg[1,2*] and Ivaylo Ivanov[1,2*]

1. Department of Chemistry, Georgia State University, Atlanta, Georgia, USA.

2. Center for Diagnostics and Therapeutics, Georgia State University, Atlanta, Georgia, USA.

† Equal contribution

Corresponding authors:

Professor Ivaylo Ivanov

Email: iivanov@gsu.edu

Tel: +1 404 413 5529

Professor Donald Hamelberg

Email: dhamelberg@gsu.edu

Tel: +1 404 413 5564

Keywords: molecular dynamics, allosteric communication and regulation, dynamic network analysis, suboptimal paths, graph theory

Abstract

Suboptimal path analysis in a protein structural or dynamical network becomes increasingly popular for identifying critical residues involved in allosteric communication and regulation. Several software packages have been developed for calculating suboptimal paths, including NetworkView, WISP, and CNAPATH (Bio3D). Although these packages work well for biological systems of moderate sizes, they either dramatically slow down or are subjected to accuracy issues when applied to large systems such as supramolecular complexes. In this work, we develop a new method called SOAN, which implements a modified version of Yen's algorithm for finding loopless *k*-shortest paths. Instead of searching the entire protein network, SOAN builds up a subgraph for path calculations based on an initial evaluation of the optimal path and its neighboring nodes. We test our method on four systems of increasing size and compare to the NetworkView, WISP, and CNAPATH methods. The result shows that SOAN is approximately five times faster than NetworkView and orders of magnitude faster than CNAPATH and WISP. In terms of accuracy, SOAN is comparable to CNAPATH and WISP and superior to NetworkView. We also discuss the influence of SOAN input parameters on performance and suggest optimal values.

Introduction

Allosteric regulation is a key functional feature of many proteins and protein complexes, involving communication between distal protein regions. The process is initiated by ligand binding or some other structural or dynamic perturbation, which occurs at one site and is subsequently propagated through the protein to influence the activities at a distal site. Knowledge of allosteric communication mechanisms has impact on the fields of rational drug discovery[1] and protein design[2]. While classical models of allosteric regulation have suggested that a binding event induces substantial conformational changes in the distal site[3,4], other studies have observed allostery in the absence of large-scale conformational change[5,6]. This suggests that subtle differences in dynamics can alter the population distribution of the conformational ensemble without drastically altering the average conformation of the biomolecule.

In recent years, numerous computational methods have been developed to elucidate this subtle form of allosteric communication. Many of these methods represent the protein topology as a graph and use network algorithms to provide a link between the connectivity of protein residues and their functional significance in the regulatory allosteric response. In the graph, nodes (or

vertices) represent the protein residues. Edges between the nodes are weighted according to geometric[7-11,14-16] or energetic criteria[12,13]. An underlying assumption is that allosteric coupling can be described as the propagation of information facilitated by the network of interacting residues. One such method, dynamic network analysis[11], incorporates dynamic information from molecular dynamics (MD) simulations and has found exceptionally wide applicability to many biological systems[11,17-21]. Specifically, residue-residue cross-correlations are computed from the MD trajectories and mapped onto the edges of the network. Using edge weights based on dynamic correlation has advantages compared to static contact networks, accounting for dynamic conformational rearrangements occurring during MD. Once constructed, these dynamic networks encapsulate a wealth of information on the hierarchical organization and communication pathways of a biomolecule in a particular functional state. Conveniently, these important network features can be readily computed using methods derived from graph theory[22-24]. One such feature is suboptimal paths (or shortest paths) between a source and target residue (e.g. residues located on two distinct sites in a protein). Dynamic network models have traditionally focused on the shortest path (or optimal path), presumed to be the most significant contributor to protein allostery. However, suboptimal paths, slightly longer than the optimal, have also been shown to have outsized influence on allosteric responses. Determining these paths and their statistical distribution through the nodes of the protein network could help identify key functional residues involved in allosteric regulation.

Recent advances in computer architecture have made the simulation of large macromolecular complexes computationally feasible. However, conventional suboptimal paths analysis may be ill-suited to handle networks derived from large systems since the runtime is expected to increase significantly with the increased number of edges. In this work, we propose a neighbors-based approach for rapidly computing suboptimal paths between two protein residues in a dynamic network. Our protocol is written in the Python programming language and takes advantage of well-established Python packages, including NumPy and NetworkX[25-29]. To establish the utility and computational efficiency of our approach, we provide benchmark calculation on four biological systems of increasing size: (i) acid-β-glucosidase (GlcCerase) comprised of ~ 500 residues, (ii) the *E.coli* replication machinery DNA polymerase III (Pol III) comprised of ~ 2000 residues, (iii) the human transcription factor IIH (TFIIH) comprised of ~ 4000 residues and (iv) the human pre-initiation complex (PIC) comprised of ~ 10000 residues. Additionally, we compare the computational efficiency and accuracy of our approach with widely used network analysis methods: CNAPATH in the Bio3D R package[30-32], the Weighted Implementation of Suboptimal Paths (WISP)[33], and NetworkView[24].

Methods

Building the network. Prior to computing suboptimal paths using the neighbors-based approach the adjacency matrix *C* must be constructed. Embedded within the adjacency matrix is the connectivity of graph *G=(V,E)*, where *G* is an undirected graph containing vertices (or nodes) *V* and edges *E*. Note that *C* is a *n x n* symmetric matrix whose rank equals the total number of vertices in the graph (*n*). For weighted graphs, the value of the matrix element C_{ij} is,

$$C_{ij} = \begin{cases} w_{E_{ij}}, & if\ E_{ij} \in G \\ 0, & else \end{cases}. \quad (1)$$

where w_{Eij} is the weight of the edge connecting nodes *i* and *j*. Our protocol has been designed to work with adjacency matrices that have been derived in a manner consistent with dynamic network analysis[11]. As discussed previously, the dynamical network model represents each Cα atom in the protein as a node, although other choices are possible. Edges between nodes are then determined and weighted based on the interdependence among protein residues. In the case of MD, an edge is drawn between two nodes if the minimal (non-hydrogen) atomic distance between the two corresponding residues is ≤ 4.5 Å for more than 75% of the MD trajectory. Additionally, residues adjacent to the node under consideration and directly connected through chemical bonds are ignored (i.e. residues $(i \pm 1)$), as these are expected to be within the distance cutoff and highly correlated. In some network models, this definition of adjacency is further extended to include second or third neighbors in the amino acid sequence (e.g. $i \pm 2$ or $i \pm 3$). The weight of an edge is determined by the cross-correlation, c_{ij}, between the two corresponding residues,

$$c_{ij} = \frac{\langle (\mathbf{r}_i - \langle \mathbf{r}_i \rangle) \cdot (\mathbf{r}_j - \langle \mathbf{r}_j \rangle) \rangle}{\langle \|\mathbf{r}_i - \langle \mathbf{r}_i \rangle\|^2 \rangle^{1/2} \langle \|\mathbf{r}_j - \langle \mathbf{r}_j \rangle\|^2 \rangle^{1/2}}. \quad (2)$$

Here, <·> is the ensemble average (estimated from MD simulations) and r_i (r_j) is the positional vector of node *i* (*j*). Prior to the calculation, all simulation frames are structurally superimposed onto a reference structure (e.g., the first frame) based on the backbone or Cα atoms. There are numerous software packages available that can compute these values directly from MD trajectories, including Carma[34], CPPTRAJ[35], WISP[33], and Bio3D[30-32], to name a few. In order to examine the shortest paths in protein networks, the elements of the cross-correlation matrix *C* are transformed to distances using,

$$w_{ij} \equiv d_{ij} = -\log(|c_{ij}|). \quad (3)$$

Note that d_{ij} in equation 3 is not a physical distance (e.g. Cartesian-based) but rather a distance metric in correlation space. Moreover, taking the negative logarithm of the cross-correlation ensures that strongly correlated and anticorrelated residues will correspond to shorter distances d_{ij}.

Employing neighbors to identify an initial set of suboptimal paths. Upon construction of the *C* matrix, the first step in our protocol is to find the single, most direct route (the optimal path) in the network between a source (*s)* and target (*t)* node. This is readily accomplished using Dijkstra's algorithm. However, we are interested in paths connecting *s* and *t* that are slightly longer than the optimal path. This general definition of suboptimal paths, based on length, allows us to make an important assumption: if the suboptimal paths are only slightly longer than the optimal path, then it follows that at some point these paths will diverge from the optimal path via a node that is directly adjacent to the optimal pathway. In other words, the neighbors of the optimal path will be the most likely candidates for suboptimal pathway composition. To clarify, a node *j* is considered a neighbor of node *i* if there is an edge E_{ij} between them. Thus, the terms neighbor and adjacent are used interchangeably.

For each node comprising the optimal path we identify all nodes that are directly adjacent. The shortest paths are then computed, again using Dijkstra's algorithm, from $s \rightarrow n_j$ and $n_j \rightarrow t$, where n_j is the neighbor. These paths are then stitched together at the n_j junction. Any path that folds back on itself (i.e. $s,\ldots,n_{j-1},n_j,n_{j-1},\ldots,t$) is discarded as these do not constitute a direct route to the target. By default, our code expands away from the optimal path by 1 neighbor. However, this is a free parameter that permits any level of expansion up to including the entire graph. Thus, one can, in principle, include the neighbors of the optimal path neighbors and so on. This would, however, increase the run time of the code.

Improving path statistics using a subgraph derived from the initial set of suboptimal paths. The previous step represents a first pass through the graph since the paths produced are highly unique. This is a consequence of Dijkstra's algorithm, which will only identify multiple paths between two nodes if the path lengths are equal. Unfortunately, a set of unique paths will provide very poor path statistics, making the computation of statistical quantities like node degeneracies impractical. Node degeneracy is particularly important, as it aids in the identification of critical node residues in the allosteric network. A natural remedy, then, is to utilize the node composition of the paths found in the previous step to limit the search space and apply more robust path exploration techniques to improve the statistics. Hence, we can use the initial set of shortest paths and the nodes that comprise them to build a subgraph *S* of the original graph *G*. Let $P=\{p_1,\ldots,p_N\}$

be the set of shortest paths between *s* and *t* found previously. For each p_i in *P* there is a set of nodes that is unique to p_i but collectively belong to the graph *G*. These nodes form the basis of the subgraph *S* and retain the connectivity to each other defined by *G*. In short, the subgraph *S* is simply a subset of the original graph *G*, bound by the nodes that make up the initial set of shortest path *P*,

$$S = (V, E)_{V|E \in P \text{ and } G}. \quad (4)$$

The benefits of this approach are two-fold: (i) the complexity of the search space is significantly reduced, thus decreasing the overall run time and (ii) a search space bound by an initial set of shortest paths virtually guarantees that every new path found is, by definition, suboptimal.

From the subgraph *S* we can now find the shortest paths between *s* and *t* by employing the *k*-shortest paths algorithm proposed by Yen[36]. This algorithm computes the single-source *k*-shortest loopless paths for a graph with non-negative weights. Upon finding the shortest path, the *k*-shortest paths algorithm then proceeds to find the next shortest path and so forth, up to *k*, which is a free parameter (optimal values selection for free parameters is covered under Discussion). Since we already have an initial set of suboptimal paths found in the previous steps, our code discards any paths identified with *k*-shortest paths that overlap with this set, thus avoiding redundancy. If it is not possible to find *k*-shortest paths within the subgraph, our code will then expand to the next level of neighbors, rebuild the subgraph and continue Yen's algorithm until *k* is reached.

An overview of the method can be seen in Figure 1. In general, our methodology proceeds through the following steps:

1. Compute the optimal path between *s* and *t* using Dijkstra's algorithm.
2. Identify all nodes adjacent to the optimal path computed in step 1.
3. Compute all shortest paths between $s \rightarrow n_i$ and $n_i \rightarrow t$ using Dijkstra's algorithm.
4. Combine paths from step 3 at the n_i junction.
5. Identify all unique nodes from the initial set of shortest paths computed in steps 3 and 4.
6. Build a subgraph *S* of the original graph *G* using the nodes identified in step 5 and the interconnectivity defined by *G*.
7. Find *k* shortest paths between *s* and *t* in the subgraph *S* using the *k*-shortest paths algorithm by Yen.
8. If the number of specified paths (*k*) is found within the subgraph, output the results; else, return to step 2 and expand to the next level of neighbors.

Molecular dynamics simulations. Modeling, system setup and system equilibration have been described previously for GlcCerase[37], Pol III[38], TFIIH[21] and PIC[21]. In this work, production MD runs totaling 2.1 μs (GlcCerase), 200 ns (PolIII) and 300 ns (TFIIH and PIC) in the isothermal-isobaric ensemble were obtained using the Amber18 or Amber16 (GlcCerase) CUDA code[39,40]. In the simulations, temperature and pressure were set to 300 K and 1 atm, respectively. For analysis, snapshots from the MD trajectories were collected at 1-ps intervals for the GlcCerase system, 5-ps intervals for the Pol III system and 20-ps intervals for the TFIIH and PIC systems.

Adjacency matrix construction. For each system**,** nodes were positioned at all Cα atoms of protein residues and P atoms of DNA (if applicable). Edges were then drawn between nodes if the physical distance between them was less than 4.5 Å for more than 75% of the trajectory frames. Additionally, we ignored residues directly attached via atomic bonds (i.e. *i+1* and *i-1*). The cross-correlations between residues were then computed with CPPTRAJ[35] and used to weight the edges of in-contact nodes in the manner described above. Sources and targets for all suboptimal path calculations are included in Table 1.

Weighted implementation of suboptimal paths (WISP) protocol. As input for WISP, we provided the adjacency matrix and the contact map of each simulated system. One of the benefits of WISP is that it will compute both of these objects directly from the MD trajectories. However, for consistency and to ensure that the runtime only reflects the suboptimal paths calculation, we supplied pre-computed adjacency matrices and contact maps to initiate WISP. Another benefit of WISP is the parallelization, which enables the code to utilize multiple cores on a CPU. Thus, for each system, the WISP calculation employed 6 cores on a Linux 64-bit machine with an Intel i7-5930K processor.

Suboptimal paths with NetworkView. Separate from the NetworkView plugin to VMD is the *subopt* executable, written entirely in the C programming language. A natural benefit of this code is that it is compiled for the specific machine architecture and, thus, tends to outperform programs written in interpretive languages (e.g. Python or R). As input for *subopt*, we only provided the pre-computed adjacency matrices. Additionally, *subopt* requires an offset value to be added to the optimal path length. This value is set by the user. Large offsets tend to increase the search space, while small offsets can severely limit the search space. To be consistent with both WISP and our method, we subtracted the optimal path length from the largest path length found with our method to determine the offset for *subopt*. Since *subopt* does not have a parameter for the number of desired paths, fine-tuning the offset was the only way to control the number of computed

suboptimal paths. Unlike WISP, *subopt* has not been parallelized to utilize multiple cores. Thus, each calculation only used 1 core on a Linux 64-bit machine with an Intel i7-5930K processor.

Suboptimal paths with CNAPATH in Bio3D. This protocol implements the original Yen's algorithm[36] for finding *k*-shortest paths. First, a dynamical network is built using the 'cna()' function from Bio3D. Then, the 'cnapath()' function of Bio3D is called to search for (sub)optimal paths. The function is parallelized and was tested using 16 or 2 (for the PIC system) CPU cores (Intel Xeon E5-2665) on a Linux 64-bit machine. A more detailed description of the protocol along with examples of application is available[32].

Suboptimal paths with the Subsets of Adjacent Nodes (SOAN) method. As input for our method, we only provided the pre-computed adjacency matrices. Since our code will determine when and if it needs to expand beyond the neighbors directly adjacent to the optimal path, we left the neighbor level at the default value of 1. Like *subopt*, our code is not parallelized for multiple cores, thus, each calculation was performed using 1 core on a Linux 64-bit machine with an Intel i7-5930K processor.

Results and Discussion

We first tested the computational efficiency of each suboptimal path method using four distinct biological systems of increasing size: (i) acid-β-glucosidase (GlcCerase), a 497 amino acid lysosomal hydrolase[41], (ii) DNA polymerase III (Pol III), a 2033 residue replicative enzyme in *E. Coli* bacteria[42], (iii) human transcription factor IIH (TFIIH), a complex comprised of 10 protein subunits and 3995 amino acids involved in both transcription and nucleotide excision repair[43], and (iv) human pre-initiation complex (PIC), a massive 22-subunit and 9900 amino acid complex involved in transcription initiation[44]. All systems have been previously characterized using graph theory methods[21,37,38] and, therefore, serve as excellent test cases for comparison of methods for suboptimal path determination. For each system and method tested, we determined the runtime needed to compute 1000 shortest paths (Table 2 and **Fig. 2**). In the case of Pol III, TFIIH and PIC, WISP did not finish in a practical amount of time (see note in Table 2) and was, therefore, excluded from direct efficiency comparison with the other methods. Overall, our results demonstrate that SOAN is remarkably fast compared to the other approaches. The only exception is the runtime of NetworkView for the smallest of our test systems, GlcCerase. However, even for GlcCerase, both NetworkView and SOAN performed within a negligible difference of ~ 0.5 seconds. More importantly, as the number of edges in the simulated system increases, SOAN demonstrates the best scalability. In contrast, the runtime of CNAPATH and NetworkView

increases faster with the increase of the total number of edges (**Fig. 2**). NetworkView performs well on small to medium size systems but its advantages disappear for protein networks the size of PIC with ~ 35,000 edges. Moreover, SOAN still outperforms NetworkView for Pol III (7,405 edges), TFIIH (12,896 edges) and PIC (32,851 edges), with speedups of ~ 3x or greater. Thus, for larger systems SOAN appears to be an optimal choice.

Next, we determined the node degeneracy for each residue in each set of computed paths. Node degeneracy offers a quantitative approach to identify key residues in the protein topology that may be critical for allosteric signaling and communication. Here we compare per-residue node degeneracies computed with SOAN to ones obtained with other network analysis approaches. Suboptimal paths computed from SOAN were mapped to the individual structures, in addition to node degeneracies computed from paths determined with each method (**Fig. 3** and **4**). For GlcCerase, SOAN was found to be consistent with the results of WISP, NetworkView and CNAPATH. Minor deviations were observed for the individual node degeneracies, the largest of which was 0.034 for residue F417 of GlcCerase. Similarly, for Pol III, SOAN is consistent with CNAPATH and NetworkView with the largest deviation coming from residue L494 (0.069). For larger systems (TFIIH and PIC), the SOAN results are in excellent agreement with CNAPATH. We observed very minor deviations in node degeneracies for TFIIH, with the largest being 0.003 for residue N299. For PIC, the node distributions determined from SOAN and CNAPATH match exactly. Surprisingly, for the TFIIH and the PIC systems, node degeneracies computed with NetworkView deviate substantially from the ones computed with SOAN or CNAPATH. Notably, 8 out of the 22 node degeneracies computed for TFIIH have a difference of 0.12 or higher. For PIC, NetworkView's consistency with SOAN and CNAPATH is considerably worse. Nearly 40% of the node degeneracies calculated with NetworkView have a deviation of 0.10 or higher. Strikingly, 53% of these nodes are not predicted at all with NetworkView. A possible explanation is that NetworkView is based on Floyd's algorithm[45], while the SOAN and CNAPATH codes are all based on some variant of Yen's algorithm. In addition to algorithmic differences, NetworkView ignores the decimal when summing the edge weights (i.e. 0.94 becomes 94). Combined with manually setting an offset, this could potentially lead to accumulation of errors and substantial deviations in the suboptimal path distribution for larger systems. Thus, as the search space grows, the NetworkView algorithm could become more numerically unstable. Indeed, in the calculated top 1000 suboptimal paths, both SOAN and CNAPATH give the same range of path length for all the test systems (e.g., 1.507–1.512 for PIC), which is shorter than NetworkVIew (1.528–1.544). This indicates that SOAN and CNAPATH are more accurate because suboptimal paths are defined by a set of unique loopless shortest paths. However, we note that without access to the source code

of *subopt* (the program for searching suboptimal paths in the NetworkView package), it is impossible for us to pinpoint the exact source of this numerical behavior of NetworkView for large biological systems. In any case, SOAN appears to be extremely accurate when compared with its algorithmically related predecessor CNAPATH for all system sizes.

Minor deviations in node degeneracies are easily attributable to the reduced search space from taking a subset of the original network graph. To evaluate how the reduction of search space affects the SOAN protocol, we tested the algorithm at different expansion levels (up to 10 levels), assessing both the accuracy and runtime at each level in relation to the full graph (**Fig. 5 and Table 3**). For each level, the first 1000 shortest paths were computed. The accuracy of each level was determined using the root mean square error (RMSE), a global metric which measures the standard deviation of the differences in the node degeneracies between any given expansion level and the full graph. In general, the SOAN protocol demonstrates a linear increase in runtime (except for GlcCerase, in which the increase of runtime substantially slows down after level 3), accompanied by only minor increases in the overall accuracy. Runtimes between levels 1 and 2 appear to show the most increase with a factor of ~ 3 in every case. Importantly, the RMSE converges to full graph result within the first two levels. The only exception to this is GlcCerase which converges at level 3. Thus, as system size increases, less expansion around the optimal path is required to achieve accuracy comparable to the evaluation on the full graph. It is important to note that the global error of the first level is relatively small for every test case. Expanding to the second level results in marginal improvements in accuracy. Considering the larger systems (TFIIH and PIC), the loss in accuracy at the first level is negligible, and in the case of the PIC, nonexistent. Thus, larger systems benefit from the speedup of level 1, while suffering minimal losses in accuracy.

Another free parameter of the SOAN method is the number of suboptimal paths, k. To evaluate how node degeneracy varies over k, we scanned a range of k from 1 up to 5000 (**Fig. 6**). From the result, we verified that computing 1000 suboptimal paths is sufficient to obtain converged node degeneracies and determine the critical nodes along allosteric communication pathways. The maximal degeneracy difference between k=1000 and k=5000 is 0.07–0.11 depending on the system. In general, smaller systems converge slightly faster than larger systems (**Fig. 6 A** and **6B** versus **Fig. 6C** and **6D**).

Conclusions

Here, we present SOAN, a method for rapidly computing suboptimal paths between defined sites in dynamic networks of proteins and macromolecular assemblies. A unique feature of SOAN is that it searches a reduced representation of the protein network graph, one that is derived from neighbors of the optimal path. This feature drastically reduces the calculation runtime. Another powerful feature of the SOAN approach is the freedom to choose the level of graph reduction, allowing users to fine-tune the accuracy of the method. Thus, users can strike a balance between speed and accuracy.

By benchmarking on four distinct biological systems we have demonstrated the utility and versatility of SOAN. Large systems benefit substantially from the SOAN approach. More importantly, SOAN outperforms other popular software packages. The SOAN method has been implemented in Python and made available on GitHub (https://github.com/tdodd3/SOAN).

Acknowledgments

This work was supported by National Science Foundation grant MCB-2027902 to I.I. T.D. was supported by a Molecular Basis of Disease fellowship from Georgia State University. X.Q.Y. and D.H. are supported by National Science Foundation grant MCB-2018144. Computational resources were provided in part by an allocation from the National Science Foundation XSEDE program CHE110042. An award of computer time was provided by the INCITE program. This research also used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725.

Disclosure Statement

No potential competing interest was reported by the authors.

Data Availability

This code is freely available at https://github.com/tdodd3/SOAN .

Figure Legends

Figure 1. Schematic representation of the SOAN protocol

Figure 2. Runtimes of SOAN, CNAPATH and NetworkView as a function of total network edges. Runtimes to compute 1000 shortest paths for each network (in seconds) are plotted on a logarithmic scale versus the number of edges. The inset denotes the method used.

Figure 3. Suboptimal paths and node degeneracies generated with the SOAN method. The first 1000 suboptimal paths (red tubes) were generated using SOAN and mapped to the structure of (A) GlcCerase, (B) Pol III and (C) TFIIH. Source and target nodes are denoted by black spheres and labeled. Critical nodes are highlighted by green spheres. Node degeneracies of highlighted critical nodes for (D) GlcCerase, (E) Pol III and (D) TFIIH. Inset denotes method used to calculate node degeneracies.

Figure 4. Suboptimal paths generated from SOAN for the PIC. The first 1000 suboptimal paths (red tubes) mapped to the structure of the PIC. Critical nodes are highlighted by domain(s), colored and labeled in the inset. Node degeneracies are outlined by a colored box denoted by domain inset. The bar graph inset denotes method used to computed node degeneracies.

Figure 5. Comparison of different expansion levels for the SOAN method. Runtimes to find the first 1000 shortest paths at expansion levels 1-10 for (A) GlcCerase, (B) Pol III, (C) TFIIH and (D) PIC. The time to reach solution using the full graph is denoted by the dashed red line. RMSE of node degeneracies between levels 1-10 and full graph for (E) GlcCerase, (F) Pol III, (G) TFIIH and (H) PIC.

Figure 6. Convergence of node degeneracy over the number of suboptimal paths for SOAN. Computed node degeneracies using varying number of suboptimal paths (*k*) up to 5000 are displayed for select residues (as colored lines) in (A) GlcCerase, (B) Pol III, (C) TFIIH and (D) PIC. Residues are chosen if they show significant (>0.25) node degeneracy at any evaluated *k* value (1–5000).

References

1. Wagner, J.R. et al. Emerging Computational Methods for the Rational Discovery of Allosteric Drugs. *Chem Rev* **116**, 6370-90 (2016).
2. Raman, S. Systems Approaches to Understanding and Designing Allosteric Proteins. *Biochemistry* **57**, 376-382 (2018).
3. Monod, J., Wyman, J. & Changeux, J.P. On the Nature of Allosteric Transitions: A Plausible Model. *J Mol Biol* **12**, 88-118 (1965).
4. Koshland, D.E., Jr., Nemethy, G. & Filmer, D. Comparison of Experimental Binding Data and Theoretical Models in Proteins Containing Subunits. *Biochemistry* **5**, 365-85 (1966).
5. Cooper, A. & Dryden, D.T. Allostery without Conformational Change. A Plausible Model. *Eur Biophys J* **11**, 103-9 (1984).
6. Tsai, C.J., del Sol, A. & Nussinov, R. Allostery: absence of a change in shape does not imply that allostery is not at play. *J Mol Biol* **378**, 1-11 (2008).

7. del Sol, A., Fujihashi, H., Amoros, D. & Nussinov, R. Residues crucial for maintaining short paths in network communication mediate signaling in proteins. *Mol Syst Biol* **2**, 2006 0019 (2006).
8. Di Paola, L., De Ruvo, M., Paci, P., Santoni, D. & Giuliani, A. Protein contact networks: an emerging paradigm in chemistry. *Chem Rev* **113**, 1598-613 (2013).
9. Di Paola, L. & Giuliani, A. Protein contact network topology: a natural language for allostery. *Curr Opin Struct Biol* **31**, 43-8 (2015).
10. Chennubhotla, C. & Bahar, I. Signal propagation in proteins and relation to equilibrium fluctuations. *PLoS Comput Biol* **3**, 1716-26 (2007).
11. Sethi, A., Eargle, J., Black, A.A. & Luthey-Schulten, Z. Dynamical networks in tRNA:protein complexes. *Proc Natl Acad Sci U S A* **106**, 6620-5 (2009).
12. Vijayabaskar, M.S. & Vishveshwara, S. Interaction Energy Based Protein Structure Networks. *Biophys J* **99**, 3704-15 (2010).
13. Ribeiro, A.A. & Ortiz, V. Determination of Signaling Pathways in Proteins through Network Theory: Importance of the Topology. *J Chem Theory Comput* **10**, 1762-9 (2014).
14. Yao, X.Q., Momin, M. & Hamelberg, D. Elucidating Allosteric Communications in Proteins with Difference Contact Network Analysis. *J Chem Inf Model* **58**, 1325-1330 (2018).
15. Kannan, N. & Vishveshwara, S. Identification of side-chain clusters in protein structures by a graph spectral method. *J Mol Biol* **292**, 441-64 (1999).
16. Ghosh, A. & Vishveshwara, S. A Study of Communication Pathways in Methionyl- tRNA Synthetase by Molecular Dynamics Simulations and Structure Network Analysis. *Proc Natl Acad Sci U S A* **104**, 15711-15716 (2007).
17. Gasper, P.M., Fuglestad, B., Komives, E.A., Markwick, P.R. & McCammon, J.A. Allosteric Networks in Thrombin Distinguish Procoagulant vs. Anticoagulant Activities. *Proc Natl Acad Sci U S A* **109**, 21216-22 (2012).
18. VanWart, A.T., Eargle, J., Luthey-Schulten, Z. & Amaro, R.E. Exploring Residue Component Contributions to Dynamical Network Models of Allostery. *J Chem Theory Comput* **8**, 2949-2961 (2012).
19. Scarabelli, G. & Grant, B.J. Kinesin-5 Allosteric Inhibitors Uncouple the Dynamics of Nucleotide, Microtubule, and Neck-Linker Binding Sites. *Biophys J* **107**, 2204-13 (2014).
20. Yao, X.Q. et al. Dynamic Coupling and Allosteric Networks in the Alpha Subunit of Heterotrimeric G Proteins. *J Biol Chem* **291**, 4742-53 (2016).
21. Yan, C. et al. Transcription preinitiation complex structure and dynamics provide insight into genetic diseases. *Nature Structural & Molecular Biology* (2019).
22. Newman, M.E. Modularity and community structure in networks. *Proc Natl Acad Sci U S A* **103**, 8577-82 (2006).
23. Newman, M.E. The structure and function of complex networks. *SIAM review* **45**, 167-256 (2003).
24. Eargle, J. & Luthey-Schulten, Z. NetworkView: 3D display and analysis of protein.RNA interaction networks. *Bioinformatics* **28**, 3000-1 (2012).
25. Hagberg, A., Swart, P. & S Chult, D. Exploring network structure, dynamics, and function using NetworkX. (Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008).
26. Dubois, P.F. & Yang, T.-Y. Extending python with fortran. *Computing in science & engineering* **1**, 66-73 (1999).
27. Peterson, P. F2PY: a tool for connecting Fortran and Python programs. *International Journal of Computational Science and Engineering* **4**, 296-305 (2009).
28. Virtanen, P. et al. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature methods* **17**, 261-272 (2020).
29. Ascher, D., Dubois, P.F., Hinsen, K., Hugunin, J. & Oliphant, T. Numerical python. (Citeseer, 2001).

30. Grant, B.J., Rodrigues, A.P., ElSawy, K.M., McCammon, J.A. & Caves, L.S. Bio3d: An R Package for the Comparative Analysis of Protein Structures. *Bioinformatics* **22**, 2695-2696 (2006).
31. Skjærven, L., Yao, X.-Q., Scarabelli, G. & Grant, B.J. Integrating Protein Structural Dynamics and Evolutionary Analysis with Bio3D. *BMC Bioinformatics* **15**, 399 (2014).
32. Grant, B.J., Skjaerven, L. & Yao, X.Q. The Bio3D packages for structural bioinformatics. *Protein Sci* (2020).
33. Van Wart, A.T., Durrant, J., Votapka, L. & Amaro, R.E. Weighted Implementation of Suboptimal Paths (WISP): An Optimized Algorithm and Tool for Dynamical Network Analysis. *J Chem Theory Comput* **10**, 511-517 (2014).
34. Glykos, N.M. Software news and updates. Carma: a molecular dynamics analysis program. *J Comput Chem* **27**, 1765-8 (2006).
35. Roe, D.R. & Cheatham, T.E., 3rd. PTRAJ and CPPTRAJ: Software for Processing and Analysis of Molecular Dynamics Trajectory Data. *J Chem Theory Comput* **9**, 3084-3095 (2013).
36. Yen, J.Y. Finding the k shortest loopless paths in a network. *management Science* **17**, 712-716 (1971).
37. Souffrant, M.G., Yao, X.-Q., Momin, M. & Hamelberg, D. N-Glycosylation and Gaucher Disease Mutation Allosterically Alter Active-Site Dynamics of Acid-β-Glucosidase. *ACS Catalysis* **10**, 1810-1820 (2020).
38. Dodd, T. et al. Polymerization and editing modes of a high-fidelity DNA polymerase are linked by a well-defined path. *Nat Commun* **11**, 5379 (2020).
39. Gotz, A.W. et al. Routine microsecond molecular dynamics simulations with AMBER on GPUs. 1. Generalized born. *Journal of chemical theory and computation* **8**, 1542-1555 (2012).
40. Salomon-Ferrer, R., Götz, A.W., Poole, D., Le Grand, S. & Walker, R.C. Routine microsecond molecular dynamics simulations with AMBER on GPUs. 2. Explicit solvent particle mesh Ewald. *Journal of chemical theory and computation* **9**, 3878-3888 (2013).
41. Erickson, A.H., Ginns, E.I. & Barranger, J.A. Biosynthesis of the lysosomal enzyme glucocerebrosidase. *J Biol Chem* **260**, 14319-24 (1985).
42. Benkovic, S.J., Valentine, A.M. & Salinas, F. Replisome-mediated DNA replication. *Annual review of biochemistry* **70**, 181-208 (2001).
43. Tsutakawa, S.E. et al. Envisioning how the prototypic molecular machine TFIIH functions in transcription initiation and DNA repair. *DNA repair* **96**, 102972 (2020).
44. He, Y. et al. Near-atomic resolution visualization of human transcription promoter opening. *Nature* **533**, 359-365 (2016).
45. Floyd, R.W. Algorithm 97: shortest path. *Communications of the ACM* **5**, 345 (1962).

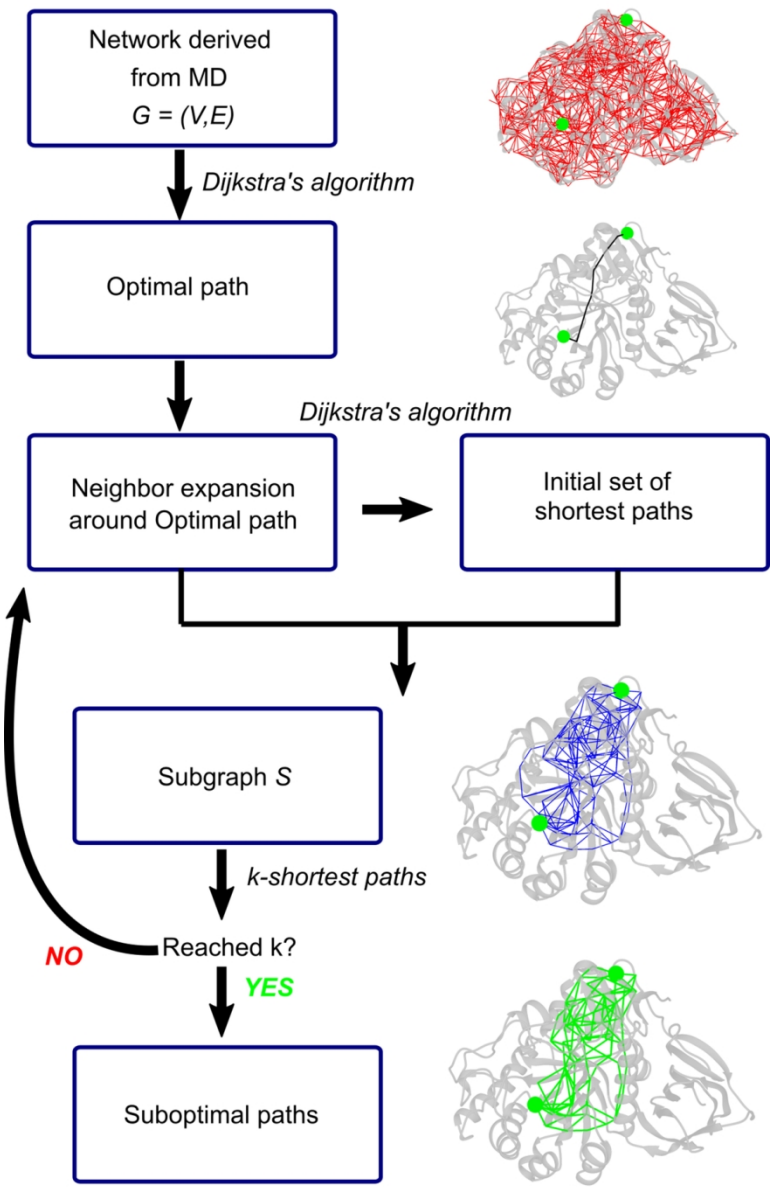


Figure 1

101x158mm (300 x 300 DPI)

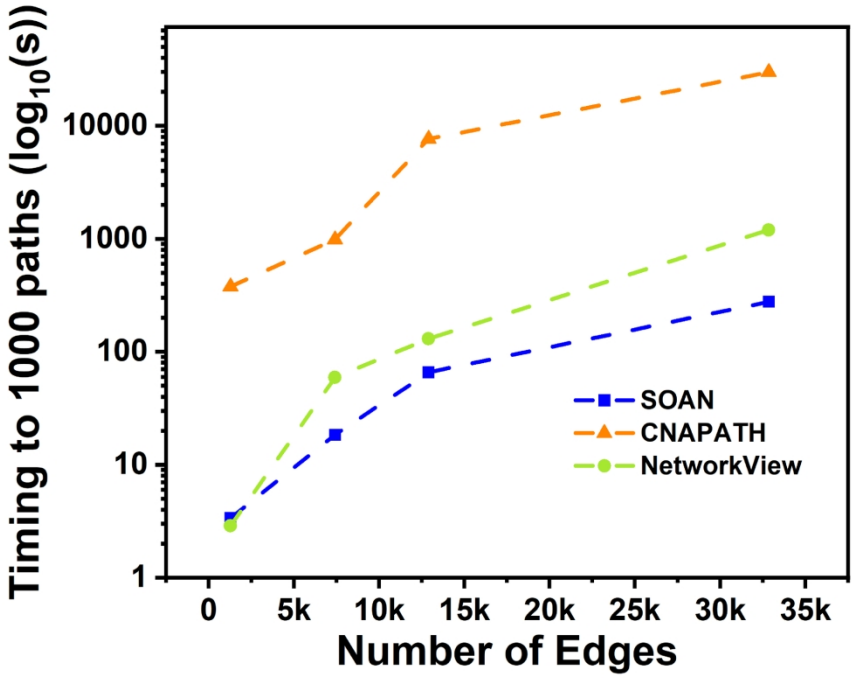


Figure 2

193x147mm (300 x 300 DPI)

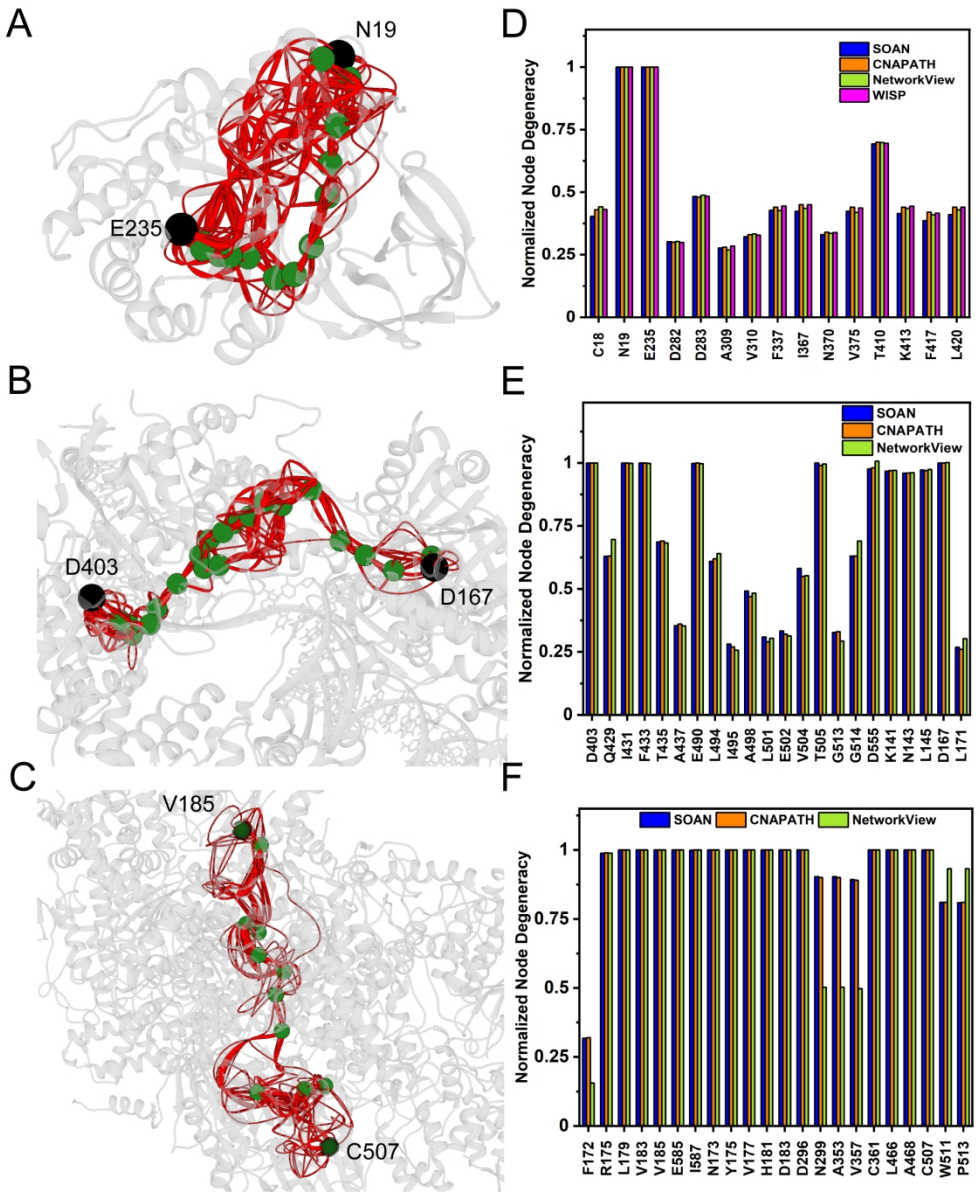

A
N19
E235
B
D403
D167
C
V185
C507
D
SOAN
CNAPATH
NetworkView
WISP
Normalized Node Degeneracy
C18 N19 E235 D282 D283 A309 V310 F337 I367 N370 V375 T410 K413 F417 L420
E
SOAN
CNAPATH
NetworkView
Normalized Node Degeneracy
D403 Q429 I431 F433 T435 A437 E490 L494 I495 A498 L501 E502 V504 T505 G513 G514 D555 K141 N143 L145 D167 L171
F
SOAN
CNAPATH
NetworkView
Normalized Node Degeneracy
F172 R175 L179 V183 V185 E585 I587 N173 Y175 V177 H181 D183 D296 N299 A353 V357 C361 L466 A468 C507 W511 P513

Figure 3

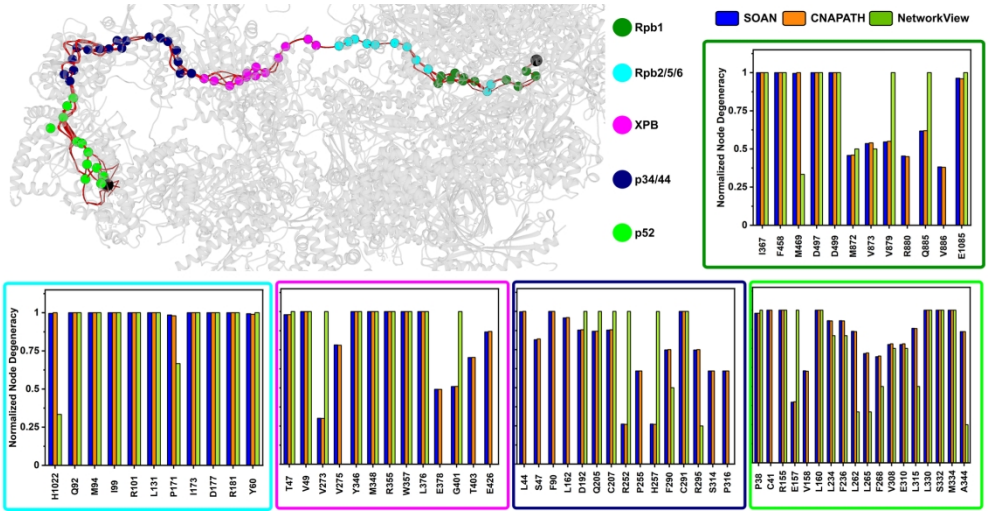


Figure 4

193x97mm (300 x 300 DPI)

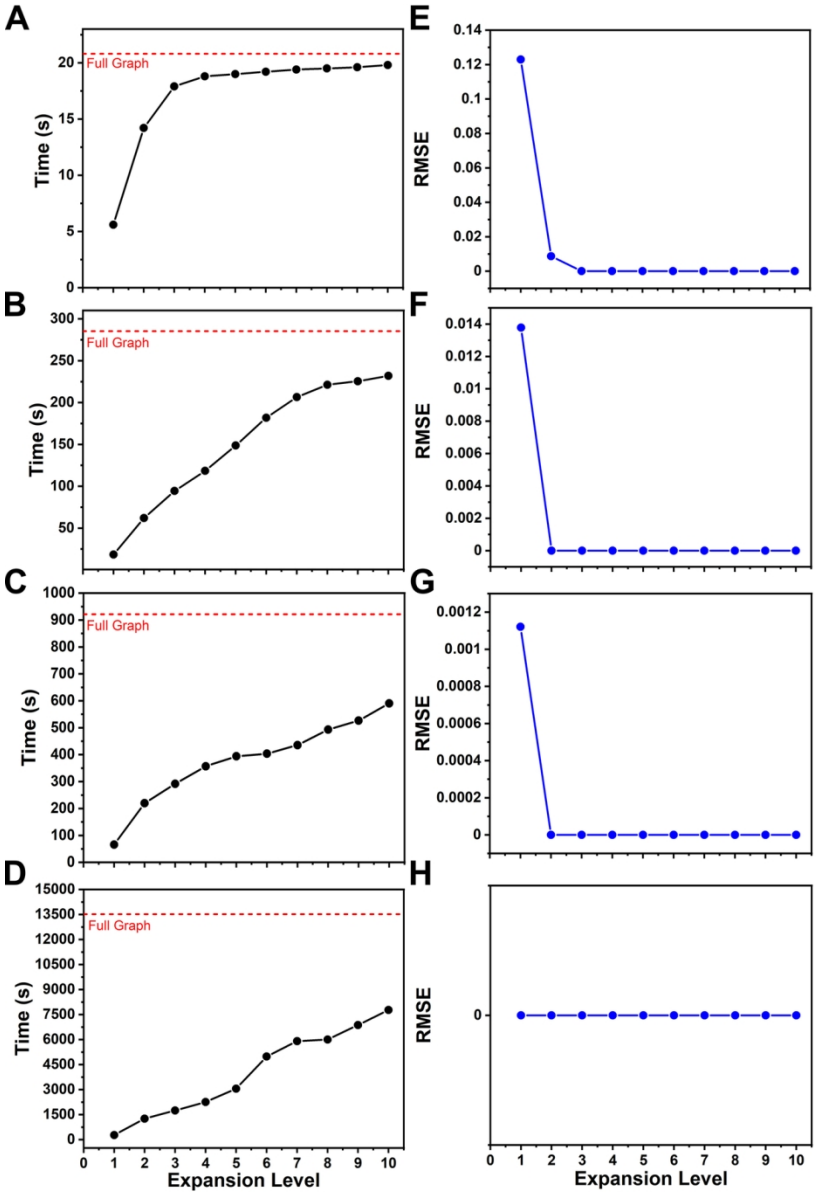


Figure 5

114x166mm (300 x 300 DPI)

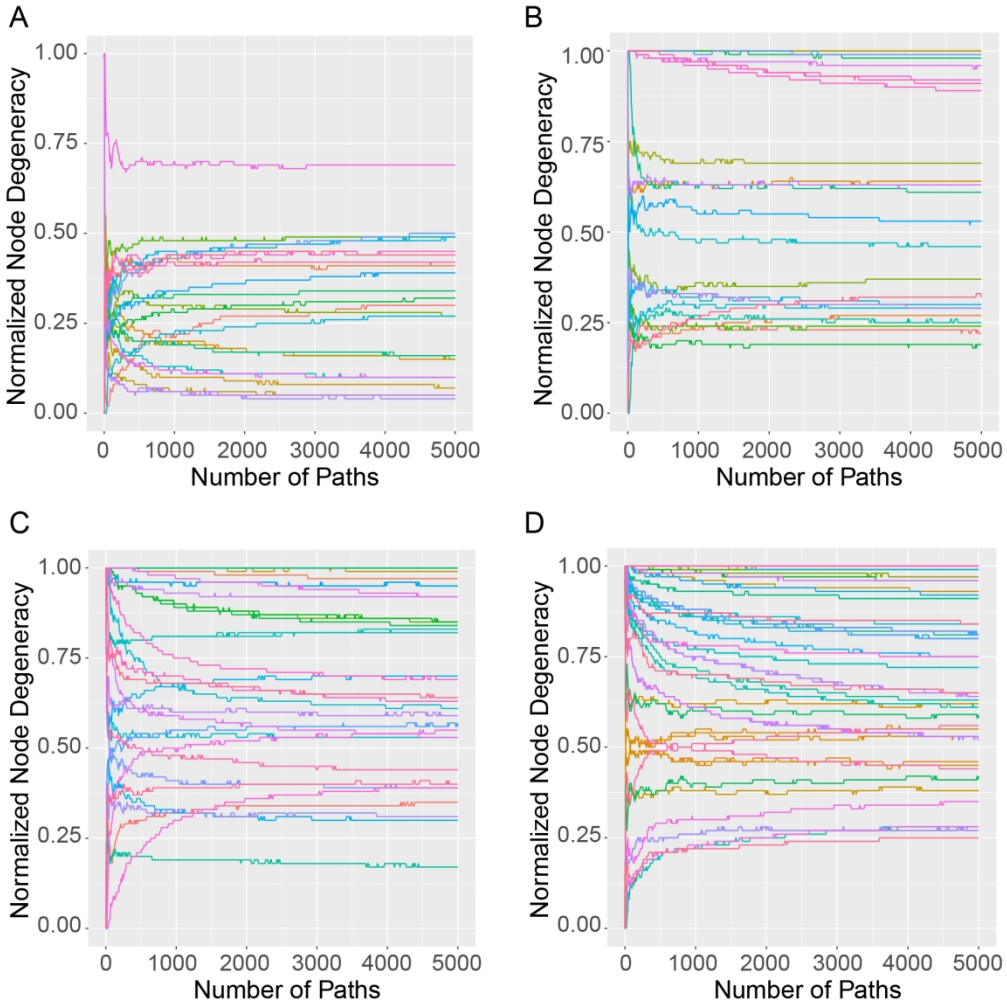

A
B
C
D
Normalized Node Degeneracy
1.00
0.75
0.50
0.25
0.00
0
1000
2000
3000
4000
5000
Number of Paths

Figure 6

Table 1 – Sources and targets used for suboptimal paths calculations

System	Source	Target
GlcCerase	Asn19	Glu235
PolIII	Asp403	Asp17
TFIIH	Val185	Cys507
PIC	Asp497	Cys291

Table 2 – Timings (in seconds) to find 1000 paths in all systems using SOAN, WISP, NetworkView, and CNAPATH.

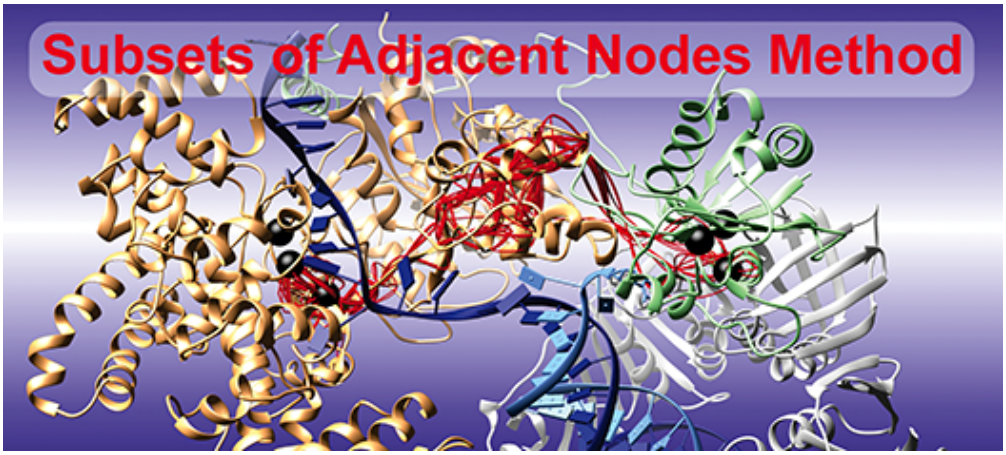
System	Size (#nodes, #edges)	SOAN	WISP	NetworkView	CNAPATH
GlcCerase	497, 1279	3.67	1487.43	2.88	378.78
PolIII	2033, 7405	18.4	N/A[1]	59.18	985.71
TFIIH	3995, 12896	65.8	N/A[1]	130.47	7671.68
PIC	9900, 32851	277.1	N/A[1]	1200.02	29855.11

1 – Calculation did not finish within 20 days

Table 3 – Timings (in seconds) at increasing expansion levels

	Glucosidase		Pol III		TFIIH		PIC	
Level	**Time (s)**	**% of Graph**	**Time (s)**	**% of Graph**	**Time (s)**	**% of Graph**	**Time (s)**	**% of Graph**
1	3.7	15	18.4	4	65.8	3	**277.1**	**4**
2	14.2	47	**62.1**	**17**	**219.7**	**12**	1259.0	14
3	**17.9**	**68**	94.4	27	291.7	15	1751.0	19
4	18.8	73	118.4	35	356.9	18	2259.4	24
5	19.0	79	148.7	44	393.8	20	3055.1	29
6	19.2	83	181.9	52	403.4	22	4982.3	34
7	19.4	85	206.5	57	435.4	24	5908.1	39
8	19.5	86	221.3	60	493.1	26	6005.2	44
9	19.6	88	225.5	63	526.3	27	6871.6	47
10	20.0	90	231.9	66	590.6	29	7776.8	52
Full Graph	20.8	100	285.4	100	921.2	100	13521.6	100

Values in bold indicate where node degeneracy converges with full graph



Graphical Abstract