Supplementary material for SAGA: A subgraph matching tool for biological graphs

1 EXAMPLES ILLUSTRATING THE SAGA ALGORITHM

This section presents some detailed examples of the SAGA subgraph matching algorithm.

1.1 Example *FragmentIndex*

The index structure is discussed in detail in Section 2.3.1 of the paper. A sample *FragmentIndex*, with $k = 3$ and $d_{max} = 2$ for the database shown in Figure 1, is presented in Figure 2. In this index, the $groupSeq$'s are ordered by the group IDs, and the $nodeSeq$'s are ordered according to the $groupSeq$'s. If u, v, w is the $nodeSeq$, then the corresponding $distSeq$ is $d(u, v)$, $d(u, w)$, $d(v, w)$. Note that node v_8 with the label L_8 in G_1 belongs to two groups B and D, thus for this node set $\{v_1, v_3, v_8\}$, there are two index entries $\{(B, E, E), G_1, (v_8, v_1, v_3), (1, 2, 2), 5\}$ and $\{(D, E, E), G_1, (v_8, v_1, v_3), (1, 2, 2), 5\}$ in the index.

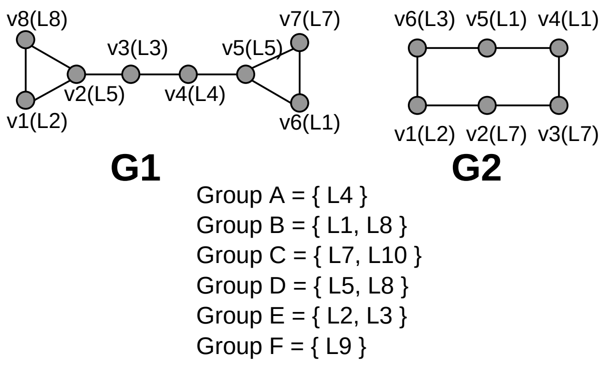

Fig. 1. Example database graphs

Fragment Index

Group Seq	Graph ID	Node Seq	Distance Seq	DistSum
A,B,C	G1	v4,v6,v7	2,2,1	5
……	……	……	……	……
A,C,D	G1	v4,v7,v2	2,2,4	8
		v4,v7,v5	2,1,1	4
……	……	……	……	……
B,E,E	G1	v8,v1,v3	1,2,2	5
	G2	v4,v1,v6	3,2,1	6
		v5,v1,v6	2,1,1	4
……	……	……	……	……
D,E,E	G1	v2,v1,v3	1,1,2	4
		v5,v1,v3	4,2,2	8
		v8,v1,v3	1,2,2	5

Fig. 2. The *FragmentIndex* for the example database

1.2 Example of step 2 of the matching algorithm

As an example of the second step of the SAGA matching algorithm (refer to Section 2.3.2 in the paper), Figure 3(b) shows the hit-compatible graph for the database graph G_1 in Figure 1 when querying Q in Figure 3(a), with $MaxPairDist = 1$. The nodes in the hit-compatible graph are denoted by rectangles. Two maximal cliques (shown as dotted circles) are detected in this hit-compatible graph. Therefore, this step produces two candidate matches in G_1 for query Q, namely $Q(v_1, v_2, v_3, v_4) \leftrightarrow G_1(v_4, v_5, v_7, v_6)$, and $Q(v_1, v_2, v_4) \leftrightarrow G_1(v_4, v_2, v_8)$.

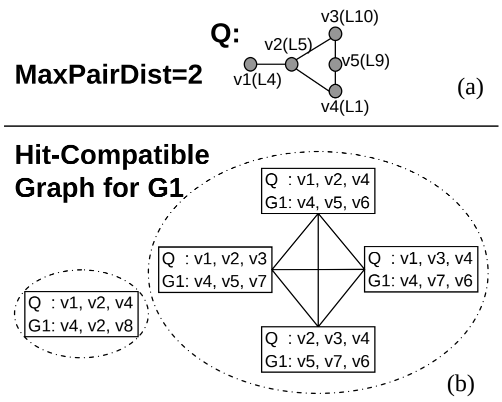

Fig. 3. (a) An example query Q (b)The hit-compatible graph for G_1 when querying Q.

2 STEP 1 DETAILS OF THE MATCHING ALGORITHM

In step 1 of the matching algorithm (Section 2.3.2), we probe the $FragmentIndex$ to find matching database fragments for the query fragments. We use the $groupSeq$ and $sumDist$ attributes to search the index. In order to match a query fragment, the database fragments must have the same $groupSeq$ value. And we also develop safe bounds for the $sumDist$ attribute as follows: Suppose that q is the query and p is a database graph. From the structure similarity restriction defined in Section 2.3.2 of the paper , we get the following inequality: $\sum_{u,v\in\hat{V}_q,u<v} |d_q(u,v) - d_p(\lambda u, \lambda v)| \leqslant \frac{k(k-1)}{2} \times \frac{MaxPairDist}{w_e}$, where k is the fragment size. In addition, we have the following trivial inequality:

$$|\sum_{u,v\in\hat{V}_q,u<v} d_q(u,v) - \sum_{u,v\in\hat{V}_q,u<v} d_p(\lambda u,\lambda v)| \leqslant \sum_{u,v\in\hat{V}_q,u<v} |d_q(u,v) - d_p(\lambda u,\lambda v)|$$

Using the two inequalities above, we can conclude that a database fragment f_d cannot match the query fragment f_q, if $|f_d.\textit{sumDist} - f_q.\textit{sumDist}| > \frac{k(k-1)}{2} \times \frac{MaxPairDist}{w_e}$. In other words, when probing the *FragmentIndex* in the first level of filtering, we only fetch the database fragments $\{t \mid t \in \textit{FragmentIndex}, t.groupSeq = f_q.groupSeq, f_q.sumDist - \frac{k(k-1)}{2} \times \frac{MaxPairDist}{w_e} \leqslant t.sumDist \leqslant f_q.sumDist + \frac{k(k-1)}{2} \times \frac{MaxPairDist}{w_e}\}$.

The probing condition above includes an equality search and a range search. It imposes several optimization opportunities. First, to reduce the IO costs, we can group all the probes by the *groupSeq*. A good way of implementing the *FragmentIndex* is to order the physical layout of the index by *groupSeq* and *sumDist* attributes. Then, probes with the same *groupSeq* have very high spatial locality, which reduces the number of random IOs that are incurred during the index probes. In addition, for each group of probes with the same *groupSeq* value, we can optimize the range query scans on the *sumDist* attribute. Essentially, if query ranges overlap, a query can be issued with the union of the ranges rather than several overlapping individual queries, which further reduces the IO cost.

It is possible that more than one node in a fragment has the same group label. To correctly handle this case, we simply expand the query probe set to include a probe set for every possible node sequence for the same group sequence.

3 DETAILS FOR QUERYING DISEASE-ASSOCIATED PATHWAYS

The entire list of 162 KEGG human pathways that we used in our evaluation is: hsa00010, hsa00020, hsa00030, hsa00031, hsa00040, hsa00051, hsa00052, hsa00053, hsa00061, hsa00062, hsa00071, hsa00072, hsa00100, hsa00120, hsa00130, hsa00140, hsa00150, hsa00190, hsa00193, hsa00220, hsa00230, hsa00240, hsa00251, hsa00252, hsa00260, hsa00271, hsa00272, hsa00280, hsa00290, hsa00300, hsa00310, hsa00330, hsa00340, hsa00350, hsa00351, hsa00360, hsa00361, hsa00362, hsa00363, hsa00380, hsa00400, hsa00401, hsa00410, hsa00430, hsa00440, hsa00450, hsa00460, hsa00471, hsa00472, hsa00480, hsa00500, hsa00510, hsa00512, hsa00520, hsa00521, hsa00530, hsa00531, hsa00532, hsa00534, hsa00550, hsa00561, hsa00562, hsa00563, hsa00564, hsa00590, hsa00591, hsa00600, hsa00601, hsa00602, hsa00603, hsa00604, hsa00620, hsa00623, hsa00624, hsa00625, hsa00626, hsa00628, hsa00629, hsa00630, hsa00632, hsa00640, hsa00642, hsa00643, hsa00650, hsa00660, hsa00670, hsa00680, hsa00710, hsa00720, hsa00730, hsa00740, hsa00750, hsa00760, hsa00770, hsa00780, hsa00790, hsa00791, hsa00830, hsa00860, hsa00900, hsa00902, hsa00903, hsa00904, hsa00910, hsa00920, hsa00930, hsa00940, hsa00950, hsa00960, hsa00970, hsa00980, hsa01510, hsa04010, hsa04020, hsa04060, hsa04070, hsa04080, hsa04110, hsa04120, hsa04130, hsa04140, hsa04150, hsa04210, hsa04310, hsa04330, hsa04340, hsa04350, hsa04360, hsa04370, hsa04510, hsa04512, hsa04514, hsa04520, hsa04530, hsa04540, hsa04610, hsa04612, hsa04620, hsa04630, hsa04650, hsa04660, hsa04662, hsa04664, hsa04670, hsa04710, hsa04720, hsa04730, hsa04740, hsa04742, hsa04810, hsa04910, hsa04920, hsa04930, hsa04940, hsa04950, hsa05010, hsa05020, hsa05030, hsa05040, hsa05050, hsa05060, hsa05120.

The list of the ten disease pathways that were used as queries in Section 3.2.1 is shown in Table 1. A clickable graphical representation of the matches is available at `http://enigma.eecs.umich.edu/kegg_result/kegg_disease_result.html`.

Category	KEGG ID	Pathway	# nodes	# edges
Metabolic Disorders	hsa04930	Type II diabetes mellitus	33	36
	hsa04940	Type I diabetes mellitus	22	2
	hsa04950	Maturity onset diabetes of the young	34	33
Neuro-degenerative Disorders	hsa05010	Alzheimer’s disease	23	17
	hsa05020	Parkinson’s disease	19	10
	hsa05030	Amyotrophic lateral disease	24	13
	hsa05040	Huntington’s disease	24	28
	hsa05050	Dentatorubropallidoluysian atrophy	8	10
	hsa05060	Prion disease	11	15
Infectious Disease	hsa05120	Epithelial cell signaling in Helicobacter pylori infection	57	26

Table 1. The ten disease associated human pathways in KEGG