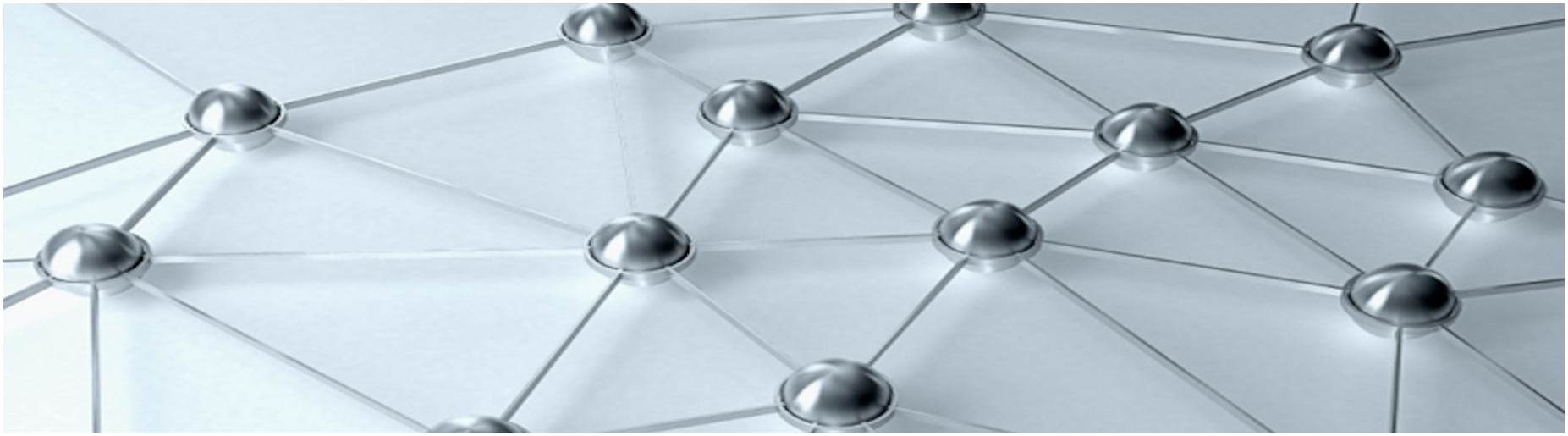


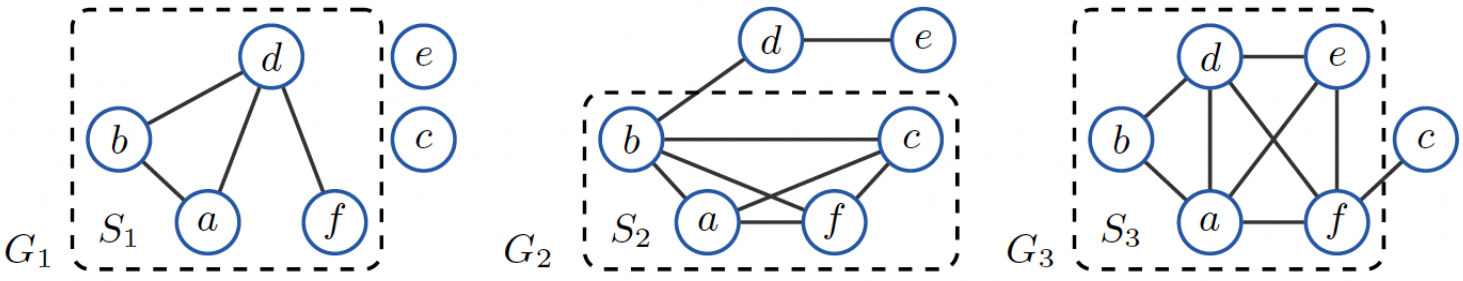
Jaccard-constrained dense subgraph discovery

Chamalee Wickrama Arachchi, Nikolaj Tatti

University of Helsinki, Finland

DS23

What do we do?



- Three graph snapshots : G_1, G_2, and G_3
- Each graph snapshot contains 6 vertices, different edge sets.
- Three subgraphs : S_1, S_2, and S_3.

Jaccard-constrained dense subgraph discovery problem

Input:

- A sequence of graph snapshots.
- Weight parameter : λ.

Output:

- A sequence of subgraphs.

Such that:

- (Sum of the densities $+$ $\lambda\times$ sum of pairwise similarities) is maximized.

Densest Subgraph Problem

- Definition of density:

$$d(S) = \frac{|E(S)|}{|V(S)|}$$

- Densest subgraph problem : Find a subset of vertices which maximizes the density.
- Exact[1] and greedy[2] algorithms.

[1] Andrew V Goldberg. Finding a maximum density subgraph. 1984.

[2] Moses Charikar. Greedy approximation algorithms for finding dense components in a graph. In International workshop on approximation algorithms for combinatorial optimization, pages 84–95. Springer, 2000.

Densest Common Subgraph Problem

- Extension for multiple graph snapshots.
- DCS : Find a common set of vertices which maximizes sum of the densities.
- Variants of objectives :
  - maximize sum of the densities [3].
  - maximize min density [4].

[3] Semertzidis, K., Pitoura, E., Terzi, E. and Tsaparas, P., 2019. Finding lasting dense subgraphs. Data Mining and Knowledge Discovery, 33(5), pp.1417-1445.

[4] Vinay Jethava and Niko Beerenwinkel. Finding dense subgraphs in relational graphs. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pages 641–654. Springer, 2015.

Jaccard coefficient

- Similarity between two sets.
- S_i : subset of vertices from ith graph snapshot.
- S_j : subset of vertices from jth graph snapshot.

$$J(S_i, S_j) = \frac{|S_i \cap S_j|}{|S_i \cup S_j|}$$

- When $S_i = S_j$, pairwise Jaccard coefficient is one.

Definitions and our objective

- Density of ith subgraph

$$d(S_i) = \frac{|E(S_i)|}{|S_i|} \quad .$$

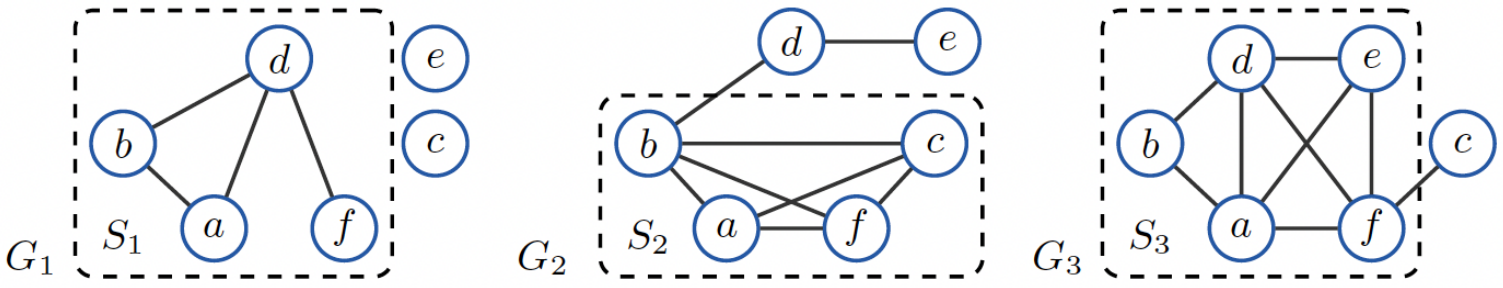
- k : Length of the graph sequence
- Sum of the densities from all graph snapshots

$$d(\mathcal{S}) = \sum_{i=1}^{k} d(S_i) \quad .$$

- Objective

$$q(\mathcal{S}; \lambda) = d(\mathcal{S}) + \lambda \sum_{i=1}^{k} \sum_{j=i+1}^{k} J(S_i, S_j) \quad .$$

Example



- Let $\lambda = 0.3$, sets: $S_1 = \{a, b, d, f\}$, $S_2 = \{a, b, c, f\}$, and $S_3 = \{a, b, d, e, f\}$.
- The sum of densities is $\frac{4}{4} + \frac{6}{4} + \frac{8}{5} = 4.1$.
- $J(S_1, S_2) = \frac{3}{5} = 0.6$, $J(S_1, S_3) = \frac{4}{5} = 0.8$, and $J(S_2, S_3) = \frac{3}{6} = 0.5$.
- Our objective is equal to $4.1 + 0.3 \times (0.6 + 0.8 + 0.5) = 4.67$.

Our problem

- Given a graph sequence $= \{G_1, \ldots, G_k\}$, with $G_i = (V, E_i)$,
  and a real number λ,
  find a collection of subset of vertices $\mathcal{S} = \{S_1, \ldots, S_k\}$,
  such that $q(\mathcal{S})$ is maximized.

- It turns out that finding subgraphs maximizing our objective is **NP**-hard.

Algorithm - 1

Algorithm 1: ITR$(\mathcal{G}, \lambda, \mathcal{S})$, finds subgraphs with good $q(\cdot; \lambda)$

```
while changes do
    foreach i = 1, ..., k do
        C ← V;
        foreach j = 2, ..., |V| do
            u ← arg max_{v∈C} q(S_1, ..., S_{i−1}, C \ {v}, S_{i+1}, ..., S_k);
            C ← C \ {u};
        S_i ← best tested C, if the score improves;
return S;
```

Figure: Iterative algorithm

- Initialization : DCS and individual densest subgraphs.

Algorithm - 2

Algorithm 2: GRD$(\mathcal{G}, \lambda)$, finds subgraphs with good $q(\cdot; \lambda)$

$\mathcal{S} \leftarrow S_1, \ldots, S_k$, where $S_i = V$;
while there are nodes **do**
    $u, j \leftarrow \arg \max_{v,i|v \in S_i} q(S_1, \ldots, S_{i-1}, S_i \setminus \{v\}, S_{i+1}, \ldots, S_k)$;
    $S_j \leftarrow S_j \setminus \{u\}$;
return best tested $\mathcal{S}$;

Figure: Greedy algorithm

Experiments with Synthetic Datasets

- $|V^d|$ and $|V^s|$: Initial number of dense and sparse vertices.
- $|E|$: The expected number of edges.
- k : The number of snapshots.
- p_d, p_s, and p_c : The dense, sparse, and cross edge probabilities.

Dataset	$\|V^d\|$	$\|V^s\|$	$\mathrm{E}[\|E\|]$	k	p_d	p_s	p_c	d_{true}	d_{dcs}	d_{ind}	J_{min}
Syn-1	100	900	672.8	10	0.05	0.0005	0.0005	34.68	24.26	35.74	0.47
Syn-2	100	5 000	3 850.2	5	0.05	0.0001	0.0001	40.47	13.03	40.57	0.18
Syn-3	120	1 200	3 922	5	0.06	0.005	0.002	27.4	17.62	27.62	0.45
Syn-4	250	5 000	4 709	8	0.03	0.001	0.001	55.79	29.65	55.98	0.27
Syn-5	500	3 500	12 136.29	7	0.05	0.0003	0.0003	112.11	87.27	112.12	0.53
Syn-6	350	3 500	32 015	5	0.06	0.005	0.002	67.96	52.17	67.96	0.49

- d_{dcs} : The density of densest common subgraph.
- d_{ind} : The sum of densities of locally densest subgraph from each graph snapshot.
- $J_{min} = \min_{i<j} J(V_i^d, V_j^d)$: The minimum Jaccard index between ground truth dense sets of vertices.
- d_{true} : The ground truth density of dense components.

- d_{dis} and q : The sum of densities and scores of the discovered sets.
- i : The number of iterations using Algorithm-1.
- ρ : The average Jaccard index between discovered and ground truth sets.
- $time$: The computational time in seconds.

		Itr						Grd				
Data	λ	d_{dis}	q	J_{min}	ρ	$time$	i	d_{dis}	q	J_{min}	ρ	$time$
Syn-1	0.3	34.84	43.1	0.51	0.93	10	4	34.84	43.09	0.5	0.93	30
	0.5	33.54	48.97	0.54	0.89	11	5	33.44	48.97	0.54	0.88	33
	0.7	26.43	56.08	0.84	0.74	4	2	26.22	56.08	0.86	0.73	32
Syn-2	0.3	40.57	41.18	0.18	0.99	50	3	40.57	40.77	0.18	0.99	92
	0.4	40.57	41.39	0.18	0.99	53	3	40.57	41.39	0.18	0.99	84
	0.5	40.57	41.6	0.18	0.99	50	3	40.57	41.6	0.18	0.99	87
Syn-3	0.4	27.59	29.59	0.44	0.97	16	3	27.6	29.59	0.44	0.97	29
	0.5	27.59	30.1	0.44	0.97	18	3	27.6	30.08	0.44	0.97	26
	0.8	27.44	31.65	0.46	0.98	17	3	27.6	31.56	0.43	0.98	28
Syn-4	0.4	55.75	60.38	0.27	0.98	66	2	55.89	60.42	0.27	0.98	307
	0.5	55.82	61.56	0.27	0.98	132	4	55.83	61.55	0.27	0.98	308
	0.6	55.76	62.71	0.27	0.98	132	4	55.79	62.71	0.27	0.98	285
Syn-5	0.01	112.12	112.26	0.53	1	73	2	112.12	112.26	0.53	1	249
	0.4	112.12	117.7	0.53	1	69	2	112.12	116.3	0.53	1	251
	0.5	112.12	119.09	0.53	1	103	3	112.12	119.09	0.53	1	257
Syn-6	0.1	67.96	68.61	0.49	1	42	2	67.96	68.61	0.49	1	226
	0.8	67.96	73.18	0.49	1	67	3	67.96	73.17	0.49	1	252

Experiments with Real-world Datasets

- $|V|$: The number of vertices.
- $|E|$: The number of edges.
- k : The number of snapshots.
- d_{dcs} : The density of densest common subgraph.
- d_{ind} : The sum of densities of locally densest subgraph from each graph snapshot.

Data	$\|V\|$	$\|E\|$	k	d_{dcs}	d_{ind}
Twitter-#	806	101.2	15	7.54	38.79
Enron	1 079	23.2	183	43.47	185.34
Facebook	4 117	83.13	104	14	88.65
Students	889	43.68	122	24.15	117.98
Twitter-user	4 605	109.19	93	23	90.63
Tumblr	1 980	65.3	89	36.67	103.98

Experiments with Real-world Datasets

- d_{dis} and q : The sum of densities and scores of the discovered sets.
- i : The number of iterations using Algorithm-1.
- $time$: The computational time in seconds.

		Itr				Grd		
Data	λ	d_{dis}	q	*time*	i	d_{dis}	q	*time*
Twitter-#	0.3	37.34	40.52	2.81	5	37.65	38.13	5.79
	0.5	14.14	54.54	3.06	7	31.01	42.37	6.38
	0.7	11	74.7	2.62	5	16.8	56.18	6.23
	0.8	11	83.8	2.79	6	9.33	75.47	6.88
Enron	0.05	130.43	357.29	72.02	5	122.08	343.02	816.54
	0.1	111.88	593.87	148.9	10	62.08	589.96	826.28
	0.5	95.88	2 662.7	81.56	6	20	3 139.25	854.43
	5	88.66	25 254.81	98.87	7	20	31 212.5	774.39
Facebook	0.1	52.86	126.98	214.74	7	56.03	85.64	9 461.59
	0.5	36.4	446.09	148.94	6	25.47	344.6	10 193.96
	0.7	31.25	657.92	185.91	7	23.41	475.83	9 781.9
	1	27.41	922.45	136.72	6	22.48	659.32	9 034.35
Students	0	108.9	108.9	124.38	4	102.7	102.7	2 787.8
	0.2	46.99	526.19	100.17	5	33.17	567.84	2 178.28
	0.5	46.17	1 258.41	79.65	4	33.17	1 369.86	2 256.26
	0.8	44.5	2 020.81	78.61	4	33.17	2 171.88	2 125.67
Twitter-user	0.01	79.21	81.61	572.25	8	63.53	68.53	7 524.47
	0.1	12.04	269	260.31	9	13.9	212.28	6 422.46
	0.2	11.49	545.93	149.91	5	11.64	423.1	6 770.16
	0.5	11.49	1 347.59	150.08	5	10.83	1 040.05	6 847.68
Tumblr	0.1	71.33	316.07	50.23	4	66.47	302.83	1 229.98
	0.5	64.37	1 456.47	49.31	4	62.97	1 291.82	1 151.85
	0.7	59.25	2 182.29	48.96	4	62.97	1 783.35	1 217.76

Conclusion

- Introduced a novel Jaccard weighted, dense subgraph discovery problem for graphs with multiple snapshots..
- Our goal was to find a dense subset of vertices from each graph snapshot such that the sum of densities and the similarity between the snapshots is maximized.
- Proved that our problem is **NP**-hard.
- Designed an iterative algorithm and a greedy algorithm which runs in polynomial time.
- Showed experimentally that the algorithms could find the ground truth using synthetic datasets.
- Showed experimentally that the number of iterations was low in iterative algorithm.
- Discovered dense collections in real world datasets and computational time is reasonable.

Thank you for your attention!!!

Speedup tricks

- The bottleneck in both algorithms : Finding the next vertex to delete.
- Select S_i, let $v \in S_i$ and $\mathcal{S}'$:- $\mathcal{S}$ with S_i replaced with $S_i \setminus \{v\}$.
- The score difference between $\mathcal{S}$ and $\mathcal{S}'$:

$$q(\mathcal{S}') - q(\mathcal{S}) = \frac{|E(S_i)| - \deg v}{|S_i| - 1} - \frac{|E(S_i)|}{|S_i|} + \lambda \sum_{j \neq i} J(S_i \setminus \{v\}, S_j) - J(S_i, S_j) \quad .$$

- In Alg-1 : To find the optimal v and i, we group the nodes in S_i such that the second term in objective is equal for the nodes in the same group.
- Maintain nodes in a collection of a priority queues keyed by its degree.
- To maintain the difference of the Jaccard indices, we maintain the sizes of intersection $|S_i \cap S_j|$ and the union $|S_i \cup S_j|$ for all i and j.
- To find the optimal v and i, we find the vertex with the smallest degree in each group, and then compare these candidates among different groups.