Protein structure alignment using maximum cliques and local search

Protein Structure Alignment Using Maximum Cliques and Local Search

Wayne Pullan

School of Information and Communication Technology,
Griffith University, Gold Coast, QLD, 4215, Australia
Email: w.pullan@griffith.edu.au

Abstract. The protein structure alignment problem addresses the question of measuring the degree of similarity, in three–dimensional structure, of two proteins. The representation of each protein using a simple contact map allows the correspondence graph for the protein pair to be generated and the maximum clique within this graph provides a measure of the structural similarity between the two proteins. This study uses a recently developed maximum clique algorithm, Phased Local Search (PLS), to locate the maximum cliques within correspondence graphs.

1. Introduction

In bioinformatics, structural comparison of proteins is useful in several domains. For example, as protein function is intrinsically tied to a protein's structure [1], identifying structural similarity between a protein and other proteins whose function is known can allow the prediction of that protein's function. Over the last decade, a number of techniques for structurally comparing proteins have been developed however, none have proved adequate across a range of applications. A relatively new technique, Contact Map Overlap (CMO), first proposed in [2] (and subsequently shown to be NP-complete [3]), is to identify alignments between protein contact maps with the goal of maximising the number of consistent alignments. A protein consists of a chain of residues (amino acids). When a protein folds into its tertiary (lowest energy) structure, residues that are not directly adjacent in the chain may be physically close in space. The contact map for a protein is a simple representation of this three–dimensional structure and is the matrix of all pairwise distances between the components of the protein. For this study, the components of the protein are identified as the alpha carbon atoms (C_α) of each amino acid. The contact map can be further simplified into a 0–1 contact map by encoding each pairwise distance as one if the pairwise distance is less than some distance threshold (typically in the range 4 - 8 Å and 5.5 Å in this study) and zero otherwise. As shown in [4], the CMO problem can be directly translated to a maximum clique (MC) problem which calls for finding the maximum sized sub-graph of pairwise adjacent vertices in a given graph. Formally, the MC problem can be stated as: Given an undirected graph $G = (V, E)$, where V is the set of all vertices and E the set of all edges, find a maximum size clique in G, where a clique K in G is a subset of vertices, $K \subseteq V$, such that all pairs of vertices in K are connected by an edge, *i.e.*, for

all $v, v' \in K$, $\{v, v'\} \in E$, and the size of the clique K is the cardinality $|K|$ of K. The maximum clique problem is to maximise $|K|$, the cardinality of K. MC is NP-hard and the associated decision problem is NP-complete [5]. Therefore, large and hard instances of MC are typically solved using heuristic approaches of which the most recent is Phased Local Search (PLS) [6], a reactive algorithm that interleaves sub-algorithms which alternate between sequences of iterative improvement and plateau search. The differences between these sub-algorithms are primarily in the vertex selection method and the perturbation mechanisms used to overcome search stagnation. Extensive computational experiments [6] have shown that PLS has equivalent, or improved, performance compared to other state-of-the-art MC search algorithms, on a range of widely studied benchmark instances.

2. The PLS Algorithm

PLS [6] is now described using the following additional notation:
$N(i) = \{j \in V : \{i, j\} \in E\}$ — the vertices adjacent to i; K — current clique of G; and, $C_p(K) = \{i \in V : |K \setminus N(i)| = p\}$, $p = 0, 1$ — the set of all vertices not adjacent to exactly p vertices in K.

Algorithm PLS (G**, *tcs*, *max-selections*)**

Input: graph G; integers *tcs* (target clique size), *max-selections*
Output: K of size *tcs* **or** 'failed'

1. *selections* $:= 0$, *pu* $:= 0$, *pd* $:= 2$;
2. $sa :=$ *Random*, *iterations* $:= 50$;
3. <Randomly select a vertex $v \in V$, $K := \{v\}$ >;
4. $\forall\, i \in V$, $p_i := 0$;
5. **do**
6. **do**
7. **while** $C_0(K) \setminus U \neq \emptyset$ **do**
8. $v :=$*Select*$(C_0(K), sa)$;
9. $K := K \cup \{v\}$;
10. *selections* := *selections* $+ 1$;
11. **if** $|K|$ = *tcs* **then return** K;
12. $U := \emptyset$;
13. **end while**
14. **if** $C_1(K) \setminus U \neq \emptyset$ **then**
15. $v :=$ *Select*$(C_1(K) \setminus U, sa)$;
16. $K := [K \cup \{v\}] \setminus \{i\}$, $U := U \cup \{i\}$, where $\{i\} = K \setminus N(v)$;
17. *selections* := *selections* $+ 1$;
18. **end if**;
19. **while** $C_0(K) \neq \emptyset$ **or** $C_1(K) \setminus U \neq \emptyset$;
20. $iterations := iterations - 1$;
21. *UpdatePenalties*(sa);
22. *Perturb*(sa);
23. **until** *selections* $\geq$ *max-selections*
24. **return** 'failed';

PLS uses three sub-algorithms within the *Select* function which are effective for three different instance types. The first sub-algorithm, *Random*, effectively solves instances where the maximal clique consists of vertices with a wide range of vertex degrees. The second sub-algorithm, *Penalty*, uses vertex penalties to bias the search towards cliques containing lower degree vertices. The vertex penalties are increased when the vertex is in the current clique when a perturbation occurs and are subject to occasional decrease, which effectively allows the sub-algorithm to 'forget' vertex penalties over time. PLS adaptively modifies the frequency of penalty decreases to obtain near optimal performance. The third PLS sub-algorithm, *Degree*, uses vertex degrees to bias the search towards cliques containing higher degree vertices.

3. Empirical Performance Results

For this study, two protein structure alignment benchmarks were utilised to evaluate the performance of PLS on this type of problem. Benchmark–1 was used in [4] (the correspondence graphs for this benchmark were obtained directly from the authors of this paper) and consists of 10 different protein structure alignment problems. The proteins in this benchmark all contain approximately 50 residues and the correspondence graphs have up to $3\,000$ vertices and $700\,000$ edges.

Benchmark–2 was constructed using proteins from the Protein Data Bank [7]. The Universality Similarity Measure (USM) software [1] [8] (with a 5.5 Å threshold) was used to generate the contact maps for these proteins. From the contact maps for the proteins to be compared, the two-dimensional grid G was generated and the correspondence graph created by adding an edge when the two alignments represented by pairs of vertices are a feasible solution to the CMO problem. The proteins in this benchmark range in size from 60 to 100 residues and the correspondence graphs have up to approximately $8\,000$ vertices and $9\,000\,000$ edges. All experiments for this study were performed on a dedicated computer that, for the DIMACS Machine Benchmark[2], required 0.41 CPU seconds for r300.5, 2.52 CPU seconds for r400.5 and 9.71 CPU seconds for r500.5. In the following, unless explicitly stated otherwise, all CPU times refer to the reference machine.

The performance results for PLS on Benchmark–1 are shown in Table 1. To generate these results, 100 independent trials were performed for each instance using target clique sizes corresponding to those obtained in [4]. As shown, PLS achieved a 100% success rate on all Benchmark-1 instances while using considerably less processor time than that required in [4].

The performance of PLS for Benchmark–2 is shown in Table 2. To generate these results, an extensive trial was first performed to identify the putative maximum clique size for each benchmark instance. Using the putative maximum clique size obtained for each instance, 100 independent trials of PLS were performed using a $maxSelections$ of $100\,000\,000$ to obtain the results shown in Table 2.

Figure 1 is an undirected graph showing the 0–1 contact maps for the 1KDI and 1PLA proteins and also the alignments (dotted lines) obtained by locating the max-

[1] http://www.cs.nott.ac.uk/ nxk/USM/protocol.html

[2] *dmclique*, ftp://dimacs.rutgers.edu in directory /pub/dsj/clique

imum clique within the 1KDI–1PLA correspondence graph. The consistency of the alignments can be verified by the observation that there are no intersections between any alignment lines.

Problem Instance	G Vertices	G Edges	Success Rate	Max. Clique	PLS CPU(s)	SCPU(s)
1BPI-1KNT	2 279	385 009	100	31	0.0469	1.52
1BPI-2KNT	2 436	446 657	100	29	0.0574	14.56
1BPI-5PTI	3 016	698 195	100	42	0.0299	2.4
1KNT-1BPI	2 494	462 092	100	30	0.0959	8.8
1KNT-2KNT	1 806	240 521	100	39	0.0098	0
1KNT-5PTI	2 236	378 609	100	28	0.0353	3.68
1VII-1CPH	171	1 581	100	6	0.0001	0
2KNT-5PTI	2 184	364 315	100	28	0.0267	7.6
3EBX-1ERA	2 548	477 720	100	31	0.1257	18.88
3EBX-6EBX	1 768	225 761	100	28	0.0163	0.48
6EBX-1ERA	1 666	199 074	100	20	0.0169	8.08

Table 1. PLS performance results, averaged over 100 independent trials, for the benchmark instances from [4]. The maximum known clique size, for each instance, is shown in the 'Max. Clique' column; CPU(s) is the PLS run-time in CPU seconds, averaged over all successful trials, for each instance. 'SCPU(s)' is the CPU time reported in [4], scaled by 0.08 to allow some basis for comparison with the reference computer used in this study.

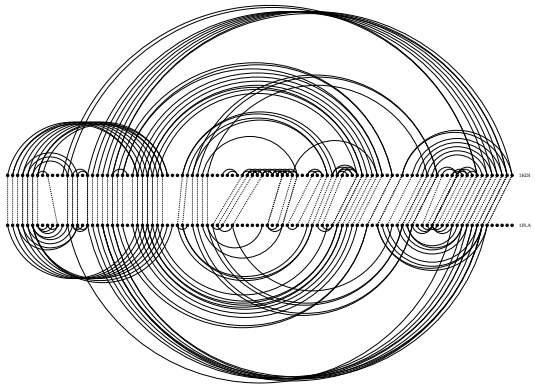

Fig. 1. Undirected graph representation of the 0–1 contact maps and putative maximal alignments for the 1KDI and 1PLA proteins. The vertices (dots) represent the residues of each protein, the solid edges (arcs) the contacts within each protein and the dashed edges show alignments identified by finding the maximum clique in the correspondence graph.

4. Conclusions and Future Work

The overall performance of PLS on the CMO instances reported here suggests that the underlying dynamic local search method has substantial potential to provide the basis for high-performance algorithms for other optimisation problems.

Problem Instance	G Vertices	G Edges	Success Rate	Clique Max.	Clique Avg.	Clique Min.	CPU(s)	Sels.	Sels. / Sec.
1A8O–1F22	2 728	1 063 344	100	25	25	25	30.52	1 412 602	46 278
1AVY–1BCT	6 278	6 842 400	99	50	49.99	49	630.30	14 726 972	23 365
1B6W–1BW5	4 131	3 095 143	100	34	34	34	84.09	2 977 231	35 407
1BAW–2B3I	7 200	5 519 222	100	53	53	53	28.36	488 140	17 215
1BCT–1BW5	4 386	3 277 240	77	36	35.77	35	1 117.91	36 305 118	32 476
1BCT–1F22	3 784	2 101 393	100	25	25	25	20.38	695 072	34 111
1BCT–1ILP	4 988	3 866 071	100	30	30	30	0.36	10 465	28 941
1BPI–2KNT	1 848	362 922	100	32	32	32	0.28	18 291	64 976
1C7V–1C7W	2 401	886 821	100	34	34	34	17.70	971 361	54 871
1C9O–1KDF	2 805	729 697	100	21	21	21	0.45	18 950	41 775
1DF5–1F22	3 960	2 462 200	100	27	27	27	46.27	1 513 993	32 718
1KDI–1BAW	7 920	6 774 365	100	53	53	53	610.03	9 700 460	15 901
1KDI–1PLA	6 424	4 392 217	100	53	53	53	369.05	7 053 785	19 113
1KDI–2B3I	7 040	5 246 576	100	47	47	47	679.82	11 884 684	17 482
1KDI–2PCY	7 216	5 583 059	98	57	56.98	56	1543.28	26 656 690	17 272
1NMF–2NEW	2 728	800 896	100	21	21	21	55.00	2 272 339	41 313
1NMG–1WDC	4 698	2 754 536	100	17	17	17	0.98	26 320	26 928
1PFN–1SVF	5 992	5 197 600	100	30	30	30	83.02	1 922 495	23 156
1PLA–1BAW	6 570	4 512 505	100	55	55	55	368.79	6 860 641	18 603
1PLA–2B3I	5 840	3 510 020	100	47	47	47	80.16	1 646 885	20 544
1PLA–2PCY	5 986	3 725 919	100	57	57	57	218.30	4 381 399	20 070
1VII–1CPH	903	99 638	100	15	15	15	0.024	3 121	128 429
1VNB–1BHB	6 120	4 011 048	100	28	28	28	138.98	2 786 011	20 046
2KNT–1KNT	1 980	402 659	100	41	41	41	0.06	3 800	64 083
2NEW–3MEF	2 552	631 920	100	16	16	16	0.14	6 248	43 357
2PCY–1BAW	7 380	5 769 409	100	66	66	66	535.34	8 920 572	16 663
2PCY–2B3I	6 560	4 475 832	100	52	52	52	136.20	2 524 044	18 532
3EBX–1ERA	2 205	356 245	100	19	19	19	0.04	2 279	51 444
3EBX–6EBX	2 331	461 771	100	25	25	25	0.51	24 855	48 735
5PTI–1BPI	1 596	285 692	100	35	35	35	0.17	12 602	75 733
5PTI–1KNT	1 710	303 273	100	31	31	31	0.063	4 505	71 508
5PTI–2KNT	1 672	290 649	100	32	32	32	0.63	44 616	70 339
6EBX–1ERA	1 295	168 119	100	22	22	22	0.02	1 986	92 373

Table 2. PLS performance results, averaged over 100 independent trials, for the PDB protein pairs in Benchmark-2. 'G' is the correspondence graph for each protein pair, the sizes found for each maximum clique are shown as maximum, average and minimum found over the 100 trials while 'Sels.' is the average number of vertices that were added to the clique over the 100 trials.

References

1. Lesk, A.M., (2001) Introduction to Protein Architecture. Oxford University Press. Oxford, UK.
2. Godzik, A., Skolnick, J., Kolinski, A., (1993) Regularities in interaction patterns of globular proteins. Protein Eng. 6, 801–810.
3. Goldman, D., Istrail, S., Papadimitriou, C., (1999) Algorithmic aspects of protein structure similarity. Proc. 40th Annual IEEE Sympos. Foundations Comput. Sci., IEEE Computer Society, 512–522.
4. Strickland, D.M., Barnes, E., Sokol, J.S., (2005) Optimal protein structure alignment using maximum cliques. Operations Research, 53, 389-402.
5. Garey, M.R., Johnson, D.S., (1979) Computer Intractability: A Guide to the Theory of $\mathcal{NP}$-Completeness.
6. Pullan, W.J., (2006) Phased local search for the maximum clique problem. Journal of Combinatorial Optimization, 12(3) 303-323.
7. Berman, H.M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T.N., Weissing, H., Shindyalov, I.N., Bourne, P.E., (2000) The protein data bank. Nucleic Acids Research, 28, 235-242.
8. Krasnogor, N., Pelta, D.A., (2004) Measuring the similarity of protein structures by means of the universal similarity metric. Bioinformatics, 20, 1015-1021.