An Efficient Approach to the Protein Structure Alignment Problem

Mikhail Batsyn, Evgeny Maslov, Alexey Nikolaev, Pablo San Segundo

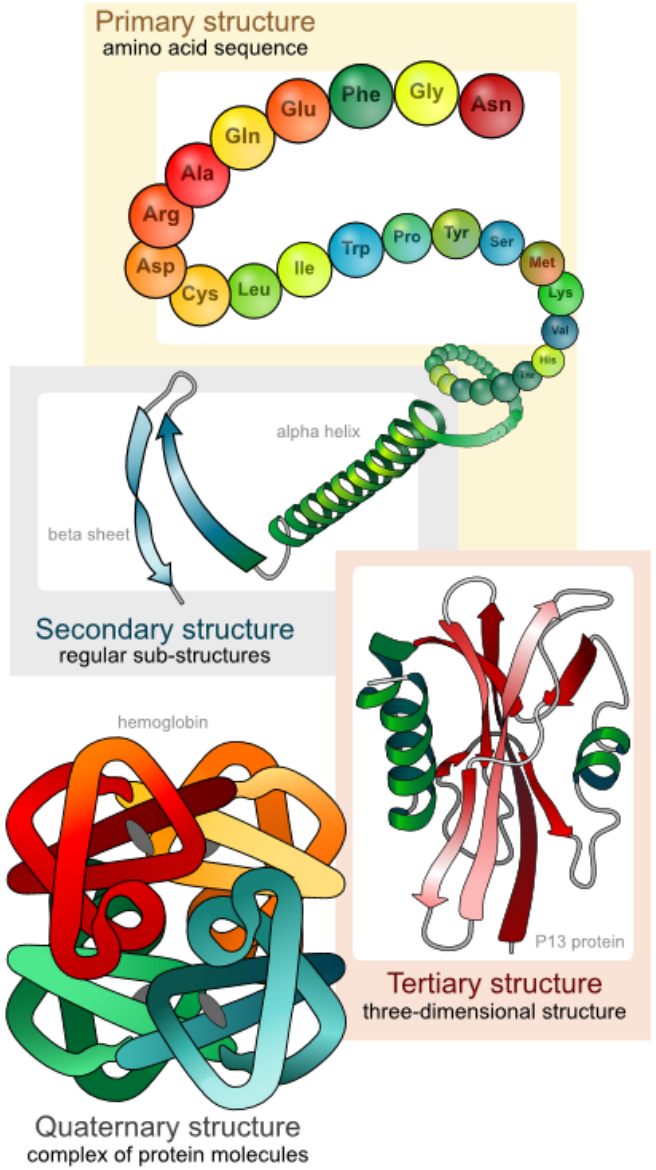
Laboratory of Algorithms and Technologies for Network Analysis (LATNA)
National Research University Higher School of Economics
Nizhny Novgorod, Russia

Outline

- Protein structure alignment problem
- Suggested approach
- Computational results
- Summary

Protein structure

- Primary structure – amino acid sequence
- Secondary structure – regular substructures
- Tertiary structure – three-dimensional structure
- Quaternary structure – complex of protein molecules

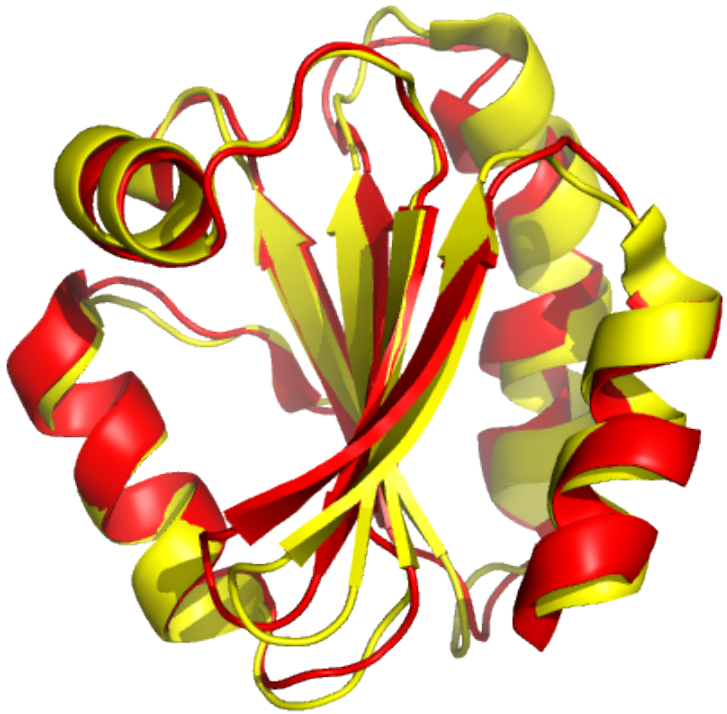


Protein structure alignment problem

Objective – an optimal alignment between two protein amino acid structures.

This problem is NP-hard since it can be polynomially reduced to the maximum clique problem.

Proteins of similar 3D-structure usually have same functions



Existing methods

- CMO – Contact Map Overlap
  A. Godzik and J. Skolnick. Flexible algorithm for direct multiple alignment of protein structures and sequences.
  CABIOS, 10:587–596, 1994.
- VAST – Vector Alignment Search Tool
  J-F. Gibrat, T. Madej, and S.H. Bryant. Surprising similarities in structure comparison.
  Current Opinion in Structural Biology., 6:377–385, 1996.
- DAST – Distance-based Alignment Search Tool
  N. Malod-Dognin, R. Andonov, N. Yanev. Maximum Cliques in Protein Structure Comparison.
  SEA 2010 9th International Symposium on Experimental Algorithms, 6049, 106-117, 2010.

Suggested approach

- Based on the efficient branch and bound algorithm for the maximum clique problem developed by authors.
- Applies powerful preprocessing to the input graph
- Has general nature and does not depend on the alignment graph structure

Protein alignment

A protein is defined by a sequence of amino acids.

An amino acid is specified by its 3D-coordinates (the coordinates of its α-carbon atom are used).

Amino acid i of protein P_1 is compatible with amino acid k of protein P_2 ($i \leftrightarrow k$) if i and k belong to the same secondary structure element: α-helix, β-strand, or coil (we use KAKSI software for SSE prediction).

Distance between amino acids i and j is measured as Euclidean distance: $d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}$

Protein alignment

Objective – find the largest set of amino acids i_1, i_2, …, i_ω in P_1 matching a set of amino acids k_1, k_2, …, k_ω in P_2 so that each pair i_a, i_b ($i_a < i_b$) in P_1 matches the pair k_a, k_b ($k_a < k_b$) in P_2:

1) $i_a < i_b$, $k_a < k_b$: the order is preserved

2) $i_a \leftrightarrow k_a$: i_a, k_a belong to the same SSE

   $i_b \leftrightarrow k_b$: i_b, k_b belong to the same SSE

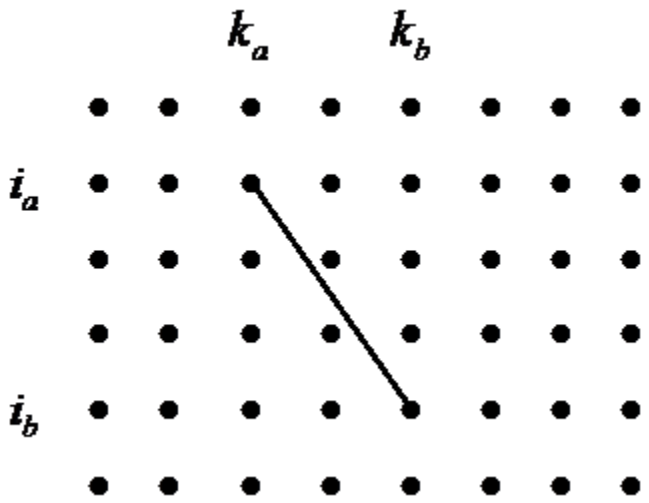
3) $|d_{i_a i_b} - d_{k_a k_b}| < \tau$: distances are almost the same

Protein alignment graph

Vertices: $(i.k)$ where $i \in P_1$, $k \in P_2$ and $i \leftrightarrow k$

Edges: $(i_a.k_a)$ and $(i_b.k_b)$ are connected with an edge if $i_a < i_b$, $k_a < k_b$, $|d_{i_a i_b} - d_{k_a k_b}| < \tau$

Objective – find the maximum clique in the alignment graph

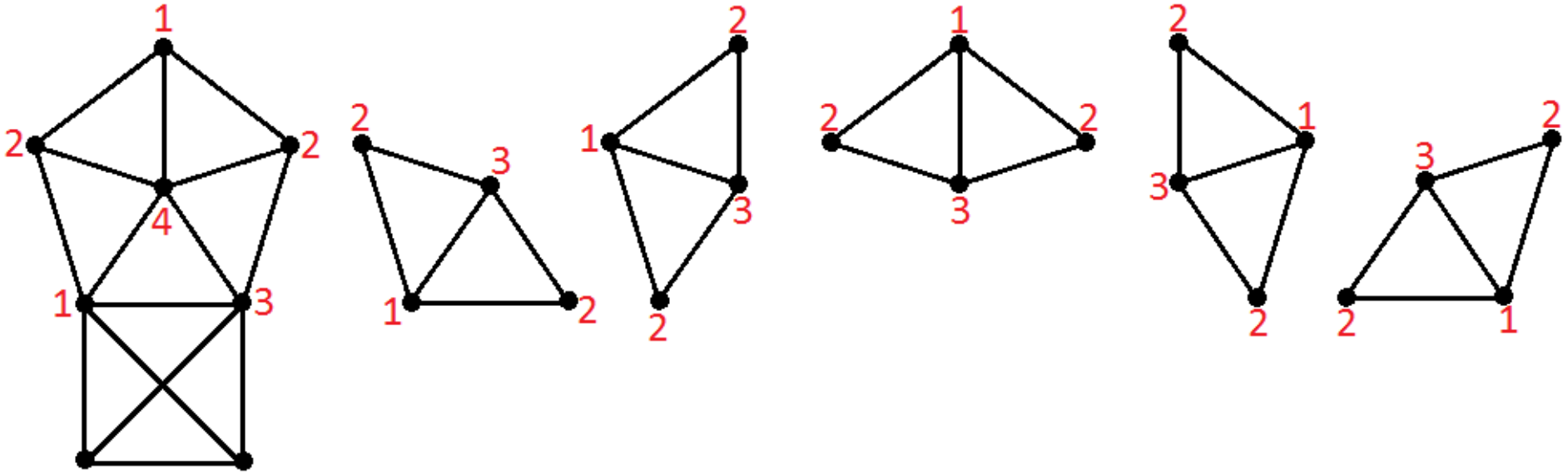


Alignment graph preprocessing

- Heuristic solution – Iterated Local Search (ILS):

Andrade, D.V., Resende, M.G.C., Werneck, R.F. Fast local search for the maximum independent set problem. Journal of Heuristics 18(4), 525-547 (2012)

- Remove vertices for which an upper bound is not greater than the heuristic solution value.



Computational results

40 proteins with 97 – 256 amino acids from Skolnick test set are used:
http://cssb.biology.gatech.edu/skolnick/people/jeff.html

For large alignment graphs the preprocessing reduces the number of vertices by more than 15% in average and the number of edges by more than 20%.

For the largest graph: $|V|=25100, |E|=31259143$

After preprocessing: $|V|=19268, |E|=23333871$

Computational results

Table 1. Running time comparison for small alignment graphs

Instances		Ostergard, sec	Our, sec
d1k32b	d1n6ei	6.20	2.36
d1k32b	d1n6fb	2.48	2.05
d1k32b	d1n6ff	2.41	2.06
d1k32b	d1n6dd	24.01	2.77
d1n6dd	d1n6ei	9.02	2.32
d1n6dd	d1n6fb	87.02	2.67
d1n6dd	d1n6ff	62.11	2.67

ACF algorithm is 9 times faster in average than Ostergard on these graphs. Our approach is 11 times faster.

Computational results

Table 2. Running time comparison for large instances

Instances		$\|V\|$	$\|E\|$	ω	ω_{ILS}	ILS + Preprocess, sec	Ostergard, sec	Our, sec
d1amk_	d1b9bA	24208	28526291	76	76	68	4000000	423
d1amk_	d1tri_	23688	27254559	50	50	23	4000000	5260
d1amk_	d3ypiA	23152	26096428	53	53	81	4000000	1257
d1amk_	d8timA	23838	27739393	42	42	1694	1975437	18060
d1aw2A	d3ypiA	23893	27528114	50	50	866	4000000	23331
d1amk_	d1aw2A	25100	30045134	52	52	404	4000000	11320
d1aw2A	d1btmA	25423	30655812	43	40	288	4000000	9386
d1aw2A	d1tmhA	25706	31397640	65	65	215	4000000	8365
d1aw2A	d1treA	25448	30898250	46	46	551	469600	2846
d1tmhA	d1treA	25262	30270402	42	42	1230	3133832	9068

On large instances the suggested approach is more than 375 times faster than Ostergard

On moderate instances our approach is 40 times faster in average than Ostergard. ACF is only 20 times faster on such instances

Table 3. Running time comparison for moderate instances

Instances		$\|V\|$	$\|E\|$	ω	ω_{ILS}	ILS, sec	Ostergard, sec	Our, sec
d1amk_	d2b3iA	6192	2127737	23	23	1.9	50	9
d1b00A	d1b9bA	11467	7159288	31	31	20.9	3197	111
d1b00A	d1htiA	11221	6637902	41	41	2.4	9750	19
d1b71A	d8timA	16717	14576531	34	23	133.6	5667	1015
d1b9bA	d2pcy_	6735	2376861	23	22	17.2	122	29
d1b9bA	d3chy_	6340	2353878	29	29	3.4	13698	17
d1bawA	d1btmA	6861	2939442	25	25	3.9	2172	37
d1bawA	d2plt_	4692	1734838	27	27	0.6	91	5
d1bawA	d4tmyA	3156	719655	21	21	0.8	6.7	1.6
d1byoA	d1treA	6621	2754094	25	25	2.7	104	15
d1dbwA	d1ydvA	11723	6964457	35	35	0.8	3609	61
d1dpsA	d1nat_	3133	378554	25	21	0.1	219	0.4
d1dpsA	d1qmpA	3009	387178	25	17	0.0	748	0.3
d1dpsA	d3chy_	2240	167322	26	14	0.8	362	0.9
d1dpsA	d4tmyA	2100	144179	26	26	0.0	94	0.1
d1ier_	d1qmpB	9667	5133743	27	27	13.9	4849	65
d1ier_	d3chy_	10189	5544475	26	26	9.2	4632	77
d1kdi_	d1tri_	6417	2433204	27	27	0.0	3936	10
d1kdi_	d2b3iA	4427	1184704	38	38	0.1	5.5	0.7
d1nin_	d8timA	6969	2737857	23	23	5.5	667	18
d1ntr_	d4tmyB	5078	1706264	29	29	0.8	44	3
d1pla_	d1treA	7009	2917919	24	24	30.5	157	46
d1qmpA	d1qmpD	5997	2374341	46	46	0.9	6217	2.3
d1qmpC	d1qmpD	5892	2387556	40	40	0.8	4476	3.8
d1qmpD	d1treA	12084	7772858	37	37	36.7	8788	164
d1qmpD	d3chy_	6119	2523565	46	46	1.0	1291	3.1
d1rn1A	d2pcy_	4515	1331519	22	22	1.7	68	5.1
d1ydvA	d2b3iA	6541	2249656	22	22	1.0	116	11
d1ydvA	d4tmyA	11063	6236246	42	42	4.9	240	18
d3chy_	d4tmyB	5898	2324115	31	31	0.7	751	5.5

Summary

- The suggested approach is 50 times faster in average than the Ostergard's algorithm on protein alignment graphs.
- The ACF algorithm is only 16 times faster than the Ostergard's algorithm on these graphs.
- The preprocessing gives more than 20% reduction of computational time for large graphs.
- The algorithm does not use a special structure of the alignment graph as ACF does.