Fast Searching The Densest Subgraph And Decomposition With Local Optimality

Yugao Zhu[1], Shenghua Liu[1], Wenjie Feng[2], Xueqi Cheng[1]

[1]*Institute of Computing Technology, Chinese Academic of Sciences, Beijing, China*
[2]*Institute of Data Science, National University of Singapore, Singapore*
zhuyugao22@mails.ucas.ac.cn, liushenghua@ict.ac.cn, wenjie.feng@nus.edu.sg, cxq@ict.ac.cn

***Abstract*—Densest Subgraph Problem (DSP) is an important primitive problem with a wide range of applications, including fraud detection, community detection, and DNA motif discovery. Edge-based density is one of the most common metrics in DSP. Although a maximum flow algorithm can exactly solve it in polynomial time, the increasing amount of data and the high complexity of algorithms motivate scientists to find approximation algorithms. Among these, its duality of linear programming derives several iterative algorithms including Greedy++, Frank-Wolfe, and FISTA which redistribute edge weights to find the densest subgraph, however, these iterative algorithms vibrate around the optimal solution, which is not satisfactory for fast convergence. We propose our main algorithm Locally Optimal Weight Distribution (LOWD) to distribute the remaining edge weights in a locally optimal operation to converge to the optimal solution monotonically. Theoretically, we show that it will reach the optimal state of specific quadratic programming, which is called locally-dense decomposition. Besides, we show that it is not necessary to consider most of the edges in the original graph. Therefore, we develop a pruning algorithm using a modified Counting Sort to prune graphs by removing unnecessary edges and nodes, and then we can search the densest subgraph in a much smaller graph.**

I. FULL VERSION

Proof of Corollary 1. For DkS, we set $|\mathcal{S}| = k$, it has an upper bound as follows:

$$\begin{aligned}\rho(\mathcal{S}) &= \frac{\sum_{e\in\mathcal{E}(\mathcal{S})} w_e}{|\mathcal{S}|} = \frac{\sum_{e=(u,v),\, u,v\in\mathcal{S}} f_e(u) + f_e(v)}{|\mathcal{S}|} \\ &\le \frac{\sum_{e=(u,v),\, u,v\in\mathcal{S}} (f_e(u) + f_e(v)) + \sum_{e=(u,v),\, u\in\mathcal{S}, v\notin\mathcal{S}} f_e(u)}{|\mathcal{S}|} \\ &= \frac{\sum_{u\in\mathcal{S}} \sum_{e\ni u} f_e(u)}{|\mathcal{S}|} = \frac{\sum_{u\in\mathcal{S}} l_u}{|\mathcal{S}|} \\ &= \frac{\sum_{i=0}^{j-1} \lambda_i * |B_i| + (k - |B_{j-1}|) * \lambda_j}{k}\end{aligned}$$

Therefore, result 2 holds up. As for result 1, if $k = |B_j|$, according to property 3 the equivalency condition in line 2 holds up because $f_e(u) = 0$ if $u \in \mathcal{S}$ and $v \notin \mathcal{S}$, then result 1 holds up. These results also hold up for DalkS because if $|S| > k$, it will add more nodes into $\mathcal{S}$ with lower upper bounds on their loads. □

Before proofs of Lemma 2, we introduce the definition of k-core from [53]: k-core is the maximal subgraph G_k in graph G, the degree of where any vertex v in G_k is satisfied with $\mathrm{d}_{G_k}(v) \ge k$.

proof of Lemma 2. The Greedy algorithm just moves any node whose degree is the lowest in the remaining graph $\mathcal{H}$. Let us remark $A(k)$ as the nodeset of k-core for specific k. We claim that when deleting a node $u \in A(k)$, there mustn't be any node $v \notin A(k)$ in the remaining graph $\mathcal{H}$. We prove it by way of contradiction, w.l.o.g, we set u as the first node to be deleted in $A(k)$ in Greedy, therefore u has the lowest degree in $\mathcal{H}$ and $\mathrm{d}_{\mathcal{H}}(v) \ge \mathrm{d}_{\mathcal{H}}(u) \ge \mathrm{d}_{A(k)}(u) \ge k$ for any node v in $\mathcal{H}$, which produces a k-core subgraph with a larger size, and it leads to a contradiction. □

Proof of Theorem 5. We set $\mathcal{H}_k (k > 0)$ is the remaining graph after k iterations in Algorithm 2 and $\mathcal{H}_0$ is the initial whole graph, $\mathcal{H}'_{k+1}$ is the nodeset to be deleted in k+1 iteration. Therefore, $\mathcal{H}_{k+1} = \mathcal{H}_k \setminus \mathcal{H}'_{k+1}$,

$$\begin{aligned}\rho(\mathcal{H}_{k+1}) &= \rho(\mathcal{H}_k \setminus \mathcal{H}'_{k+1}) \\ &= \frac{\mathcal{W}(\mathcal{E}(\mathcal{H}_k)) - \mathcal{W}(\mathcal{E}(\mathcal{H}'_{k+1}))}{|\mathcal{H}_k| - |\mathcal{H}'_{k+1}|} \\ &\ge \frac{\rho(\mathcal{H}_k)\cdot|\mathcal{H}_k| - \sum_{v\in\mathcal{H}'_{k+1}} \mathrm{d}_{\mathcal{H}_k}(v)}{|\mathcal{H}_k| - |\mathcal{H}'_{k+1}|} \\ &> \frac{\rho(\mathcal{H}_k)\cdot|\mathcal{H}_k| - \sum_{v\in\mathcal{H}'_{k+1}} \rho(\mathcal{H}_k)}{|\mathcal{H}_k| - |\mathcal{H}'_{k+1}|} \\ &= \frac{\rho(\mathcal{H}_k)\cdot|\mathcal{H}_k| - \rho(\mathcal{H}_k)\cdot|\mathcal{H}'_{k+1}|}{|\mathcal{H}_k| - |\mathcal{H}'_{k+1}|} \\ &= \rho(\mathcal{H}_k)\end{aligned}$$

That means the density of graph $\mathcal{H}$ monotonically increases in iterations, then any deleted node has a lower degree (when it is being deleted) than the final density, i.e., δ. Therefore, the remaining graph $\mathcal{H}$ is a δ-core and it is the graph of some time of the greedy search according to Lemma 2.

We claim that the process before getting the δ-core is a monotonic increasing phase of density in Greedy. We can confirm two facts:

1. During Greedy, if there is a node $u \in \mathcal{H}'_1$ existing in the remaining graph, the deletion in Greedy will increase the density of the remaining graph. That's because if Greedy deletes a node $v \notin \mathcal{H}'_1$, then $\mathrm{d}_{\mathcal{H}}(v) \le \mathrm{d}_{\mathcal{H}}(u) < \rho(\mathcal{G}) \le \rho(\mathcal{H})$. $\rho(\mathcal{H})$ will increase and $\rho(\mathcal{G}) \le \rho(\mathcal{H})$ still holds up. If Greedy deletes the node u, now that $\mathrm{d}_{\mathcal{H}}(u) \le \rho(\mathcal{H})$, then $\rho(\mathcal{H})$ will also increase and $\rho(\mathcal{G}) \le \rho(\mathcal{H})$ holds up.

2. During Greedy, if there is a node $u \in \mathcal{H}_{k+1}^{'}$ existing in the remaining graph $\mathcal{H} > \mathcal{H}_k$, and there isn't any node belonging to $\mathcal{H}_k^{'}$. Then: $\rho(\mathcal{H}) \geq \rho(\mathcal{H}_k)$ because we delete more nodes with lower degrees. Therefore, when we deletes a node $v \notin \mathcal{H}_{k+1}^{'}$, then $\mathrm{d}_{\mathcal{H}}(v) \leq \mathrm{d}_{\mathcal{H}}(u) < \rho(\mathcal{G}) \leq \rho(\mathcal{H}_k) \leq \rho(\mathcal{H})$, then $\rho(\mathcal{H})$ will increase and $\rho(\mathcal{G}) \leq \rho(\mathcal{H})$ holds up. If Greedy deletes the node u, now that $\mathrm{d}_{\mathcal{H}}(v) \leq \rho(\mathcal{H})$, $\rho(\mathcal{H})$ will also increase and $\rho(\mathcal{G}) \leq \rho(\mathcal{H})$ holds up.

Therefore, the density monotonically increases in Greedy before getting δ-core. □

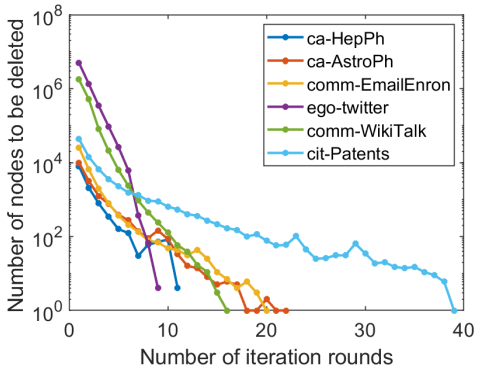


Fig. 2. Exponential decrease in the number of deleted nodes.

Algorithm 1: GREEDY DSPSOLVER

Input: Undirected graph $\mathcal{G}$; density metric $\rho(\cdot)$
Output: $\mathcal{S}^*$: the nodeset of the densest subgraph of $\mathcal{G}$.

$\mathcal{S}, \mathcal{S}^* \leftarrow \mathcal{V}$
while $\mathcal{S} \neq \emptyset$ **do**
  ▷ *find the vertex u^* with the lowest degree in $\mathcal{S}$*
  $u^* \leftarrow \arg\min_{u \in \mathcal{S}} \mathrm{d}_{\mathcal{S}}(u))$
  Remove u^* and all its adjacent edges from $\mathcal{G}$.
  ▷ *$\mathcal{S} \backslash \{u\}$: the remaining nodeset without u*
  $\mathcal{S} \leftarrow \mathcal{S} \backslash \{u\}$
  if $\rho(\mathcal{S}) > \rho(\mathcal{S}^*)$ **then**
    $\mathcal{S}^* \leftarrow \mathcal{S}$
return $\mathcal{S}^*$.

TABLE I
DATASET SOURCE AND DENSITY OF ALGORITHMS

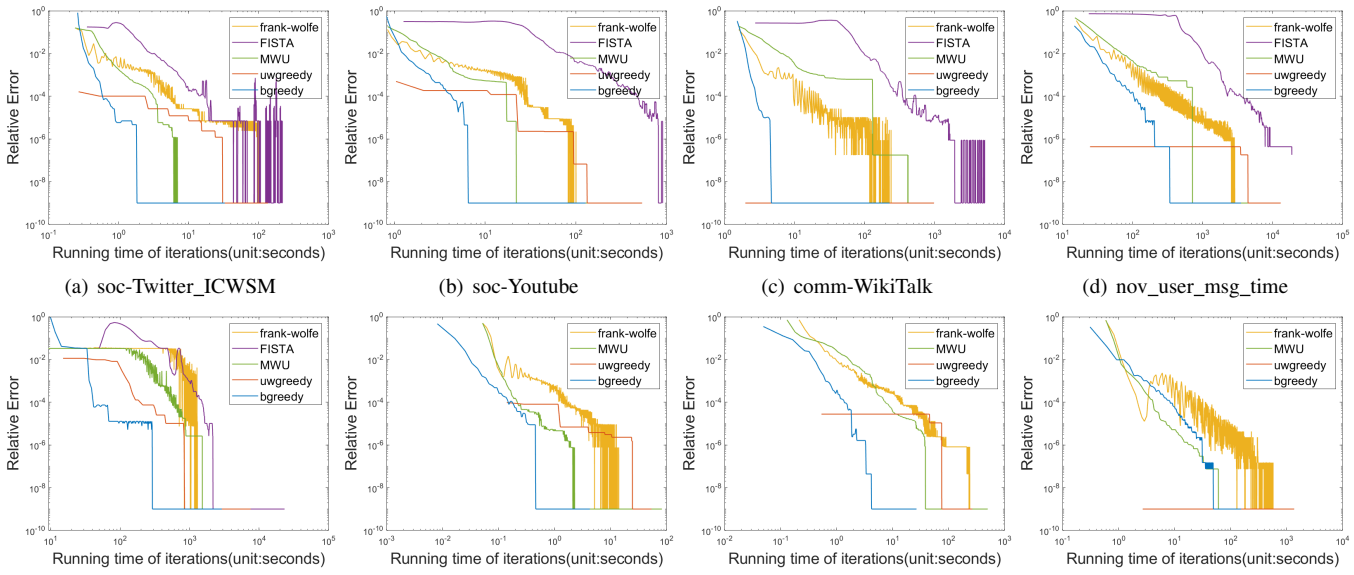
Dataset	Source	Type	Pruning	w_app	exact	DLL	uw_Pruning+DLL
ca-HepPh	Stanford's SNAP database	scholar collaboration network	119	119	119	119	119
comm-EmailEnron	Stanford's SNAP database	communication	37.316	37.344	37.344	37.337	37.337
ca-AstroPh	Stanford's SNAP database	scholar collaboration network	28.481	29.616	32.11	29.552	29.552
PP-Pathways	Stanford's SNAP database	protein interaction network	74.159	77.995	77.995	77.995	77.995
soc-Twitter_ICWSM	konect	social network	25.678	25.683	25.69	25.686	25.685
soc-sign_slashdot	Stanford's SNAP database	social network	39.376	42.132	42.132	42.132	42.132
rating-StackOverflow	konect	social network	20.209	20.209	20.21	20.209	20.209
soc-sign_epinion	Stanford's SNAP database	social network	80.168	85.599	85.637	85.589	85.589
ego-twitter	Stanford's SNAP database	social network	59.281	68.414	69.622	68.414	68.414
soc-Youtube	Stanford's SNAP database	social network	45.545	45.58	45.599	45.576	45.577
comm-WikiTalk	Stanford's SNAP database	communication	114.139	114.139	114.139	114.139	114.139
nov_user_msg_time	We own it privately.	social network	278.815	278.815	278.815	278.815	278.815
cit-Patents	AMiner scholar datasets	scholar collaboration network	132.776	135.706	137.261	135.706	135.706
soc-Twitter_ASU	ASU	social network	593.847	593.847	593.847	593.847	593.847
soc-Livejournal	Livejournal	social network	104.596	104.601	104.609	104.603	104.603
soc-Orkut	Stanford's SNAP database	social network	227.861	227.872	227.874	227.872	227.872
soc-SinaWeibo	Network Repository	social network	164.967	165.193	165.415	165.196	165.191
wang-tripadvisor	konect	rating network	13.442	13.873	14.082	–	–
rec-YelpUserBusiness	Network Repository	rating network	87.825	87.912	87.921	–	–
bookcrossing	konect	rating network	92.148	92.322	92.374	–	–
librec-ciaodvd-review	konect	rating network	233.553	233.59	233.597	–	–
movielens-10m	konect	rating network	1351.35	1351.35	1351.35	–	–
epinions	konect	rating network	595.302	595.314	595.316	–	–
libimseti	konect	social network	1645.71	1645.73	1645.73	–	–
rec-movielens	Network Repository	rating network	1801.16	1801.16	1801.16	–	–
yahoo-song	konect	rating network	46725.2	46725.2	46725.2	–	–

note: : **Pruning** (w_Pruning,uw_Pruning). **w_app**: approximation algorihtms on weighted graph(Priority Tree,Pruning+Priority Tree,BBST). **exact**: exact algorithms(maxflow,w_Pruning+maxflow). **DLL**:Doubly-linked list.

TABLE II
COMPARISON BETWEEN LOWD AND BASELINES WITHOUT THE PRUNING.

	Running time					The number of iteration rounds				
Dataset	**LOWD**	**Greedy++**	**FW**	**FISTA**	**MWU**	**LOWD**	**Greedy++**	**FW**	**FISTA**	**MWU**
ca-HepPh	0.0168	**0.0159**	0.0208	0.0411	0.0161	1	1	2	2	1
comm-EmailEnron	0.1516	**0.0714**	0.5551	4.9384	0.3012	19	2	88	255	46
ca-AstroPh	**0.3158**	0.8396	2.312	4.4843	0.6593	36	33	343	211	92
PP-Pathways	0.1174	**0.0333**	0.3214	2.8722	1.7086	9	1	28	79	164
soc-Twitter_ICWSM	**1.8343**	30.6654	96.3405	43.7426	6.1467	49	114	2816	401	175
soc-sign_slashdot	0.2275	**0.0704**	0.7881	9.033	1.1051	11	1	44	162	64
rating-StackOverflow	2.1887	**1.0537**	10.0728	227.1931	18.7819	32	2	178	1266	341
soc-sign_epinion	**0.3213**	1.4706	1.1853	8.0435	0.9581	9	12	40	90	33
ego-twitter	**1.6001**	2.9768	22.0971	51.1176	1.7266	34	21	550	384	41
soc-Youtube	**6.5253**	133.3086	82.3855	812.4268	22.0866	46	129	612	1792	161
comm-WikiTalk	4.6758	**1.984**	118.307	1942.7338	414.9261	15	1	407	2022	1386
nov_user_msg_time	**341.1144**	4464.4998	2581.9086	>16117.208	718.8805	97	174	624	>1200	177
cit-Patents_AMINER	**291.4452**	846.3628	837.8446	2172.7049	1523.2698	104	56	259	196	471
soc-Twitter_ASU	120.8906	**19.2912**	511.3170	15208.9590	2047.9598	40	1	156	1244	603
soc-Livejournal	–	–	–	–	–	–	–	–	–	–
soc-Orkut	–	–	–	–	–	–	–	–	–	–
soc-SinaWeibo_NETREP	–	–	–	–	–	–	–	–	–	–
wang-tripadvisor	**0.6686**	23.6761	30.209	–	15.6705	93	187	3894	–	2007
rec-YelpUserBusiness	**0.4621**	24.5879	5.1698	–	2.1192	54	228	626	–	239
bookcrossing	**1.5177**	92.9406	27.4916	–	8.877	78	259	1398	–	448
librec-ciaodvd-review	**4.2248**	75.7599	231.5822	–	38.6405	79	143	4635	–	773
movielens-10m	49.2341	**2.7029**	129.6344	–	60.719	159	1	450	–	211
epinions	**30.4071**	3957.2229	4302.242	–	743.6429	64	688	8893	–	1582
libimseti	**25.7101**	5276.5399	751.3750	–	146.2627	39	710	1196	–	234
rec-movielens	**50.4427**	6928.9143	306.0513	–	148.2539	54	733	355	–	172
yahoo-song	–	–	–	–	–	–	–	–	–	–

note: We ignore some datasets which are very large.">1200" and ">16117.208" means we run 1200 iterations(running time: 16117.208s) and can't still detect the densest subgraph.



(a) soc-Twitter_ICWSM (b) soc-Youtube (c) comm-WikiTalk (d) nov_user_msg_time

(e) cit-Patents_AMINER (f) rec-YelpUserBusiness (g) librec-ciaodvd-review (h) movielens-10m

Fig. 1. densest

REFERENCES

[1] H.-W. Shen and X.-Q. Cheng, "Spectral methods for the detection of network community structure: a comparative analysis," *JSTAT*, 2010.

[2] S. W. Wong, C. Pastrello, M. Kotlyar, C. Faloutsos, and I. Jurisica, "Sdregion: Fast spotting of changing communities in biological networks," in *SIGKDD'18*, 2018.

[3] Y. Liu, L. Zhu, P. Szekely, A. Galstyan, and D. Koutra, "Coupled clustering of time-series and networks," in *SDM*. SIAM, 2019.

[4] E. Cohen, E. Halperin, H. Kaplan, and U. Zwick, "Reachability and distance queries via 2-hop labels," *SIAM Journal on Computing*, vol. 32, no. 5, pp. 1338–1355, 2003.

[5] R. Jin, Y. Xiang, N. Ruan, and D. Fuhry, "3-hop: a high-compression indexing scheme for reachability query," in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, 2009, pp. 813–826.

[6] A. V. Goldberg, "Finding a maximum density subgraph," 1984.

[7] M. Charikar, "Greedy approximation algorithms for finding dense components in a graph," in *APPROX'00*, 2000.

[8] M. Danisch, T.-H. H. Chan, and M. Sozio, "Large scale density-friendly graph decomposition via convex programming," in *WWW*, 2017.

[9] S. Sawlani and J. Wang, "Near-optimal fully dynamic densest subgraph," in *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, 2020, pp. 181–193.

[10] D. Boob, Y. Gao, R. Peng, S. Sawlani, C. Tsourakakis, D. Wang, and J. Wang, "Flowless: Extracting densest subgraphs without flow computations," in *Proceedings of The Web Conference 2020*, 2020, pp. 573–583.

[11] C. Chekuri, K. Quanrud, and M. R. Torres, "Densest subgraph: Supermodularity, iterative peeling, and flow," in *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 2022, pp. 1531–1555.

[12] E. Harb, K. Quanrud, and C. Chekuri, "Faster and scalable algorithms for densest subgraph and decomposition," *Advances in Neural Information Processing Systems*, vol. 35, pp. 26 966–26 979, 2022.

[13] S. Khuller and B. Saha, "On finding dense subgraphs," in *Automata, Languages and Programming: 36th International Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part I 36*. Springer, 2009, pp. 597–608.

[14] N. Tatti and A. Gionis, "Density-friendly graph decomposition," in *WWW*, 2015.

[15] C. Ma, R. Cheng, L. V. Lakshmanan, and X. Han, "Finding locally densest subgraphs: a convex programming approach," *Proceedings of the VLDB Endowment*, vol. 15, no. 11, pp. 2719–2732, 2022.

[16] N. Tatti, "Density-friendly graph decomposition," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 13, no. 5, pp. 1–29, 2019.

[17] Y. Fang, K. Yu, R. Cheng, L. V. Lakshmanan, and X. Lin, "Efficient algorithms for densest subgraph discovery," *arXiv: Databases*, 2019.

[18] J. Bibby, "Axiomatisations of the average and a further generalisation of monotonic sequences," *Glasgow Mathematical Journal*, vol. 15, no. 1, pp. 63–65, 1974.

[19] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford large network dataset collection," http://snap.stanford.edu/data, Jun. 2014.

[20] H. Wan, Y. Zhang, J. Zhang, and J. Tang, "Aminer: Search and mining of academic social networks," *Data Intelligence*, 2019.

[21] R. A. Rossi and N. K. Ahmed, "The network data repository with interactive graph analytics and visualization," in *AAAI*, 2015. [Online]. Available: http://networkrepository.com

[22] R. Zafarani and H. Liu, "Social computing data repository at ASU," 2009. [Online]. Available: http://socialcomputing.asu.edu

[23] J. Kunegis, "Konect: the koblenz network collection," in *WWW*, 2013, pp. 1343–1350.

[24] B. Hooi, H. A. Song, A. Beutel, N. Shah, K. Shin, and C. Faloutsos, "Fraudar: Bounding graph fraud in the face of camouflage," in *SIGKDD*, 2016.

[25] N. Veldt, A. R. Benson, and J. Kleinberg, "The generalized mean densest subgraph problem," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 1604–1614.

[26] T. Lanciano, A. Miyauchi, A. Fazzone, and F. Bonchi, "A survey on the densest subgraph problem and its variants," *arXiv preprint arXiv:2303.14467*, 2023.

[27] W. Luo, C. Ma, Y. Fang, and L. V. Lakshman, "A survey of densest subgraph discovery on large graphs," *arXiv preprint arXiv:2306.07927*, 2023.

[28] A. Gionis and C. E. Tsourakakis, "Dense subgraph discovery: Kdd 2015 tutorial," in *KDD*, 2015.

[29] Y. Fang, W. Luo, and C. Ma, "Densest subgraph discovery on large graphs: applications, challenges, and techniques," *Proceedings of the VLDB Endowment*, vol. 15, no. 12, pp. 3766–3769, 2022.

[30] G. Gallo, M. D. Grigoriadis, and R. E. Tarjan, "A fast parametric maximum flow algorithm and applications," *SIAM Journal on Computing*, vol. 18, no. 1, pp. 30–55, 1989.

[31] Y. Asahiro, K. Iwama, H. Tamaki, and T. Tokuyama, "Greedily finding a dense subgraph," *Journal of Algorithms*, 2000.

[32] W. Feng, S. Liu, D. Koutra, H. Shen, and X. Cheng, "Specgreedy: unified dense subgraph detection," in *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part I*. Springer, 2021, pp. 181–197.

[33] A. Miyauchi and N. Kakimura, "Finding a dense subgraph with sparse cut," in *CIKM*, 2018.

[34] A. Anagnostopoulos, L. Becchetti, A. Fazzone, C. Menghini, and C. Schwiegelshohn, "Spectral relaxations and fair densest subgraphs," in *CIKM*, 2020.

[35] C. E. Tsourakakis, T. Chen, N. Kakimura, and J. W. Pachocki, "Novel dense subgraph discovery primitives: Risk aversion & exclusion queries," *ECML-PKDD*, 2019.

[36] A. Fazzone, T. Lanciano, R. Denni, C. E. Tsourakakis, and F. Bonchi, "Discovering polarization niches via dense subgraphs with attractors and repulsers," *Proceedings of the VLDB Endowment*, vol. 15, no. 13, pp. 3883–3896, 2022.

[37] S. Khuller and B. Saha, "On finding dense subgraphs," in *Proceedings of the 36th International Colloquium on Automata, Languages and Programming: Part I*, ser. ICALP '09. Springer-Verlag, 2009.

[38] C. Tsourakakis, "The k-clique densest subgraph problem," in *WWW*, 2015.

[39] R. Samusevich, M. Danisch, and M. Sozio, "Local triangle-densest subgraphs," in *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE, 2016, pp. 33–40.

[40] B. Sun, M. Danisch, T. Chan, and M. Sozio, "Kclist++: A simple algorithm for finding k-clique densest subgraphs in large graphs," *Proceedings of the VLDB Endowment (PVLDB)*, 2020.

[41] C. Tsourakakis, F. Bonchi, A. Gionis, F. Gullo, and M. Tsiarli, "Denser than the densest subgraph: extracting optimal quasi-cliques with quality guarantees," in *SIGKDD*, 2013, pp. 104–112.

[42] E. Galimberti, F. Bonchi, and F. Gullo, "Core decomposition and densest subgraph in multilayer networks," in *CIKM*, 2017.

[43] M. Mitzenmacher, J. Pachocki, R. Peng, C. Tsourakakis, and S. C. Xu, "Scalable large near-clique detection in large-scale networks via sampling," in *KDD*, 2015.

[44] C. Ma, Y. Fang, R. Cheng, L. V. Lakshmanan, W. Zhang, and X. Lin, "Efficient algorithms for densest subgraph discovery on large directed graphs," in *SIGMOD*, 2020.

[45] R. Andersen and K. Chellapilla, "Finding dense subgraphs with size bounds," in *WAW'09*.

[46] J. Chen and Y. Saad, "Dense subgraph extraction with application to community detection," *IEEE TKDE*, 2010.

[47] A. Costa, "Milp formulations for the modularity density maximization problem," *European Journal of Operational Research*, 2015.

[48] B. A. Prakash, A. Sridharan, M. Seshadri, S. Machiraju, and C. Faloutsos, "Eigenspokes: Surprising patterns and scalable community chipping in large graphs," in *PAKDD*, 2010.

[49] A. Beutel, W. Xu, V. Guruswami, C. Palow, and C. Faloutsos, "Copycatch: stopping group attacks by spotting lockstep behavior in social networks," in *WWW*, 2013, pp. 119–130.

[50] M. E. Newman, "Modularity and community structure in networks," *Proceedings of the national academy of sciences*, 2006.

[51] L. Qin, R.-H. Li, L. Chang, and C. Zhang, "Locally densest subgraph discovery," in *SIGKDD*, 2015, pp. 965–974.

[52] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani, "Kronecker graphs: an approach to modeling networks." *Journal of Machine Learning Research*, vol. 11, no. 2, 2010.

[53] S. B. Seidman, "Network structure and minimum degree," *Social networks*, vol. 5, no. 3, pp. 269–287, 1983.

[54] Y. Zhu, S. Liu, W. Feng, X. Cheng,"Fast Searching The Densest Subgraph And Decomposition With Local Optimality," *arXiv: Databases*, 2023.