Appendix

6.1. Proofs for Section 4

Before we prove Theorem 4.1, we first recap the b-transportation problem. Given a bipartite graph $B(V', E')$, and a required load value $b_u \geq 0, \forall u \in V'$, a *$b$-transportation* is an assignment $x_{uv} \geq 0$ to the edges of the bipartite graph such that $\sum_{v \in \delta(u)} x_{uv} = b_u$ for all $u \in V'$. Hence, it can be thought of as a fractional $b-$matching problem. Rado characterized the existence of a $b-$transportation in bipartite graphs using the following result

Theorem 6.1. *Rado, 1948 [37] Let $B(V', E')$ be a biparite graph. Then there exists a b-transportation for B if and only if $b(C) \geq \frac{1}{2} b(V')$ for each vertex cover C of B.*

In addition, we need the following lemma on the dense decomposition properties.

Lemma 6.2. *For a dense decomposition $S_1, ..., S_k$ of G with densities $\lambda_1, ..., \lambda_k$ obtained by Algorithm 1, we have: (i) $\lambda_1 > \lambda_2 > ... > \lambda_k \geq \frac{1}{2}$, (ii) For $S \subseteq S_i$, we must have $E(S, U_i) \geq \lambda_i |S|$.*

Proof: (1): Suppose for the sake of a contradiction that this is not the case, and let i be the first index where $\lambda_i \geq \lambda_{i-1}$. Then consider the set $S_{i-1} \cup S_i$ when the algorithm selected S_{i-1}. We have that

$$\lambda' = \frac{E(S_{i-1} \cup S_i) + E(S_{i-1} \cup S_i, \bigcup_{1 \leq t < i-1} S_t)}{|S_{i-1} \cup S_i|}$$

Note that S_j are disjoint by construction. So we have the simplification

$$\lambda' = \frac{E(S_{i-1}) + E(S_i) + E(S_{i-1}, \bigcup_{1 \leq t < i-1} S_t) + E(S_i, \bigcup_{1 \leq t < i} S_t)}{|S_{i-1}| + |S_i|} = \frac{\lambda_{i-1} |S_{i-1}| + \lambda_i |S_i|}{|S_{i-1}| + |S_i|} \geq \lambda_{i-1}$$

If $\lambda' = \lambda_{i-1}$ then this would be a contradiction to the maximality of S_{i-1}. If $\lambda' > \lambda_{i-1}$ then that would be a contradiction that S_{i-1} was the densest subgraph when it was chosen. Finally, $\lambda_k \geq 1/2$ is clear, the minimum density for a connected component is that of just a single edge which has density $\frac{1}{2}$.

(2): Suppose for the sake of a contradiction that this is not the case for some S_i and $S \subset S_i$, then consider the set $S_i - S$. Then we have that

$$\frac{E(S_i - S) + E(S_i - S, U_{i-1})}{|S_i - S|} = \frac{E(S_i) + E(S_i, U_{i-1}) - E(S, U_i)}{|S_i| - |S|} > \frac{\lambda_i |S_i| - \lambda_i |S|}{|S_i| - |S|} = \lambda_i$$

A contradiction to optimality of S_i. ■

We are now ready to prove Theorem 4.1.

Proof of Theorem 4.1 We construct a bipartite graph $B(V', E')$ as follows. $V' = L' \cup R'$, where the left side vertices $L' = V(G)$ and the right side vertices are the edges $R' = E(G)$. We set $b_u = \lambda_u$ for $u \in L'$, and $b_e = 1$ for $e \in R'$. We connect a vertex $i \in L'$ to $e \in E'$ if i is incident on e in G. It is clear that a $b-$transportation for B induces a feasible solution for LP 4.2. We will now show that there is a feasible $b-$transport using Theorem 6.1. First, note that $b(V') = \left(\sum_{i=1}^{k} \lambda_i |S_i| \right) + m \times 1$

But note that $m = \sum_{i=1}^{k} \lambda_i |S_i|$ (as each edge in G gets counted exactly once) which implies that $\frac{1}{2} b(V') = \sum_{i=1}^{k} \lambda_i |S_i|$. Now we will show that any vertex cover C of the bipartite graph must satisfy $b(C) \geq \sum_{i=1}^{k} \lambda_i |S_i|$ which would imply the theorem. Let $C_L = C \cap L'$ and $C_R = C \cap R'$ be the vertices in the vertex cover on the left and right respectively. Further, subdivide C_L, C_R into $C_{L1}, ..., C_{Lk}$ and $C_{R1}, ..., C_{Rk}$ where $C_{Li} = C_L \cap S_i$ and $C_{Ri} = C_R \cap (E(S_i) \cup E(S_i, \bigcup_{t<i} S_t))$. See Figure 6.3.

Consider $S_1 - C_{L1}$, we must have that $E(S_1 - C_{L1}) + E(S_1 - C_{L1}, C_{L1}) \geq \lambda_1 |S_1 - C_{L1}|$ using Lemma 6.2. But note that $E(S_1 - C_{L1}) \cup E(S_1 - C_{L1}, C_{L1})$ are precisely the edges that are not covered by C_{L1} and hence must be covered by C_{R1}, so it must be that $E(S_1 - C_{L1}) \cup E(S_1 - C_{L1}, C_{L1}) \subseteq C_{R1}$ which implies $|C_{R1}| \geq |E(S_1 - C_{L1})| + |E(S_1 - C_{L1}, C_{L1})| \geq \lambda_1 |S_1 - C_{L1}|$

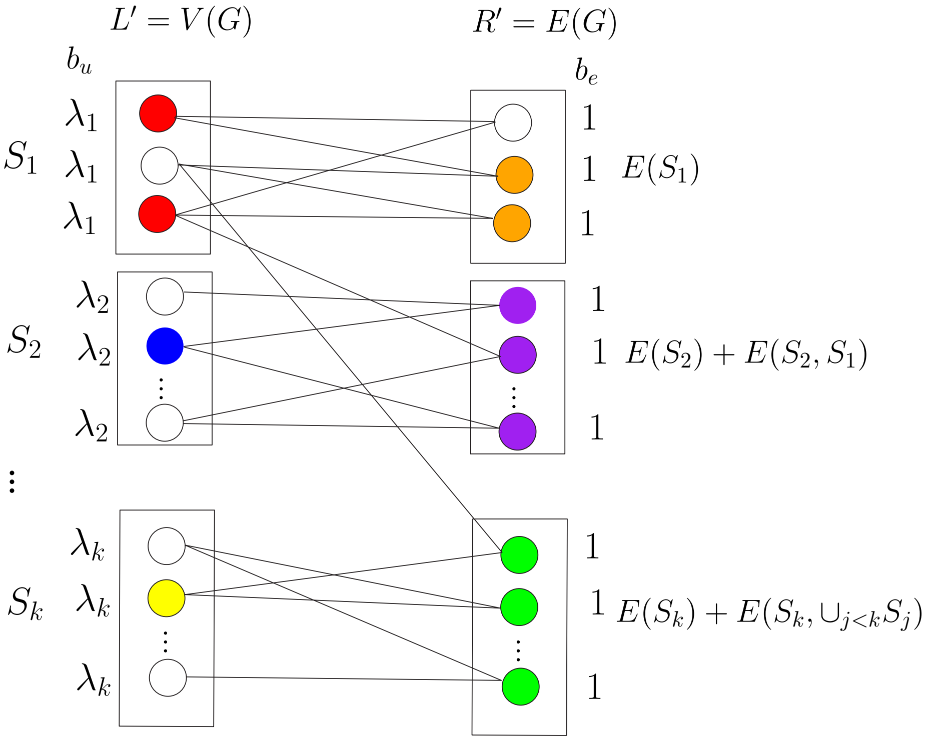

Figure 6.3: $B(V', E')$ bipartite graph in Proof of Theorem 4.1. The colored vertices are the vertex cover C. The red vertices are C_{L_1}, the blue vertices are C_{L_2}, and the yellow vertices are C_{L_k}. Similarly, the orange vertices are C_{R_1}, the purple vertices are C_{R_2}, and the green vertices are C_{R_k}.

So we have that $b(C_{L1} \cup C_{R1}) \geq \lambda_1 |C_{L1}| + \lambda_1 |S_1 - C_{L1}| = \lambda_1 |S_1|$. This analysis holds inductively to show that $b(C_{Li} \cup C_{Ri}) \geq \lambda_i |S_i|$. Summing up, we obtain the following,

$$b(C) = \sum_{i=1}^{k} b(C_{Li} \cup C_{Ri}) \geq \sum_{i=1}^{k} \lambda_i |S_i| = \frac{1}{2} b(V').$$

This finishes the proof. ■

Fujishige proved Theorem 4.3 [34] and more general versions of his theorem are also known (see [50]). Here we give a proof for the sake of completeness following the algorithmic definition of the decomposition.

Proof of Theorem 4.3: (1): Clearly $b_u \geq 0$ for all $u \in V$. Consider an arbitrary set $R \subseteq V$ and let $R_i = S_i \cap R$. Since $R_i \subseteq S_i$ and S_i is the densest set chosen during iteration i, then it must be that

$$\frac{f(R_i \cup S_1 \cup ... \cup S_{i-1}) - f(S_1 \cup ... \cup S_{i-1})}{|R_i|} \leq \frac{f(S_i \cup S_1 \cup ... \cup S_{i-1}) - f(S_1 \cup ... \cup S_{i-1})}{|S_i|} = \lambda_i$$

And hence we have that

$$b(R) = \sum_{u \in R} b_u = \sum_{i=1}^{k} \lambda_i |R_i| \geq \sum_{i=1}^{k} (f(R_i \cup S_1 \cup ... \cup S_{i-1}) - f(S_1 \cup ... \cup S_{i-1}))$$

$$\geq \sum_{i=1}^{k} (f(R_i \cup R_1 \cup ... \cup R_{i-1}) - f(R_1 \cup ... \cup R_{i-1})) = f(R)$$

Where the second inequality is by supermodularity of f and the last equality is because of the telescoping sum. Finally, note the chain of inequalities above hold with equality if $R = V$ since $R_i = S_i$. This implies $b(V) = f(V)$.

Problem	**Algorithm**	**Convergence (Current worst case number of iters.)**	**Time/iter**	**Iteration parallelizable?**
DSG-LD	Frank-Wolfe based Algorithm	$O(\frac{m\Delta(G)}{\epsilon^2})$ iterations for ϵ-load vector	$O(m)$	Yes
DSG-LD	Greedy++	Load vector not proven to converge, but experimentally does.	$O(m \log n)$	No
DSG-LD	MWU based Algorithm	$O(\frac{m\Delta(G)}{\epsilon^2})$ iterations for ϵ-load vector	$O(m)$	Yes
DSG-LD	**FISTA based Algorithm**	$O(\frac{\sqrt{m\Delta(G)}}{\epsilon})$ iterations for ϵ-load vector	$O(m)$	Yes
DSG	Bahmani *et al.* [30] primal-dual	$O(\frac{\log m}{\epsilon^2})$ for $(1-\epsilon)$ *multiplicative* DSG.	$O(m)$	Yes
DSG	Boob *et al.* [31] via mixed packing-covering LP solver	$\tilde{O}(\frac{m\Delta(G)}{\epsilon})$ for $(1-\epsilon)$ *multiplicative* DSG.	NA	NA
DSG	Greedy++	$O(\frac{\Delta(G)}{\lambda^* \epsilon^2})$ for $(1-\epsilon)$ *multiplicative* DSG	$O(m \log n)$	No
DSG	Chekuri *et al.* [2] via approximate flow	$O(\frac{\log m}{\epsilon})$ for $(1-\epsilon)$ *multiplicative* DSG.	$\tilde{O}(m)$	No
DSG	Frank-Wolfe based Algorithm	$O(\frac{mn\Delta(G)}{\epsilon^2})$ for ϵ *additive* DSG using fractional peeling from this paper.	$O(m)$	Yes
DSG	**FISTA based Algorithm**	$O(\frac{\sqrt{mn\Delta(G)}}{\epsilon})$ for ϵ *additive* DSG using fractional peeling from this paper.	$O(m)$	Yes

Figure 6.4: Summary of currently known bounds on different iterative algorithms for DSG and DSG-LD including results in this paper.

(2) : Let $b^*_u = \lambda_u$ with $b^* \in B_f$ from (1). Let $b \in B_f$ be a lexicographically minimal base. We will prove that $b = b^*$ by inductively proving that for all i, $b_u = \lambda_u$ if $u \in S_i$. Consider $i = 1$ for the base case. Since $b \in B_f$ we have

$$b(S_1) \geq f(S_1) = \lambda_1 |S_1|$$

Hence the maximum load in b is at least λ_1. Since the maximum load in b^* is λ_1, then it forces $b_u = \lambda_u$ for $u \in S_1$. Now we proceed inductively, assuming that $b_u = \lambda_u$ for $u \in S_1 \cup ... \cup S_i$. Since $b \in B_f$,

$$\sum_{h=1}^{i+1} b(S_h) = b(S_1 \cup ... \cup S_{i+1}) \geq f(S_1 \cup ... \cup S_{i+1}) = \sum_{h=1}^{i+1} (f(S_1, ..., S_h) - f(S_1, ..., S_{h-1}))$$

$$= f(S_1 \cup ... \cup S_{i+1}) - f(S_1 \cup ... \cup S_i)) + \sum_{h=1}^{i} b(S_h) = \lambda_{i+1} |S_{i+1}| + \sum_{h=1}^{i} b(S_h)$$

This implies that $b(S_{i+1}) \geq \lambda_{i+1} |S_{i+1}|$. We have $b^*_u = b_u$ for $u \in S_1 \cup ... \cup S_i$ by induction hypothesis. Since $b^*_u = \lambda_{i+1}$ for all $u \in S_{i+1}$ and $b(S_{i+1}) \geq \lambda_{i+1}|S_{i+1}|$ it follows that $b_u = \lambda_{i+1}$ for $u \in S_{i+1}$ for otherwise b is not lexicographically minimal.

Hence, by induction, $b_u = \lambda_u$ for all $u \in V$.

(3): The function $g(b) = \sum\limits_{u \in V} b_u^2$ is strictly convex. B_f is a bounded polyhedron and hence a closed convex set. In addition, we showed that B_f is feasible. Any strictly convex function with a feasible convex constraint set must have a unique solution, and so b^* must be unique.

Now consider the constraint polytope $\mathcal{C}$ defined as the intersection of the following k inequalities

$$\mathcal{C} = \{b \in \mathbb{R}^{|V|} : \forall 1 \leq r \leq k, \sum_{l=1}^{r} b(S_l) \geq \sum_{l=1}^{r} \lambda_l |S_l|\}$$

Recall if $b \in B_f$ then

$$\sum_{i=1}^{r} b(S_i) = b(S_1 \cup ... \cup S_r) \geq f(S_1 \cup ... \cup S_r) = \sum_{i=1}^{r} f(S_1 \cup ... \cup S_i) - f(S_1 \cup ... \cup S_{i-1}) = \sum_{i=1}^{r} \lambda_i \, |S_i|$$

And so $b \in B_f \implies b \in \mathcal{C}$. Hence $\min_{b \in \mathcal{C}} g(b) \leq \min_{b \in B_f} g(b)$.

We aim to prove that for the unique optimal solution of g under $\mathcal{C}$, all inequalities are active (i.e hold with equality). Let b be the optimal solution of g under $\mathcal{C}$. From (1), we have that $b(V) = f(V)$ and so the last inequality has to be active. Now suppose for the sake of contradiction that not all inequalities are active, and let i be the first index of an inequality that is not active. Since $\sum_{h=1}^{i-1} b(S_h) = \sum_{h=1}^{i-1} \lambda_h \, |S_h|$ and $\sum_{h=1}^{i} b(S_h) > \sum_{h=1}^{i} \lambda_h \, |S_h|$, then it must be that $b(S_i) > \lambda_i |S_i|$. Similarly, let $j > i$ be the first index of an inequality after i which is active (this must exist since the last inequality holds with equality). Then it must be that $b(S_j) < \lambda_j |S_j|$. Now let

$$\epsilon = \min\left(\min_{i \leq t < j}\left(\sum_{r=1}^{t} b(S_r) - \sum_{r=1}^{t} \lambda_r s_r\right), \frac{1}{2}(\lambda_i - \lambda_j)\right) > 0$$

be the minimum "excess" from inequalities i to $j-1$ and half the difference between λ_i, λ_j. Since $b(S_i) > \lambda_i |S_i|$, then there exists $b_u > \lambda_i$ for $u \in S_i$. Similarly, there exists $b_v < \lambda_j$ for $v \in S_j$. Now consider the solution of b' where $b'_u = b_u - \epsilon, b'_v = b_v + \epsilon$, and $b'_x = b_x$ otherwise. The solution is feasible in $\mathcal{C}$ because inequalities $1, ..., i-1$ stay the same (no b_u, b_v variable), inequalities $i, ..., j-1$ stay feasible (LHS decreases by ϵ), and the effect by inequality j is cancelled. However, we have that since $b_u > \lambda_i > \lambda_j > b_v$ and $\epsilon \leq \frac{1}{2}(\lambda_i - \lambda_j)$ that $(b_u - \epsilon)^2 + (b_v + \epsilon)^2 < b_u^2 + b_v^2$ So the solution b' has a strictly smaller cost, contradicting optimality of b.

This implies that all inequalities have to hold with equality in an optimal solution. So $\sum_{u \in S_i} b_u = \lambda_i |S_i|$. The sum of squares is minimized if and only if all variables have equal weight, and so it must be that $b_u = \lambda_i = \lambda_u$ for all $u \in S_i$. This shows $g(b)$ is minimized at $b_u = \lambda_u$ under $\mathcal{C}$. But recall from (1) that $b \in B_f$, and so b is the unique optimal solution for Problem 4.7 ■

We now show the equivalence of the two LPs 4.2 and 4.6 for DSG.

Proof of Theorem 4.4: Suppose $b \in B_f$. Then $b(V) = m$ and $b(S) \geq |E(S)|$ for all $S \subseteq V$. We need to prove the existence of $x \geq 0$ such that $x_{uv} + x_{vu} = 1$ for all edge $\{u, v\} \in E$ and such that the total load on each vertex is at most b. The idea from the proof of Theorem 4.1 goes through to show this. The only fact we used in the proof of Theorem 4.1 is that $b(S) \geq |E(S)|$ for certain sets $S \subseteq V$ and $b(V) = m$ which hold as we mentioned.

Suppose x, b satisfy the constraints of 4.2. Since $x_{uv} + x_{vu} = 1$ for each edge $\{u, v\}$, for any $S \subseteq V$,

$$b(S) = \sum_{u \in S} b_u = \sum_{u \in S} \sum_{v \in \delta(u) \cap S} x_{uv} + \sum_{u \in S} \sum_{v \in \delta(u) \setminus S} x_{uv} \geq |E(S)|$$

Similarly, $b(V) = \sum_{u \in V} \sum_{v \in \delta(u)} x_{uv} = m = |E(V)|$. Thus $b \in B_f$. ■

6.2. Details of FISTA based Algorithm

Proof of Lemma 5.1 $\nabla f_{uv} = 2 \sum_{w \in \delta(u)} x_{uw}$. So for $x, y \in \mathbb{R}^{2m}$,

$$\|\nabla f(x) - \nabla f(y)\|^2 = 4 \sum_{uv \in ord(E)} \left(\sum_{w \in \delta(u)} x_{uw} - y_{uw}\right)^2 = 4 \sum_{u \in V} \sum_{v \in \delta(u)} \left(\sum_{w \in \delta(u)} x_{uw} - y_{uw}\right)^2$$

$$\leq 4\Delta(G) \sum_{u \in V} \left(\sum_{w \in \delta(u)} x_{uw} - y_{uw}\right)^2 \leq 4\Delta(G)^2 \sum_{u \in V} \sum_{w \in \delta(u)} (x_{uw} - y_{uw})^2 = 4\Delta(G)^2 \|x - y\|^2$$

Where the last inequality holds by Cauchy-Schwarz inequality. ■

Proof of Lemma 5.2 Fix $u < v$ and let $p = \text{prox}_h(x)$. Then we aim to minimize

$$\sum_{uv \in ord(E)} (p_{uv} - x_{uv})^2 = \sum_{uv \in E} \left((p_{uv} - x_{uv})^2 + (1 - p_{uv} - x_{vu})^2\right)$$

So it is sufficient to minimize $(p_{uv} - x_{uv})^2 + (1 - p_{uv} - x_{vu})^2$ individually for each unordered edge $u < v$ subject to $0 \leq p_{uv} \leq 1$. Proving that the described proximal mapping minimizes this quadratic is a standard exercise so we omit it.

We describe below the full details of the FISTA based algorithm for DSG.

Algorithm 3 FISTA Algorithm for Densest Subgraph

```
Input is Graph $G$ and number of iterations $T$. Assume $E = E(G)$ is ordered (so includes $(u, v)$ and $(v, u)$ for every edge).
$\Delta = \max_{u \in G} |\delta(u)|$                                ▷ Maximum Degree
$\alpha \leftarrow \frac{1}{2\Delta}$                                ▷ Learning Rate
$x^{(0)}(u, v) = 1 \quad \forall (u, v) \in E, u < v$
$x^{(0)}(v, u) = 0 \quad \forall (u, v) \in E, u < v$
$y^{(0)} = x^{(0)}$
for $t \in [1, T]$ do
    $b^{(t)}(u) = 0 \quad \forall u \in V$                ▷ Calculate Load with respect to $y^{(t-1)}$
    for $u \in G$ do
        for $v \in \delta(u)$ do
            $b^{(t)}(u) = b^{(t)}(u) + y^{(t-1)}(u, v)$

    $g^{(t)}(u, v) = 0 \quad \forall (u, v) \in E$         ▷ Calculate Gradient with respect to $y^{(t-1)}$
    for $(u, v) \in E, u < v$ do
        $g^{(t)}(u, v) = g^{(t)}(u, v) + 2b^{(t)}(u)$
        $g^{(t)}(v, u) = g^{(t)}(v, u) + 2b^{(t)}(v)$

    $z^{(t)}(u, v) = y^{(t)}(u, v) - \alpha g^{(t)}(u, v) \quad \forall (u, v) \in E$       ▷ Descent direction
    $x^{(t)}(u, v) = 0 \quad \forall (u, v) \in E$    ▷ Calculate New $x^{(t)}$, which is projected descent direction
    for $(u, v) \in E, u < v$ do
        $diff \leftarrow z^{(t-1)}(u, v) - z^{(t-1)}(v, u)$
        if $diff \geq -1$ and $diff \leq 1$ then
            $x^{(t)}(u, v) \leftarrow \frac{diff+1}{2}$
        else if $diff > 1$ then
            $x^{(t)}(u, v) \leftarrow 1$
        else
            $x^{(t)}(u, v) \leftarrow 0$
        $x^{(t)}(v, u) \leftarrow 1 - x^{(t)}(u, v)$

    $y^{(t)}(u, v) \leftarrow x^{(t)}(u, v) + \frac{t-1}{t+2}(x^{(t)}(u, v) - x^{(t-1)}(u, v)) \quad \forall (u, v) \in E$ ▷ Calculate New $y^{(t)}$

return $x^{(T)}$
```

6.3. Approximate densest decomposition via fractional peeling

Proof of Theorem 5.5

$$f(b) - f(b^*) = \sum_{u \in V} b_u^2 - \sum_{u \in V} \lambda_u^2 = \sum_{i=1}^{k} \sum_{u \in S_i} (b_u^2 - \lambda_u^2) = \sum_{i=1}^{k} \sum_{u \in S_i} (b_u^2 - \lambda_i^2) \leq \mu$$

Now let $b_u = \lambda_u + \delta_u$, then we have from above that

$$\sum_{i=1}^{k} \sum_{u \in S_i} (2\lambda_i \delta_u + \delta_u^2) \leq \mu$$

We will first show that $\sum_{i=1}^{k} \sum_{u \in S_i} 2\lambda_i \delta_u \geq 0$. Note that for any l, we have

$$\sum_{i=1}^{l} \sum_{u \in S_i} (\lambda_i + \delta_u) = \sum_{i=1}^{l} \sum_{u \in S_i} b_u \geq \sum_{i=1}^{l} \lambda_i |S_i|$$

Since the edges with both endpoints in $S_1 \cup ... \cup S_l$ get double counted, and $x_{uv} + x_{vu} = 1$. Which implies that for all l,

$$\sum_{i=1}^{l} \sum_{u \in S_i} \delta_u \geq 0$$

Now we prove that for all l, and any $a_1 > a_2 > ... > a_l \geq 0$

$$\sum_{i=1}^{l} \sum_{u \in S_i} \delta_u a_i \geq 0$$

by induction. Observe that it holds for $l = 1$ since $a_1 \sum_{u \in S_1} \delta_u \geq 0$. For $l = r$, we have that

$$\sum_{i=1}^{r} \sum_{u \in S_i} \delta_u a_i = \sum_{i=1}^{r-1} \sum_{u \in S_i} \delta_u (a_i - a_r) + a_r \sum_{i=1}^{r} \sum_{u \in S_i} \delta_u \geq 0 + 0 = 0$$

By induction. Since $\lambda_1 > ... > \lambda_k$ by Lemma 6.2, this implies

$$\sum_{i=1}^{k} \sum_{u \in S_i} \lambda_i \delta_u \geq 0$$

Which implies

$$\sum_{i=1}^{k} \sum_{u \in S_i} \delta_u^2 \leq \mu$$

Now let $\mu = \epsilon^2$, which would imply

$$\sum_{i=1}^{k} \sum_{u \in S_i} (b_u - \lambda_u)^2 \leq \epsilon^2$$

And hence $\|b - b^*\| \leq \epsilon$ ■

6.4. Proof of Theorem 5.6

We first note the running time follows easily from using a heap in the fractional peeling subroutine to identify the next minimum-load vertex. We focus on proving the approximation factor.

In what follows, refer to Figure 6.5. Let $x^{(0)} = x$ and $b^{(0)} = b$. Each iteration $t = 1, 2, \ldots$, the algorithm runs fractional peeling over the remaining vertices, with respect to the fractional orientation given by $x^{(t-1)}$ and the loads given by $b^{(t)}$. This produces a set of vertices T_t. We remove T_t from the vertex set. $x^{(t)}$ is obtained from $x^{(t-1)}$ by assigning all the edges cut by T_t to the endpoint not in T_t. That is, for each edge $\{u, v\}$ cut by T_t in the remaining graph, where $u \in T_t$ and $v \notin T_t$ we set $x_{uv}^{(t)} = 0$ and $x_{vu}^{(t)} = 1$. We let $b^{(t)}$ denote the loads induced by $x^{(t)}$. Note that $b_u^{(t)}$ is nondecreasing for $u \notin T_t$, and nonincreasing for $u \in T_t$. In particular we have $b_u^{(t)} \geq b_u^{(0)} \geq \lambda_u - \varepsilon$ for every remaining vertex u.

We want to show that for each iteration t, and each vertex $u \in T_t$, the density of T_t is at least $\lambda_u - \varepsilon(1 + \sqrt{n})$.

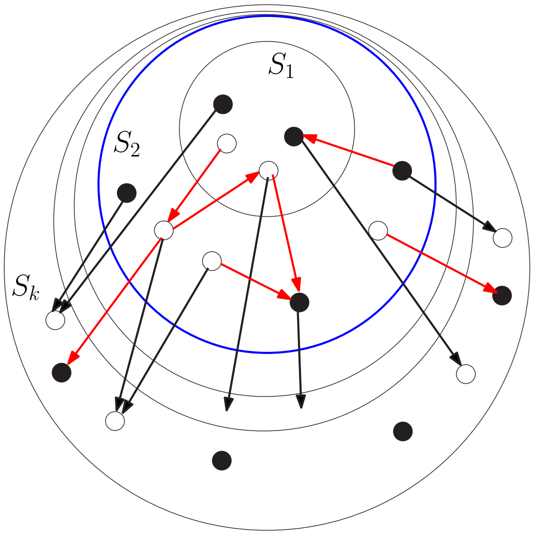

Figure 6.5: Let $S_1, ..., S_k$ be as shown in the figure. Suppose we are in iteration t. The dark vertices represent the vertices that were deleted in previous rounds (i.e $T_1 \cup ... \cup T_{t-1}$), while the white vertices are those that were not deleted yet. The sum of $\delta_2^{(t)}$ only includes the black edges leaving S_1, S_2 to "outside" white points. The red edges are all examples of edges not included in the sum.

First, for each iteration t, and each index i from the dense decomposition, let $\delta_i^{(t)}$ denote the sum,

$$\delta_i^{(t)} = \sum_{\substack{u \in S_1 \cup \cdots \cup S_i \\ v \in S_{i+1} \cup \cdots \cup S_k \\ v \notin T_1 \cup \cdots \cup T_t}} x_{uv}^{(t)}.$$

At a high-level, $\delta_i^{(t)}$ represents the sum of loads in $S_1 \cup \cdots \cup S_i$ from edges cut by $S_1 \cup \cdots \cup S_i$, except omitting the edges where the endpoint outside $S_1 \cup \cdots \cup S_i$ was taken in one of the first t iterations.

We claim that for each index i and iteration t, we have

$$\delta_i^{(t)} \leq \varepsilon\sqrt{n}.$$

We first observe that $\delta_i^{(t)}$ is non-increasing in t. Indeed, fix t, and consider a term $x_{uv}^{(t)}$ appearing in the sum. (That is, $u \in S_1 \cup \cdots \cup S_i$, $v \in S_{i+1} \cup \cdots \cup S_k$, and $v \notin T_1 \cup \cdots \cup T_t$.) In the $(t+1)$th iteration, we select a new set T_{t+1} and $x_{uv}^{(t+1)}$ is bigger than $x_{uv}^{(t)}$ only if $v \in T_{t+1}$. But in this case, $x_{uv}^{(t+1)}$ is omitted from the sum for $\delta_i^{(t)}$.

Since $\delta_i^{(t)}$ is non-increasing in t, suffices to prove the claim for $t = 0$, when $b^{(0)} = b$ and $x^{(0)} = x$. To this end, observe that

$$\sum_{u \in S_1 \cup \cdots \cup S_i} b_u = \delta_i^{(0)} + |E(S_1 \cup \cdots \cup S_i)| = \delta_i^{(0)} + \sum_{u \in S_1 \cup \cdots \cup S_i} \lambda_u.$$

Rearranging and applying the Cauchy-Schwarz inequality, we have

$$\delta_i^{(0)} = \sum_{u \in S_1 \cup \cdots \cup S_i} (b_u - \lambda_u) \leq \sqrt{n} \sqrt{\sum_{u \in S_1 \cup \cdots \cup S_i} (b_u - \lambda_u)^2} \leq \varepsilon\sqrt{n}.$$

This establishes the inequality for $t = 0$, hence all t by monotonicity.

Let $u \in T_t$ and suppose $u \in S_i$. We want to show that T_t has density at least $\lambda_u - \varepsilon(1 + \sqrt{n})$. Let v be the first vertex in $S_1 \cup \cdots \cup S_i$ peeled in the tth iteration; in particular, $\lambda_v \geq \lambda_u$. Recall that just before v is peeled, v has the lowest load remaining of any vertex, and the sum of loads of the remaining vertices counts the total number of edges in T_t. Thus the load at v is a lower bound on the density of T_t. Additionally, before v is peeled, any decrease in v's load is from edges $\{v, w\}$ where $w \in S_{i+1} \cup \cdots \cup S_k$ and $w \notin T_1, \ldots, T_{t-1}$. Thus v has load at least $b_v^{(t-1)} - \delta_i^{(t-1)}$. Putting everything together, we conclude that T_t has density at least

$$b_v^{(t-1)} - \delta_i^{(t-1)} \geq b_v^{(0)} - \varepsilon\sqrt{n} \geq \lambda_v - \varepsilon(1+\sqrt{n}), \geq \lambda_u - \varepsilon(1+\sqrt{n}),$$

as desired. This completes the proof.

6.5. Hypergraphs

The projective results for DSG can be generalized for DSS. We say a supermodular function is *projectable* if given a vector $y \in \mathbb{R}^{|V|}$, there is a fast oracle that can calculate $prox_f(y) = \min\limits_{x \in B_f} \|x - y\|_2^2$.

Theorem 6.3. *Given f, a projectable supermodular function, and an initial load vector $b^{(0)}$ for the guess of $b_u^* = \lambda_u$, there exists an algorithm that returns an approximate load vector $\hat{b}$ that uses $O(\frac{\|b^{(0)}-b^*\|}{\epsilon})$ oracle calls to the projection oracle, and satisfies $\|b - b^*\| \leq \epsilon$ for all $u \in V$.*

Proof: Consider applying FISTA [40] on Problem 4.7 where we have unconstrained optimization problem $\sum_{u \in V} b_u^2 + h(b)$ where $h(b)$ is an indicator function for B_f. We have that $\nabla f(b) = 2b$, and hence the Lipschtiz constant of ∇f is 2. Applying Lemma 5.3 with a learning rate of 0.5, we get the desired result. ■

A hypergraph $G = (V, E)$ generalizes the idea of a graph by allowing edges to have size greater than two. For example, if $V = \{a, b, c, d\}$, then a potential "edge" is $e = \{a, b, d\}$. The rank r of a hyper graph is $\max_{e \in E} |e|$. For practical purposes, r is generally "small". Given a hypergraph $G = (V, E)$ we let $E(S)$ denote the set of all hyperedges in E that are fully contained in S, that is $E(S) = \{e \in E \mid e \subseteq S\}$. One can easily verify that the function $f : 2^V \to \mathbb{R}_+$ where $f(S) = |E(S)|$ is a monotone nonnegative supermodular function.

We can generalize the approach for DSG to hypergraphs as follows. Charikar's LP relaxation can be generalized to hypergraphs. Here we focus on the dual. For $e \in E$ and $u \in e$, we define a variable $x_{e,u}$ which corresponds to the load that e assigns to u. The LP requires each e to be assigned to its end points and hence we have a constraint $\sum_{u \in e} x(e, u) = 1$. The load on u, denote by the variable b_u, is $\sum_{e: u \in e} x(e, u)$. The goal is to minimize $\max_{u \in V} b_u$. Similar to Theorem 4.4, one can prove in the same way that $b \in B_f$ if and only if $\exists x$ that induces b. Now given a vector y that induces b, we can project it on B_f by finding $x_{e,u}$ that minimizes $\sum_{u \in e}(x_{e,u} - y_{e,u})^2$ subject to $\sum_{u \in e} x_{e,u} = 1$ and $x_{e,u} \geq 0$. This is known as the simplex projection and it has a simple closed form solution (See [51] for the basic algorithm, and [52] for a recent distributed variant of the algorithm). The algorithm in this case would take $O(|e| \log |e|)$ time for each $e \in E$ to do the projection, and hence overall the projection step takes $O(p \log r)$ for all edges where $p = \sum_{e \in E} |e|$ is the representation size of the hypergraph G (p corresponds to m in graphs). We can apply FISTA analysis in a very similar fashion to that for graphs to obtain the following theorem.

Theorem 6.4. *For a hyper graph G with rank r, maximum degree $\Delta(G)$ (i.e $\max_{u \in V} |\{e : u \in e\}| = \Delta(G)$), and size $p = \sum_{e \in E} |e|$, there exists an algorithm that takes $O(\frac{\sqrt{r\Delta(G)p}}{\epsilon})$ iterations, each needing $O(p \log r)$ time, to compute an ϵ-approximate load vector $\hat{b}$ satisfying $\left\|\hat{b} - b^*\right\| \leq \epsilon$.*

6.6. Approximate load vector via minimum-cost flow

We set up the problem as a quadratic min cost flow problem. Namely, we will set the flow network $\mathcal{V} = \{s\} \cup \{a_v : v \in V\} \cup \{a_e : e \in E\} \cup \{t\}$. We add an edges to $\mathcal{E}$ of the form (s, a_v) of capacity $deg_G(v)$. We add an edge (a_v, a_e) is v if one of the endpoints of e of capacity 1. Finally, we add an edge (a_e, t) of capacity 1 for all edges $e \in E$. See Figure 6.6. One can verify that the maximum flow has cost $m = |E|$ using max flow min cut theorem. Let $\mathcal{F}$ be the set of valid maximum flows

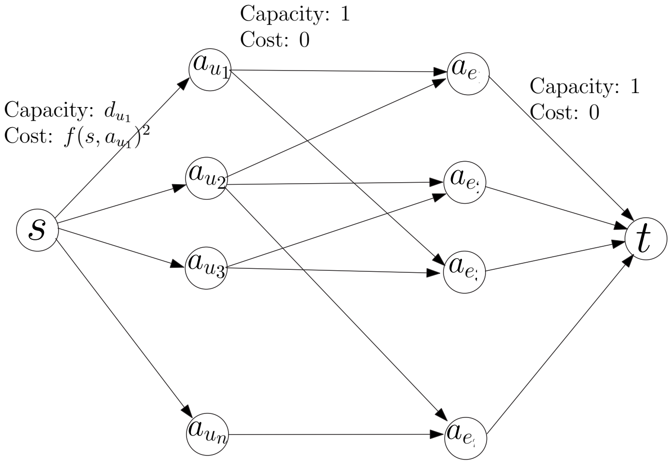

Figure 6.6: Quadratic min cost flow network

for $(\mathcal{V}, \mathcal{E})$. We are interested in the problem of

$$\begin{aligned} \text{minimize} \quad & \sum_{u \in V} f(s, a_u)^2 \\ \text{subject to} \quad & f \in \mathcal{F} \end{aligned} \tag{6.1}$$

One can verify that the optimal flow for Problem 6.1 gives the solution for Problem 4.3; any flow $f \in \mathcal{F}$ corresponds to a solution of the same cost to Problem 4.3 and vice versa. Hence the optimal flow f^* is the dense decomposition vector.

Observe that this flow network has size $O(m)$. We will use Theorem 10.14 from [43]. Define $h_{(s,a_{v_i})}(x) = x^2$ and $h_e(x) = 0$ otherwise for $e \neq (s, a_{v_i})$. These are functions which are sums of $\tilde{O}(1)$ $p-$norms. Hence, in $m^{1+o(1)}$ time, we can compute a min cost flow f of value $|E|$. Setting $C = 3$ in the Theorem, we get that

$$\sum_{u \in V} f(s, a_u)^2 \leq \sum_{u \in V} f^*(s, a_u)^2 + O(\exp(-\log^3 m)) = OPT + \frac{1}{m^{\log^2 m}} \leq OPT + \frac{1}{m^4} \leq OPT + \epsilon^2$$

Note that the smallest ϵ we care about is $\epsilon \geq n^{-2}$ (since any smaller epsilon wouldn't change the value from setting $\epsilon = n^{-2}$). The chain of inequalities hold for reasonably large m (say $m \geq 5$). Hence f is an ϵ-approximate load vector.

6.7. Further Details of Experimental Section

6.7.1. More details on MWU and FRANK-WOLFE based algorithms

We described the theoretical aspects of the FRANK-WOLFE and MWU algorithms in Sections 5.5. Here we discuss some concrete details of our implementation.

We use the FRANK-WOLFE implementation in the context of DSG as described by Danisch *et al.* [3]. The algorithm maintains an edge assignment vector that it updates in each iteration t. The edge assignment vector $x^{(t-1)}$ at the start of iteration t naturally induces a vertex load vector b^{t-1}. We loop over each edge, and set $y_{uv}^{(t)} = 1, y_{vu}^{(t)} = 0$ if $b_u^{(t-1)} < b_v^{(t-1)}$ and $y_{uv}^{(t)} = 0, y_{vu}^{(t)} = 1$ otherwise. Finally, we let $x^{(t)} = x^{(t-1)} + \frac{2}{t+2} y^{(t)}$ which itself induces a new load vector $b^{(t)}$. Each iteration takes $O(m)$ time. This is the same implementation as described in the Danisch *et al.* [3] paper except that we initialize the starting vector $x^{(0)}$ differently. Danisch *et al.* initialize $x_{uv}^{(0)} = x_{vu}^{(0)} = 0.5$ while we initialize it with the edge assignment obtained from running the Greedy algorithm.

There is an alternative way to implement the algorithm without using edge assignment variables. We maintain a vertex load vector $b^{(t)}$ of size n. Each iteration is implemented as follows. We sort the vertices $u_1 < ... < u_n$ in ascending order of $b_u^{(t-1)}$ (ties broken arbitrarily). Then, for each u_i, we set

$$b_u^{(t)} = b_u^{(t-1)} + \frac{2}{t+2} \left| u_j \in \delta(u_i) : j > i \right|$$

One can verify that the two algorithms are equivalent with a slightly different implementation. This implementation takes $O(n \log n)$ time for the sort, and an $O(m)$ loop over the graph, but does not have to store an additional array of size $2m$ for each edge. When $n \ll m$, this can make a substantial difference.

The MWU algorithm can also be implemented in two different ways (one edge based, and one vertex based). We choose to implement the algorithm in the vertex based approach in comparison to the FRANK-WOLFE based approach. Specifically, the algorithm maintains a vertex load vector that it updates in each iteration as follows. In iteration t, it sorts the vertices as $u_1 < ... < u_n$ in ascending order of $b_u^{(t-1)}$ values. Then, for each u_i, it sets

$$b_u^{(t)} = b_u^{(t-1)} + \frac{1}{t+1} \left| u_j \in \delta(u_i) : j > i \right|$$

We keep track of the edge assignment vector $x^(t)$ (that induces $b^{(t)}$) for the fractional peeling experiment — however, when reporting the running time we do not add this overhead since the algorithm does not require maintaining the $x^{(t)}$ vector.

6.7.2. Additional data

See the end of the Appendix for enlarged plots of the main paper plots and additional plots for all datasets.

1. Figure 6.7 shows the density achieved as number of iterations vary for all algorithms on all datasets when static load sorting is used instead of fractional peeling. Figure 6.8 shows the effect of adding fractional peeling to all the algorithms.
2. Figure 6.9 shows the time per iteration histogram for all datasets and all algorithms that we tested.
3. Figures 6.10 and 6.11 show the error plots (i.e sum of $\sum_{u \in V} b_u^2$) of all algorithms on all datatasets. Specifically, Figure 6.11 zooms in on the last few iterations to see what is happening near the end.
4. Figure 6.12 shows the sorted load vector after 100 iterations of FISTA in sorted order, where a vertex rank is its relative order in terms of its load vector in V, and load is the value b_u. It appears that for each S_i, FISTA focuses on adjusting b_u for most vertices in S_i, but a few vertices “lag” behind in lower/higher levels, and slowly bubble down as shown the dataset for CLOSE CLIQUES and ROADNET PA. This gives some intuition on why fractional peeling does well in practice as these “trailing” vertices will be peeled first by fractional peeling allowing the dense component to stabilize.
5. Figure 6.13 shows the wall clock time of different algorithms on all 8 datasets. Figure 6.14 shows the same figure but zoomed in on the first 20 percent of the time.

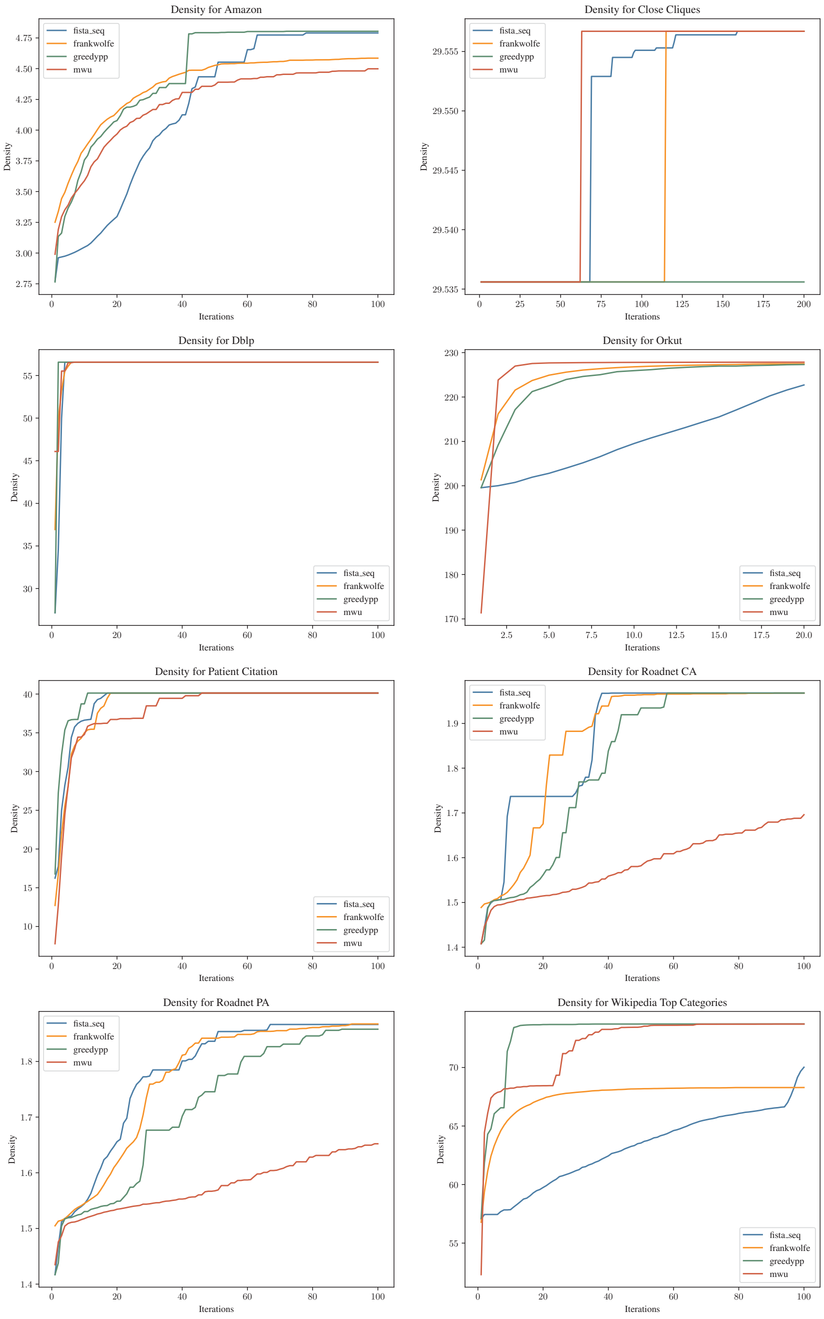

Figure 6.7: Density based on sorting loads

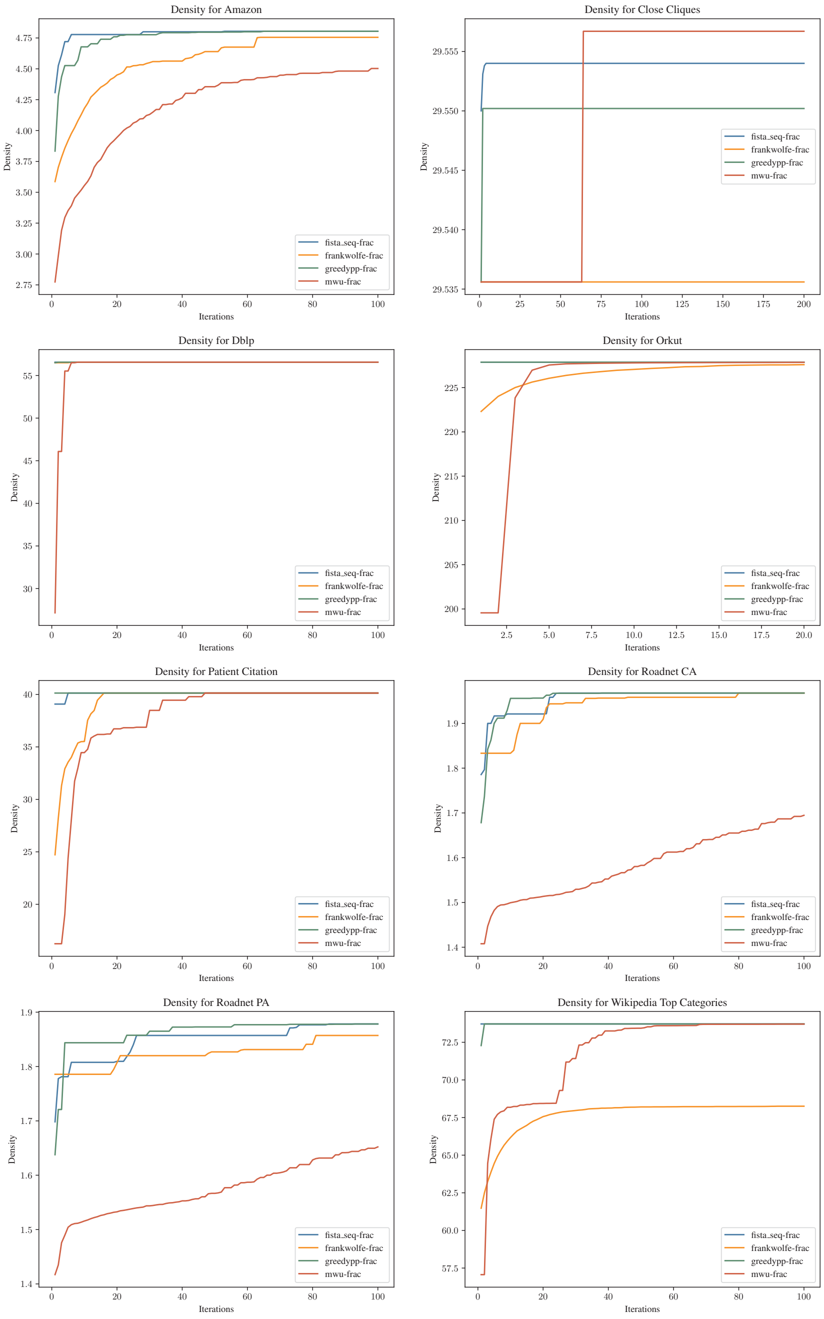

Figure 6.8: Density based on Fractional Peeling

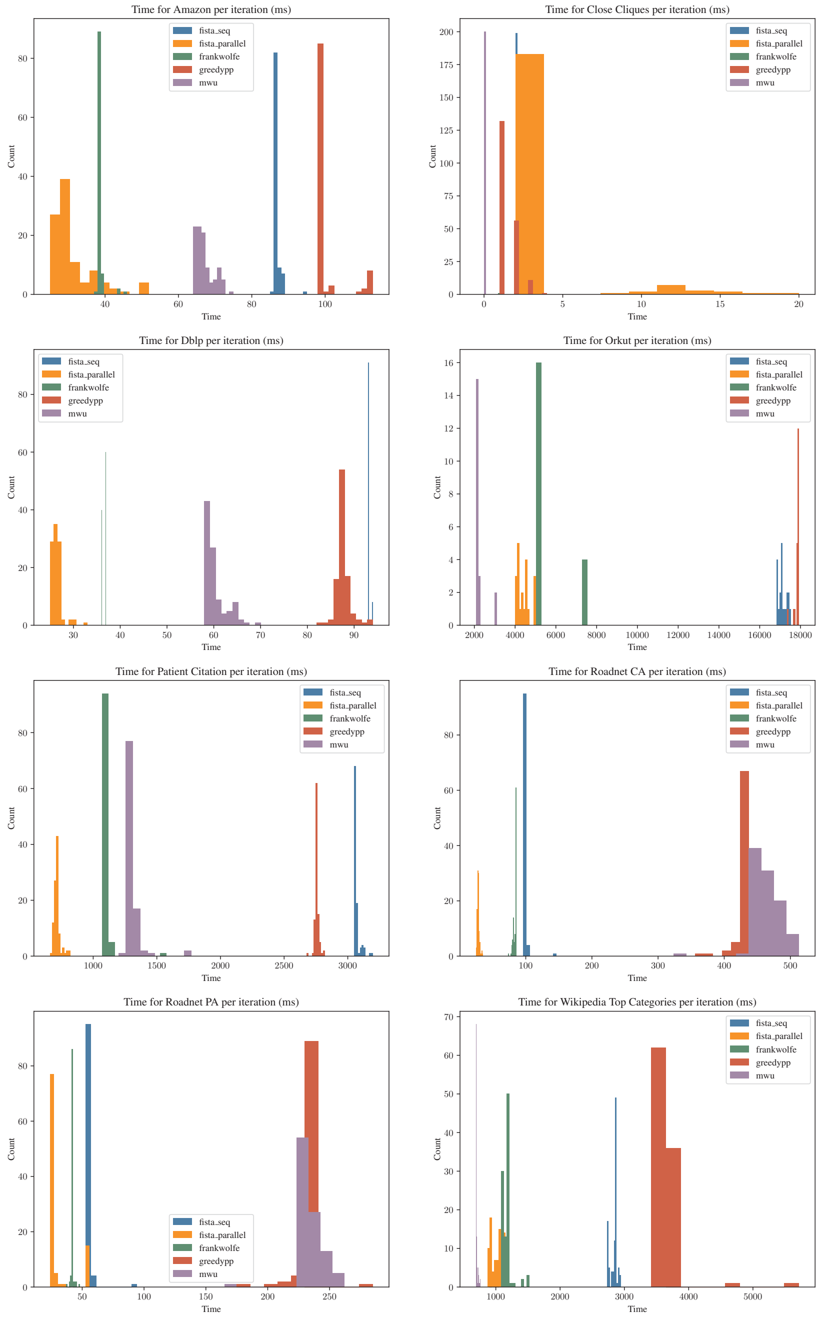

Figure 6.9: Time take per iteration histogram

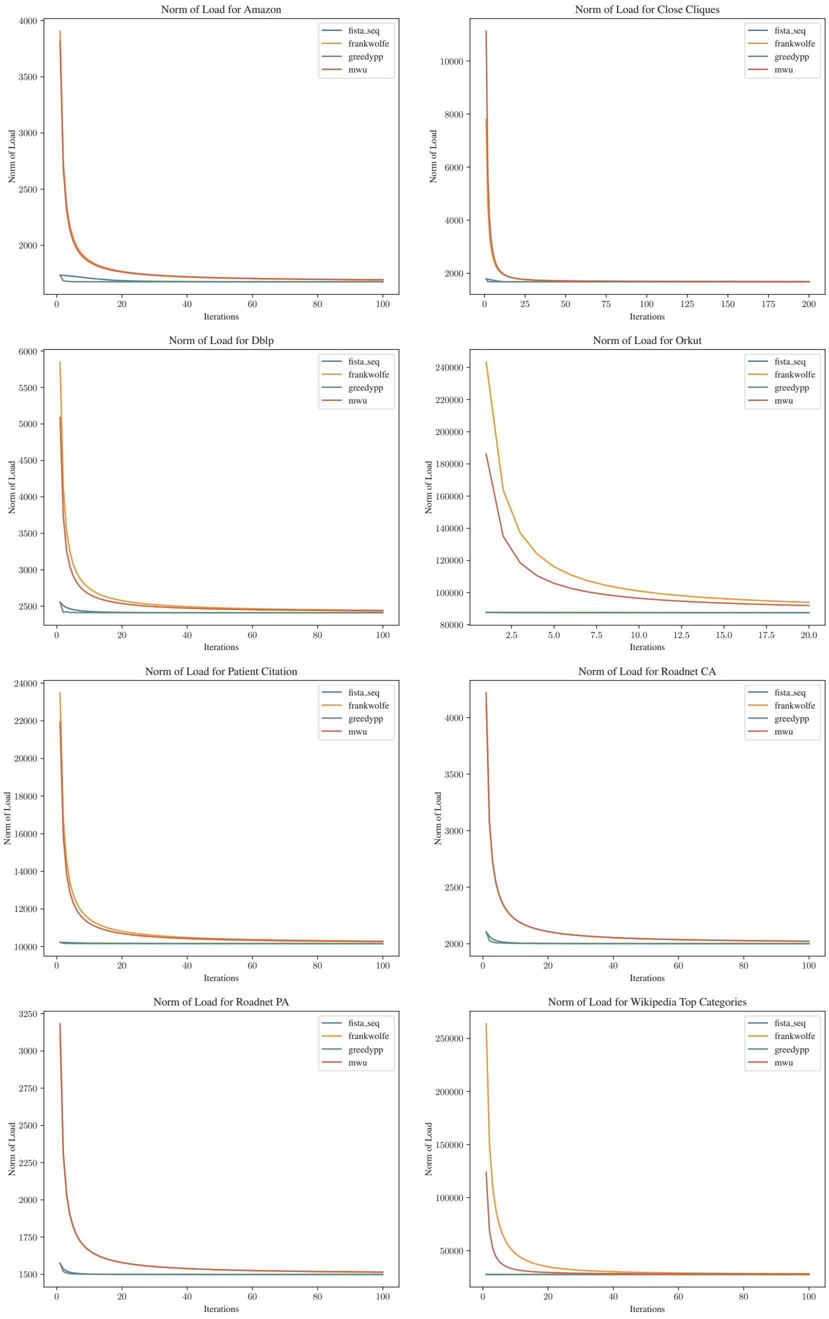

Figure 6.10: L_2 norm of load vector. See Figure 6.11 for a zoom-in on the last 20 iterations.

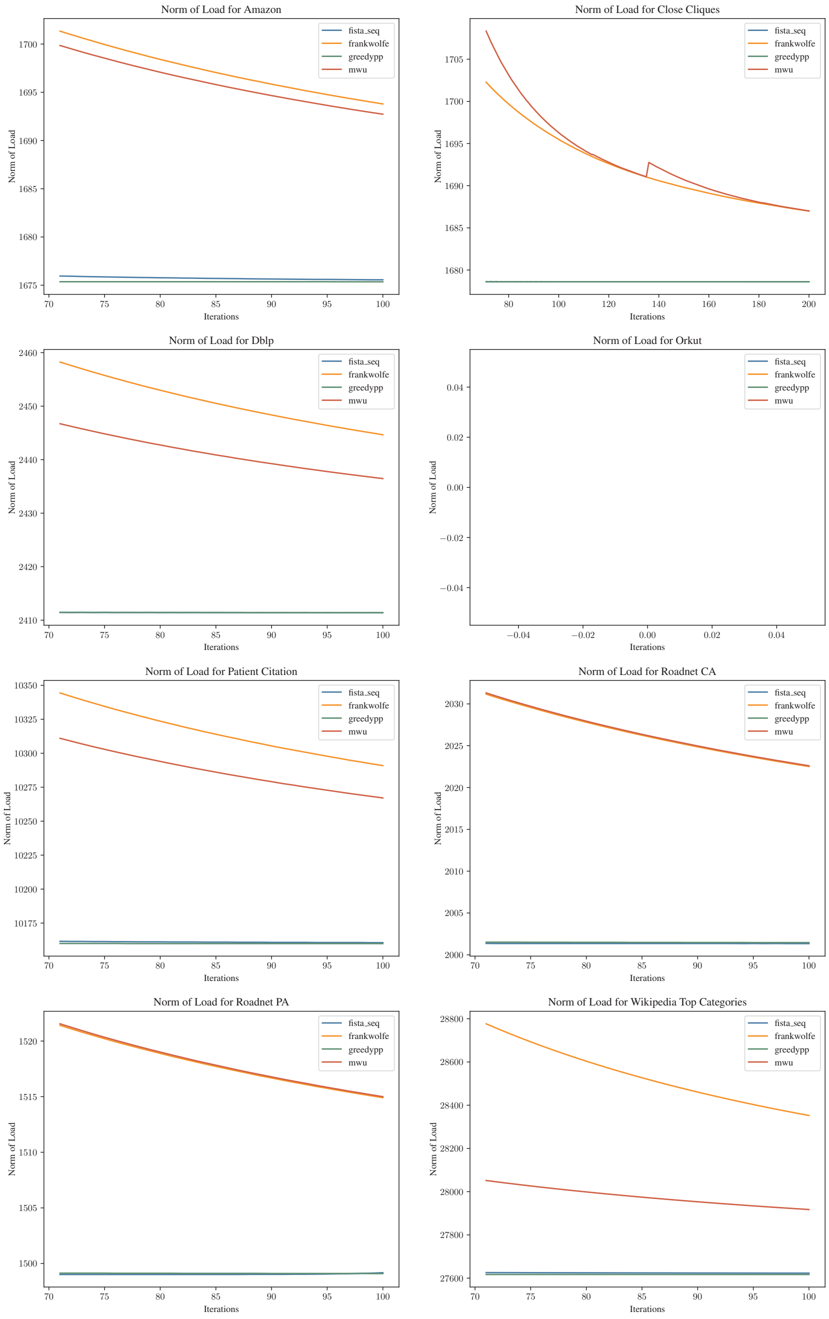

Figure 6.11: Same as Figure 6.10 but zoomed in from Iteration 70 and after.

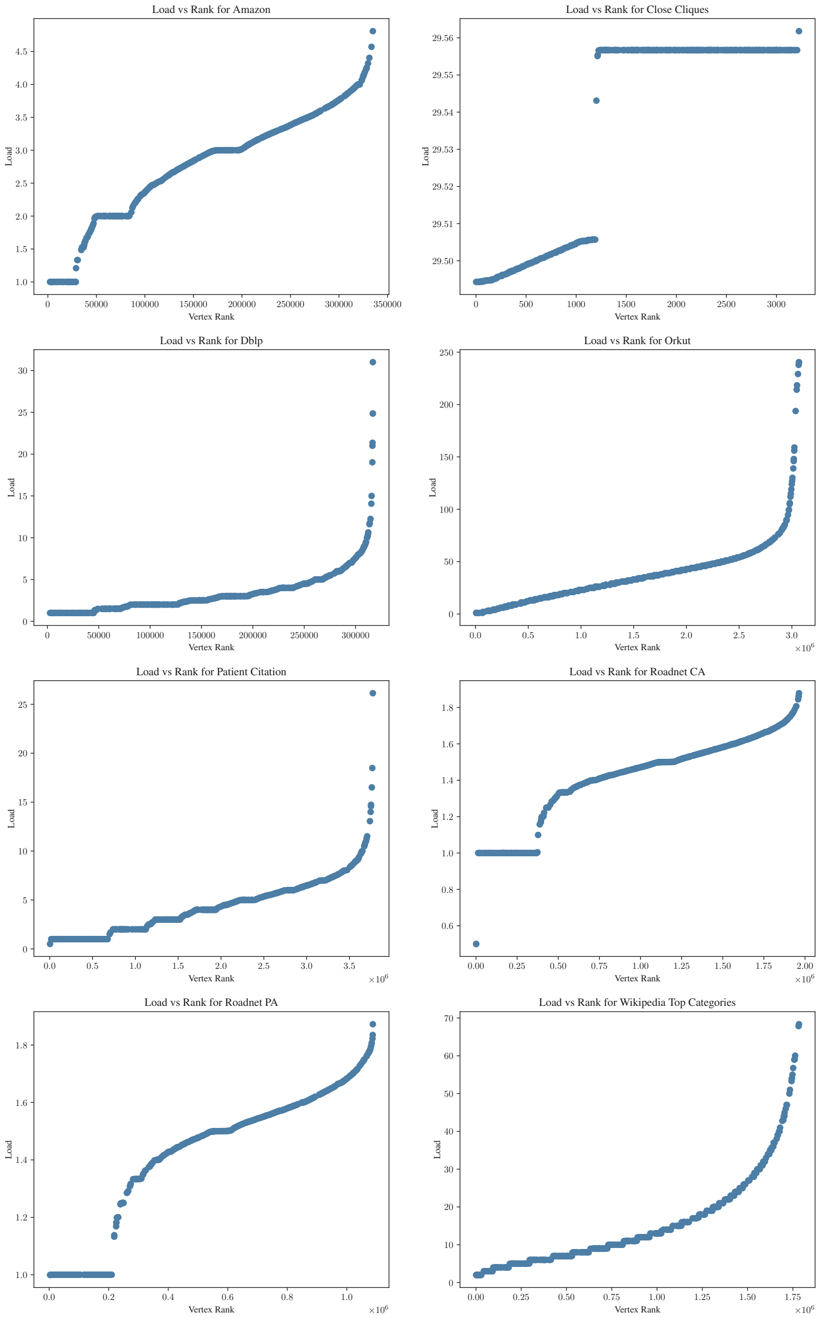

Figure 6.12: Scatter plot of sorted load vector. This can be used to approximate the densest at least k subgraph.

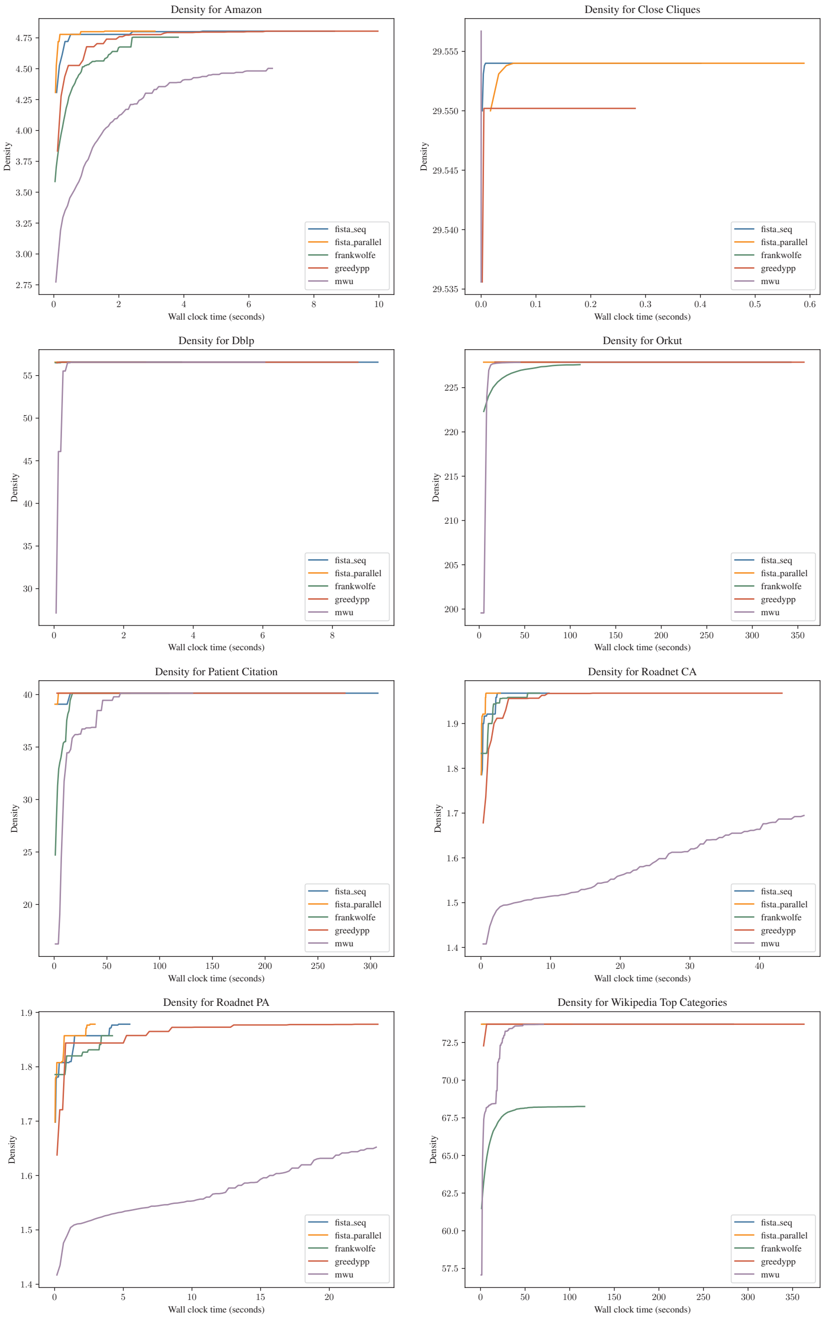

Figure 6.13: Wall clock time vs Maximum Density. See Figure 6.14 for a zoom in on first few seconds of each dataset.

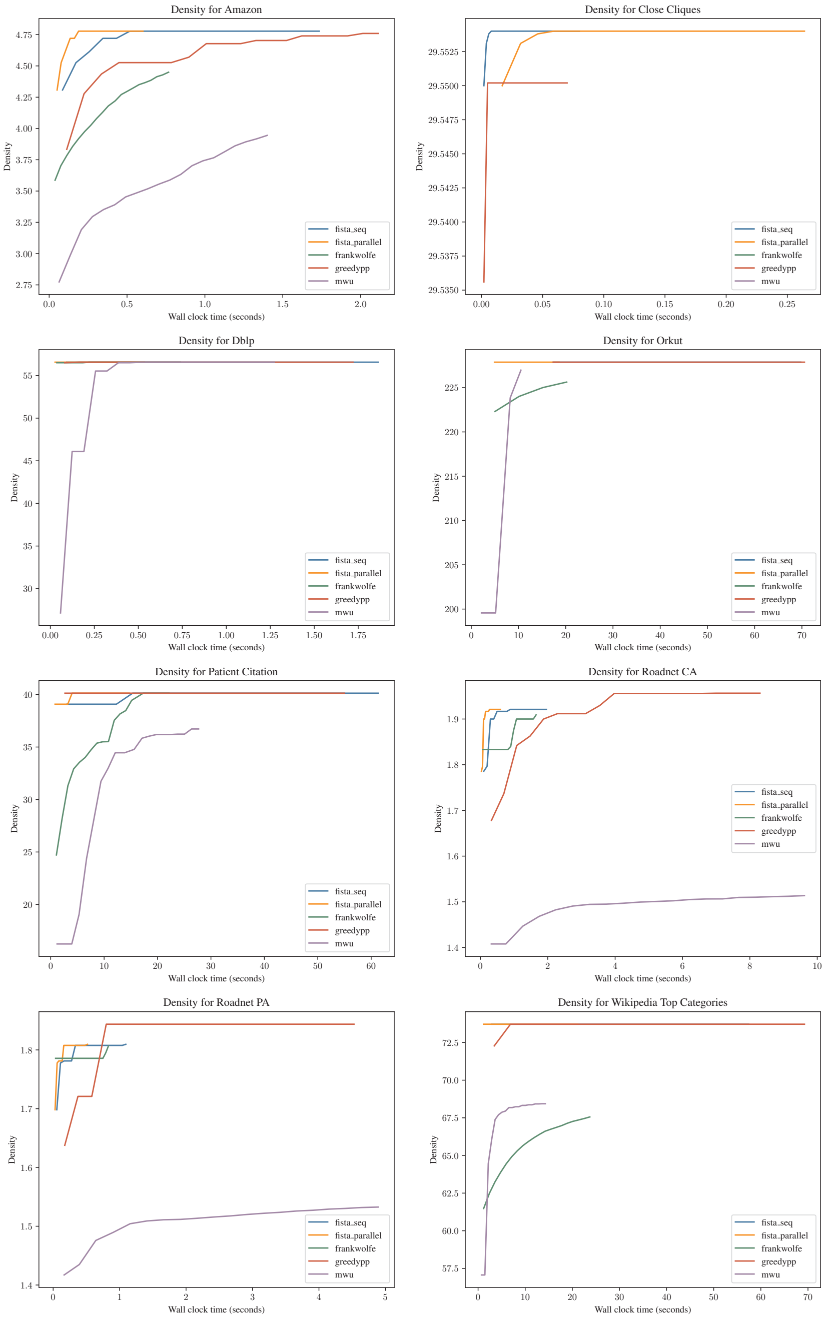

Figure 6.14: Wall clock time vs Maximum Density zoomed in on first few seconds for each dataset.