Security of Genus 3 Curves

Kim Laine
Joint work with Kristin Lauter

Microsoft Research, USA

September 28, 2015
ECC 2015, Bordeaux

Table of Contents

Basics

Recall:

- $C/\mathbb{F}_q$ a curve of genus 3
- $\mathrm{Jac}_C(\mathbb{F}_q)$ is an abelian group of size roughly q^3
- Fix $P_0 \in C(\overline{\mathbb{F}}_q)$
- Elements of $\mathrm{Jac}_C(\mathbb{F}_q)$ (reduced representation):

$$\left\{[P_1] + [P_2] + [P_3] - 3[P_0] \mid P_i \in C(\overline{\mathbb{F}}_q)\right\} \text{ invariant under } \mathrm{Frob}_q$$

- **Basically:** Points of $\mathrm{Jac}_C(\mathbb{F}_q)$ are triples of points on C

Question: Is $\mathrm{Jac}_C(\mathbb{F}_q)$ good for DLP-based crypto?

- Pollard rho complexity: $\widetilde{O}(q^{3/2})$ field multiplications
- For 128 bits of security against Pollard rho: $\log_2 q \approx 85$
- Use this for DLP-based crypto with small fields?

Hyperelliptic vs. Non-Hyperelliptic in Genus 3:

Non-Hyperelliptic curves

- *Exactly* the smooth plane quartics over $\mathbb{F}_q$
- Most genus 3 curves
- Diem: Index calculus complexity $\widetilde{O}(q)$
- For security $\log_2 q \geq 128$???

Hyperelliptic curves

- (Singular) plane models $y^2 = P(x)$, $P(x)$ a degree 7 polynomial
- Easier arithmetic
- Gaudry, Thomé, Thériault, Diem: Index calculus $\widetilde{O}(q^{4/3})$
- For security $\log_2 q > 95$???

Isogeny attacks:

$C/\mathbb{F}_q$ hyperelliptic genus 3

Explicit isogeny $f : \mathrm{Jac}_C \to \mathcal{A}$

- $\mathcal{A}/\mathbb{F}_q$ a 3-dimensional principally polarized abelian variety
- $\mathcal{A}$ *almost certainly* isomorphic to Jacobian of some non-hyperelliptic genus 3 curve C'
- Difficult but possible to construct such maps explicitly (Ben Smith, Damien Robert, David Lubicz, etc.)

Attack the non-hyperelliptic problem in time $\widetilde{O}(q)$

But how feasible is the $\widetilde{O}(q)$-attack for practical size q?

Hyperelliptic vs. Non-Hyperelliptic in Genus 3:

Non-Hyperelliptic curves

- *Exactly* the smooth plane quartics over $\mathbb{F}_q$
- Most genus 3 curves
- Claus Diem: Index calculus complexity $\widetilde{O}(q)$
- For security $\log_2 q \geq 128$???

Hyperelliptic curves

- (Singular) plane models $y^2 = P(x)$, $P(x)$ a degree 7 polynomial
- Easier arithmetic
- Index calculus complexity $\widetilde{O}(q^{4/3})$
- Isogeny attacks reduce complexity to $\widetilde{O}(q)$
- For security $\log_2 q > 128$???

Our Variant of Diem's Index Calculus

- $C/\mathbb{F}_q$ non-hyperelliptic genus 3
- On $\mathrm{Jac}_C(\mathbb{F}_q)$ consider DLP $D_1 = x \cdot D_2$ where

$$D_1 := [P_1^1] + [P_2^1] + [P_3^1] - 3[P_0]$$

$$D_2 := [P_1^2] + [P_2^2] + [P_3^2] - 3[P_0]$$

- For simplicity, suppose $P_i^j \in C(\mathbb{F}_q)$
- For simplicity, suppose subgroup $\langle D_2 \rangle$ has prime order
- Suppose the DLP has a solution.

Theorem 1 (Diem)

There is a probabilistic algorithm for solving the DLP using $\widetilde{O}(q)$ operations in $Jac_C(\mathbb{F}_q)$.

- Several generalization and variations exist (by Diem and others).
- In genus 3 all have complexity $\widetilde{O}(q)$.
- Double Large Prime Index Calculus

Double large prime index calculus briefly:

1. DLP $Q = x \cdot P$ in abelian group G, $Q, P \in G$
2. Choose subset of elements $\mathcal{F}$ called factor base, so that $Q, P \in \mathcal{F}$
3. Elements in $G \setminus \mathcal{F}$ are called large primes
4. Collect linear relations in G with at most two large primes
5. When possible, eliminate the two large primes from the relations to produce full relations, involving only factor base elements
6. After enough full relations (involving P, Q) have been found, use linear algebra to express Q in terms of P

The case of non-hyperelliptic genus 3:

- $\mathcal{F}$ a subset of $\widetilde{O}(q^{1/2})$ points of $C(\mathbb{F}_q)$ instead of elements of $\mathrm{Jac}_C(\mathbb{F}_q)$ (Note: $\#C(\mathbb{F}_q) \approx q$)
- So DLP is a formal sum of points in $C(\mathbb{F}_q)$
- **Key idea:** To find relations, intersect the curve with a line (4 points of intersection)
- Sum of points in the intersection is "zero" (roughly speaking)
- Need relation involving $\mathcal{F}$, so form lines through two $\mathcal{F}$-points
- With high probability other two points are also in $C(\mathbb{F}_q)$; most likely not in $\mathcal{F}$

Theorem 2 (Diem-Thomé)

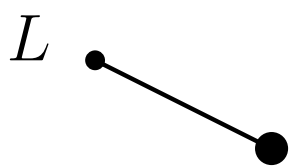
Let $L/\mathbb{F}_q$ be a generic line in $\mathbb{P}^2$ through two $\mathbb{F}_q$-points of C. Then $L \cap C$ consists of four $\mathbb{F}_q$-points with probability $1/2 + O(q^{-1/2})$.

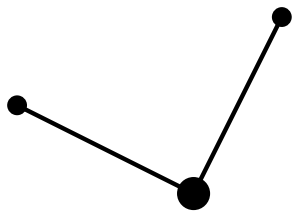
Intersection operation: Given two $\mathbb{F}_q$-points of C and line L through them, compute $L \cap C$. With probability $1/2$ return the two new $\mathbb{F}_q$-points on C.

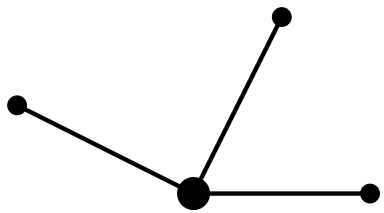
The information in these intersections is stored as a graph.

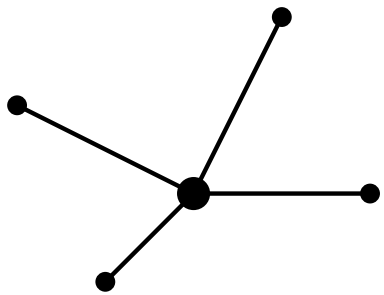
Special vertex (represents factor base elements)

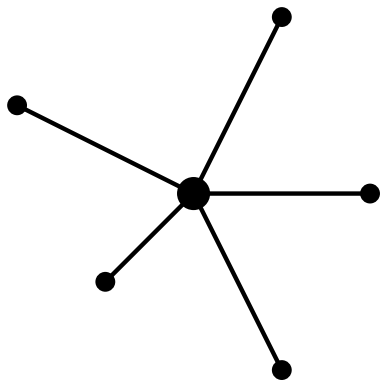
Intersection $L \cap C$:
$F_i,\ F_j,\ F,\ L$ where $F_i, F_j, F \in \mathcal{F}$



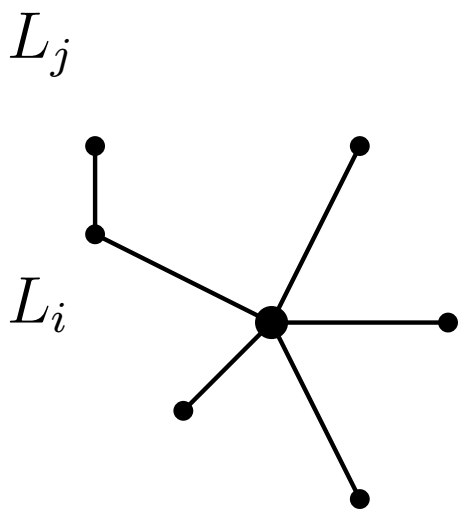


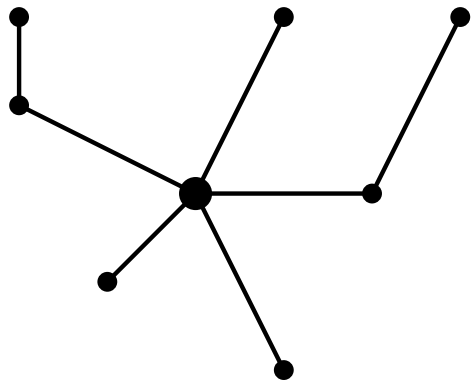


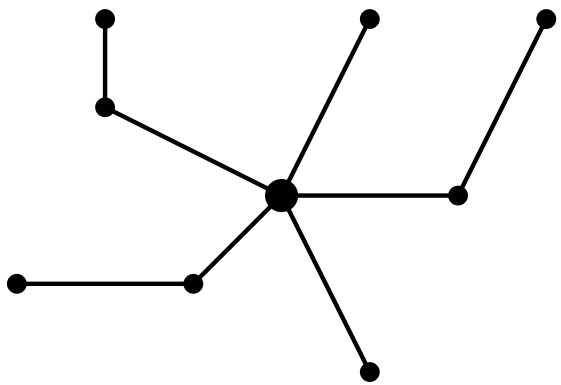


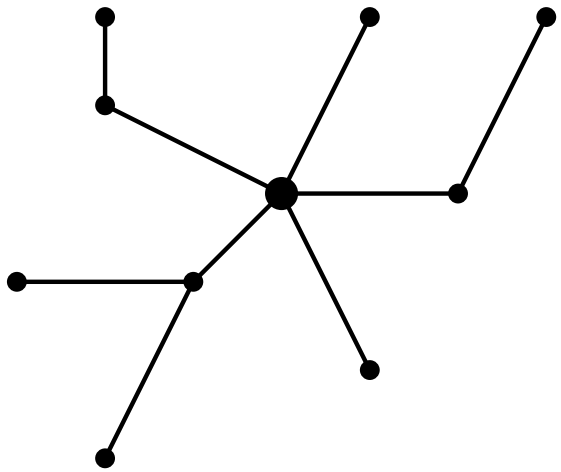


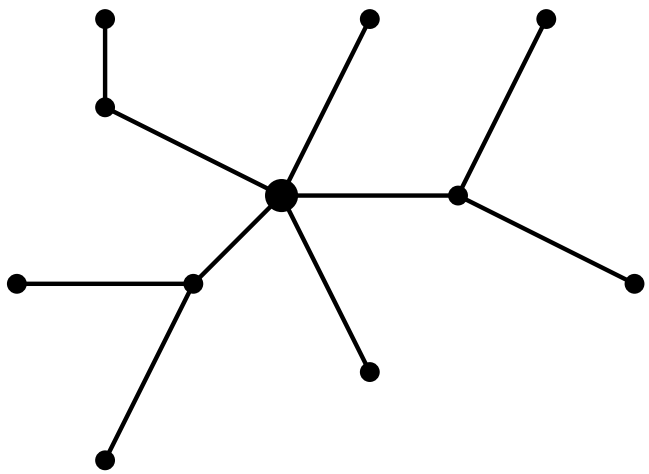
Intersection $L \cap C$:
$F_i,\ F_j,\ L_i,\ L_j$ where $L_i \in \mathbf{V}, L_j \notin \mathbf{V}$

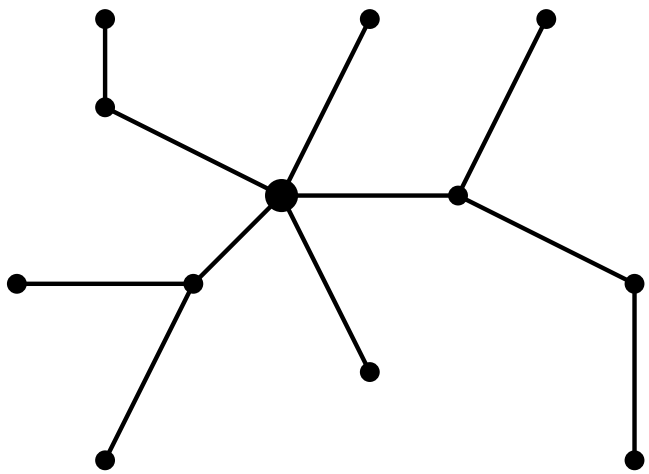


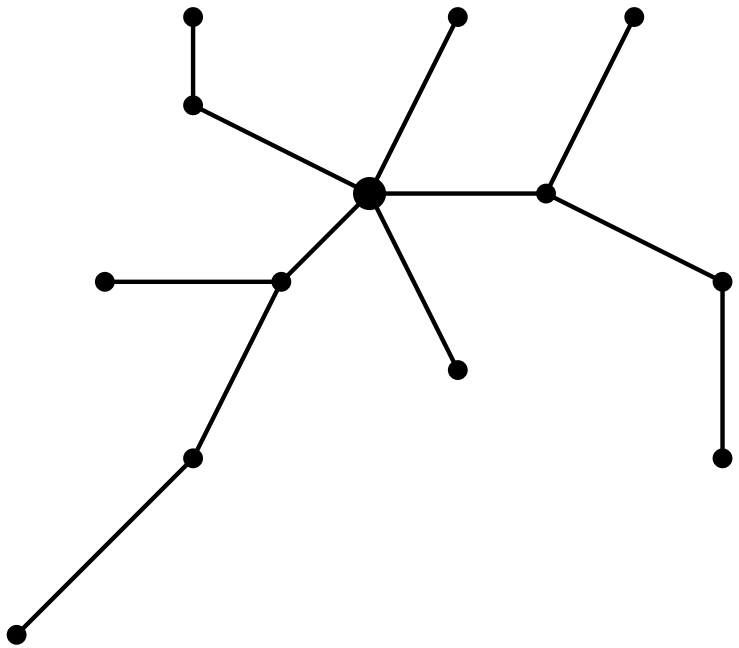


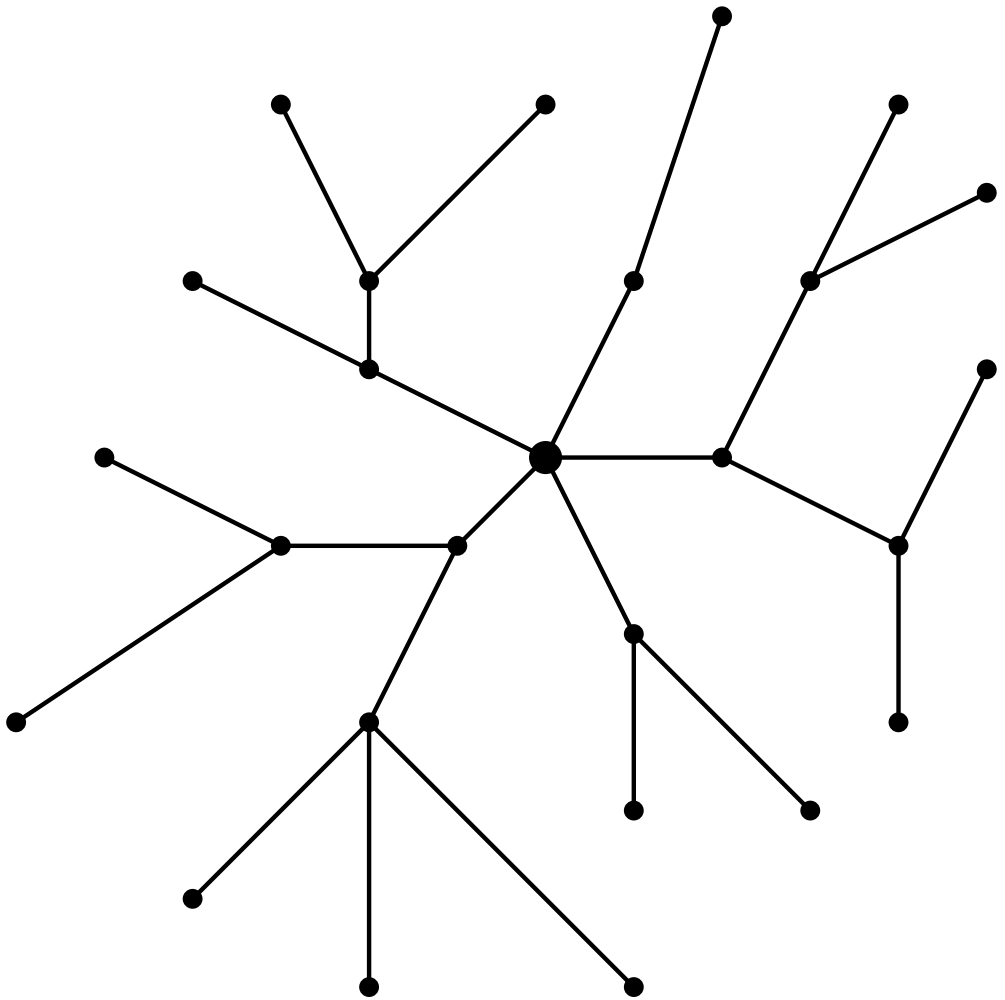






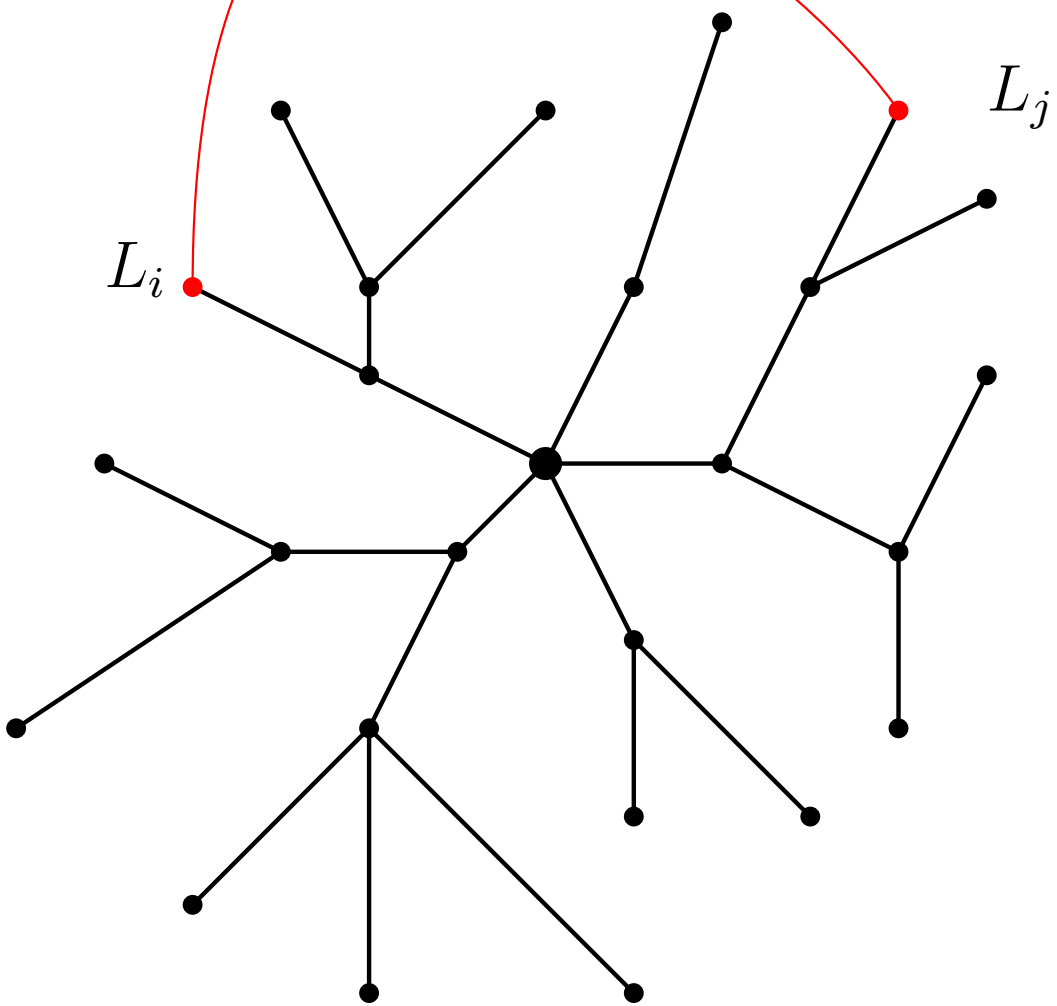


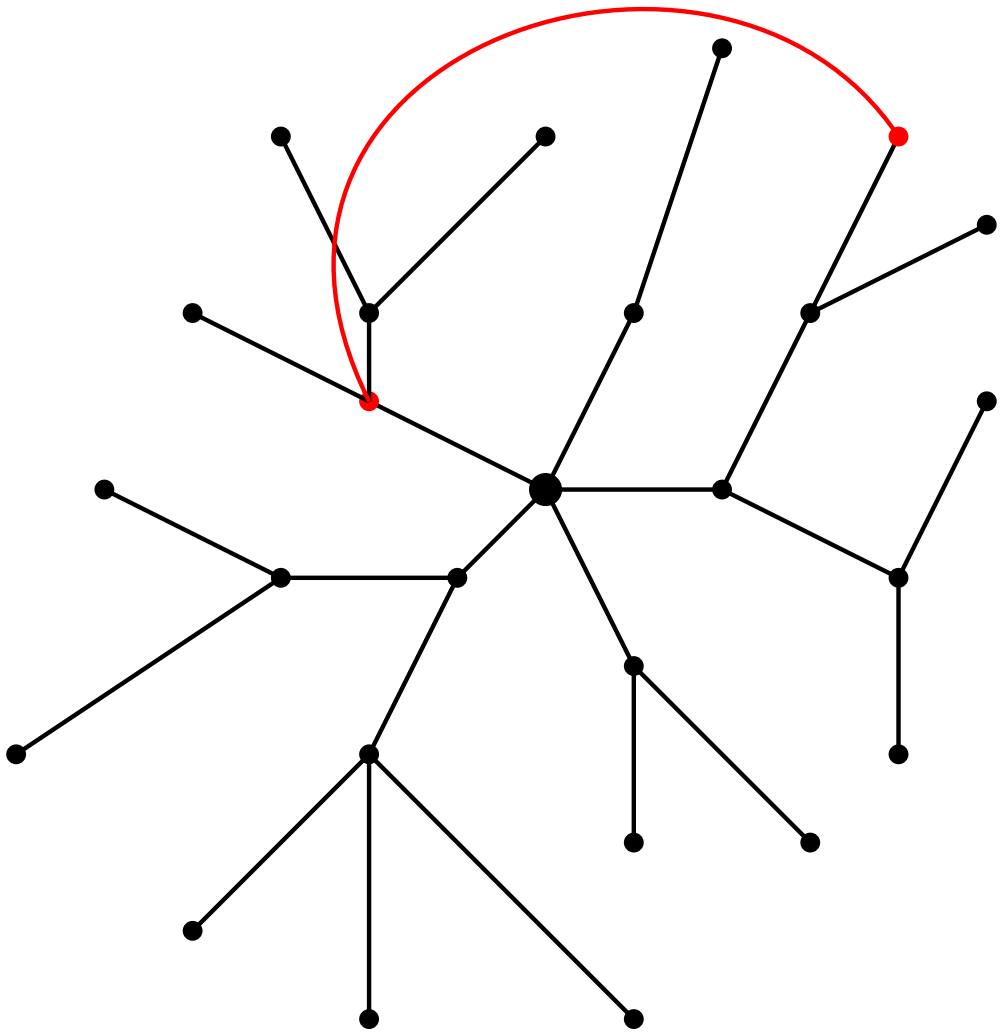


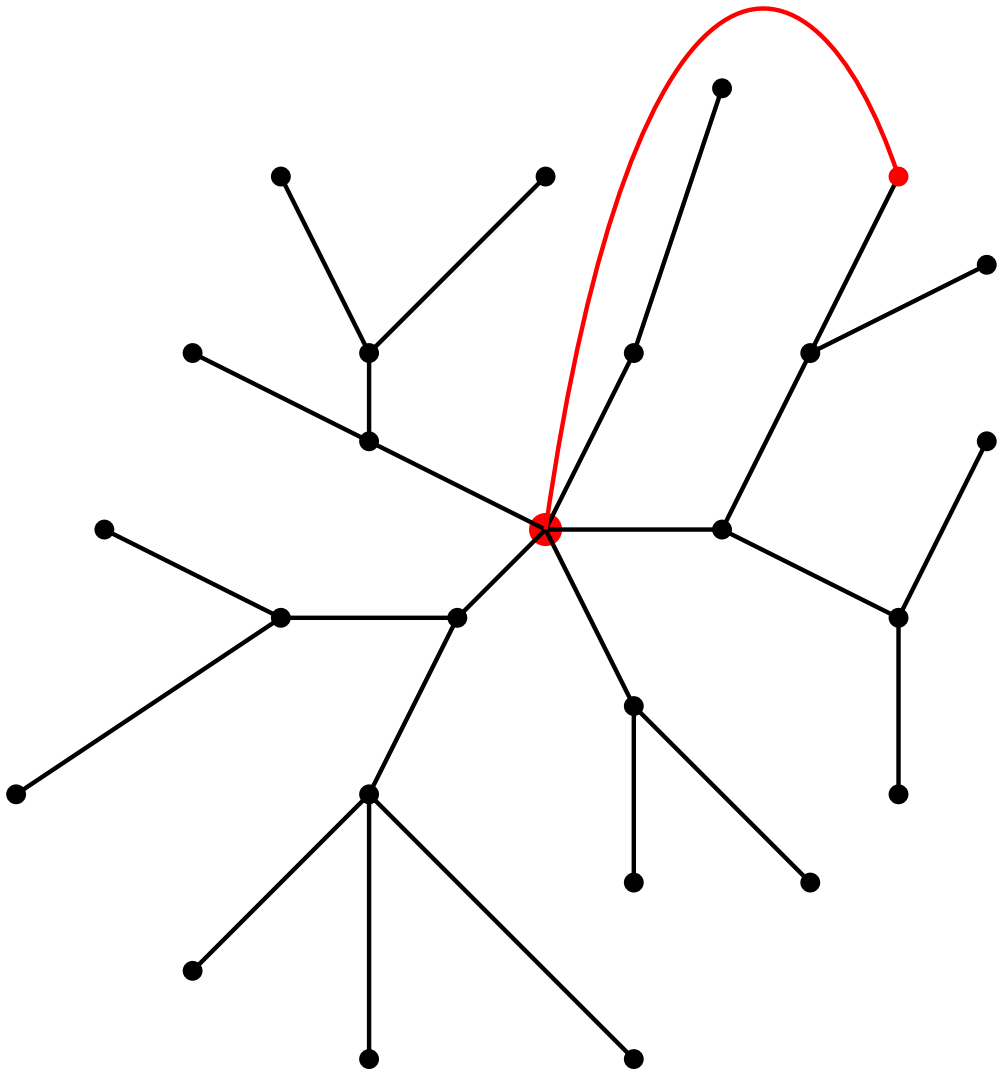


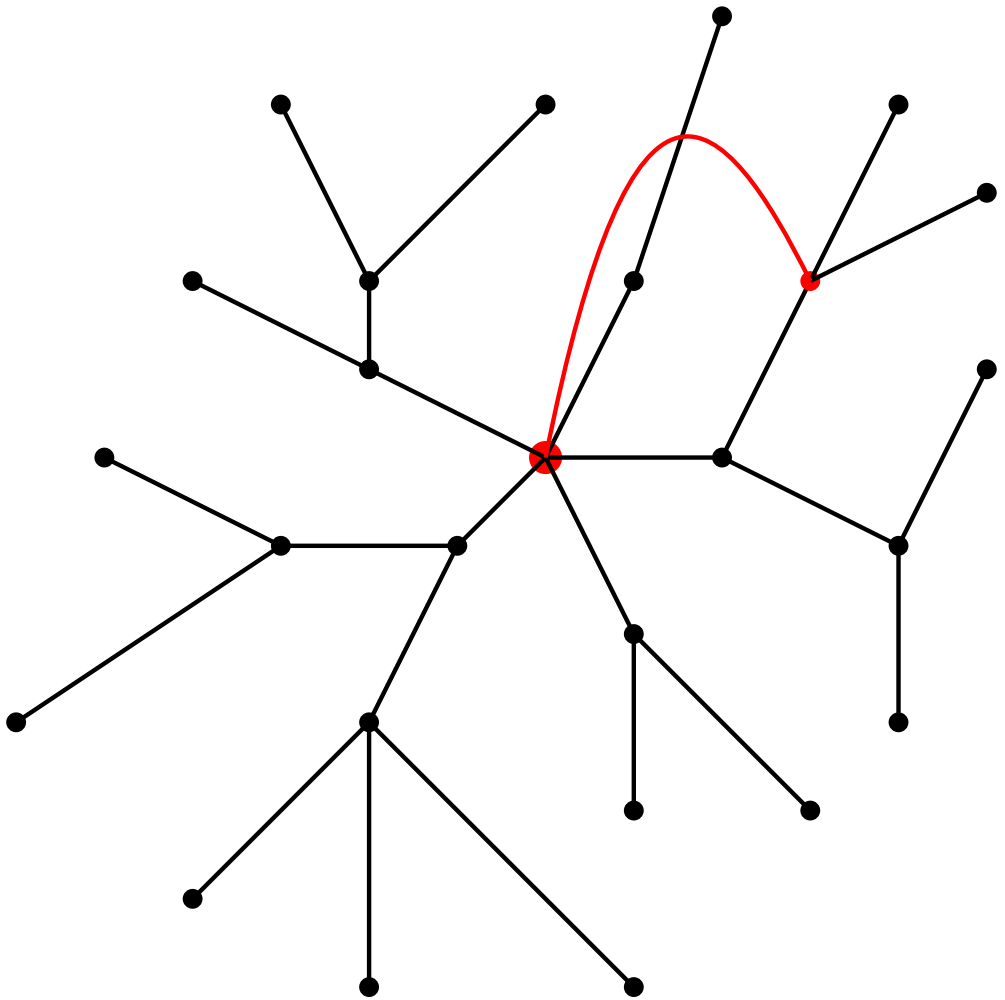
Intersection $L \cap C$:

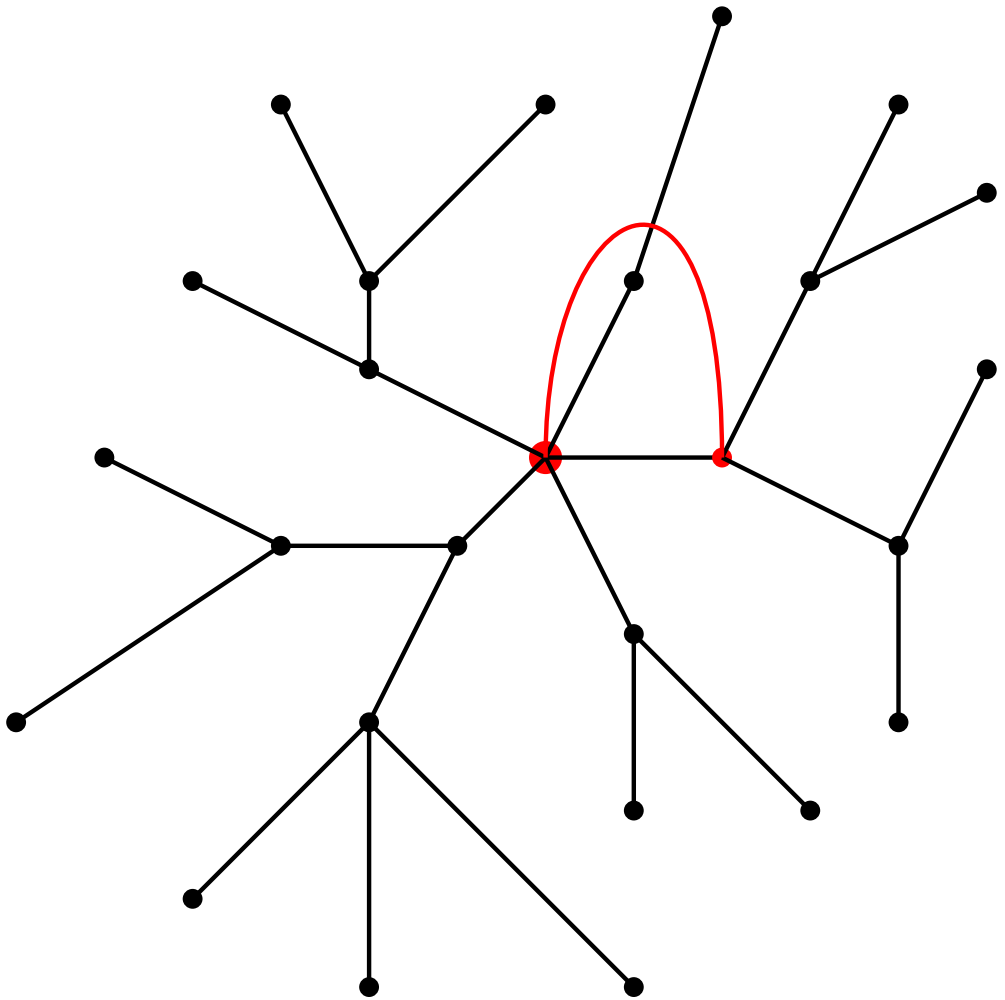
$F_i,\ F_j,\ L_i,\ L_j$ where $L_i, L_j \in \mathbf{V}$

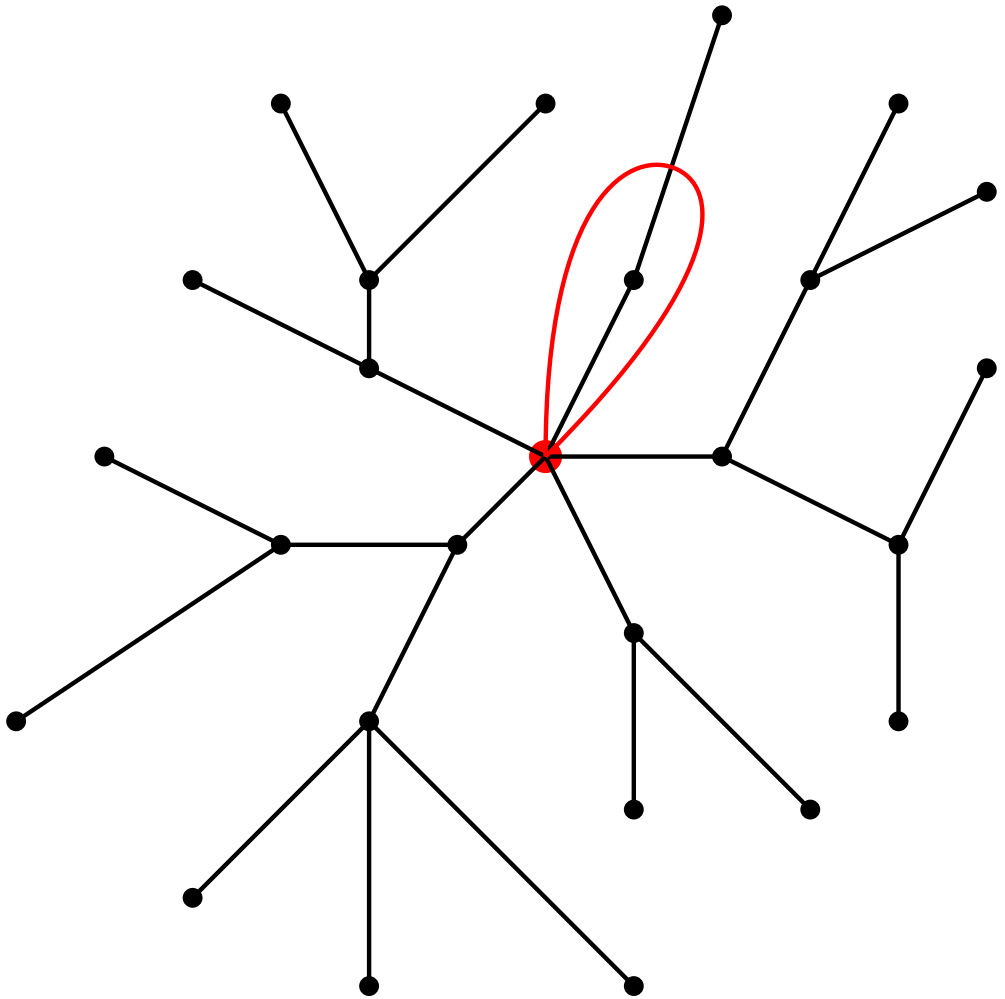












Our variant:

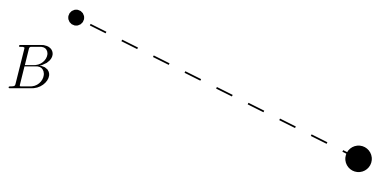
- Construct $\mathcal{F}$ iteratively starting from a much smaller initial set of points
- Graph becomes wider and shallower
- Options:
  1. Keep building graph until all relations are found
  2. Stop building graph at some point and continue with only relation search

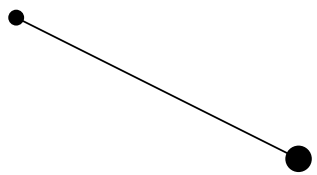
Constructing factor base iteratively:

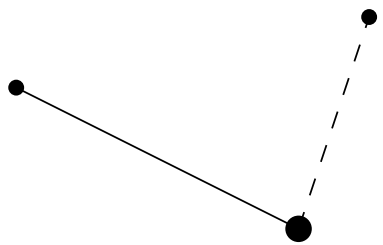
1. Let λ be a positive root of $\lambda \exp(4\lambda^8) = q^{1/8}$
2. Note: For reasonable size q, $1 < \lambda < 1.2$
3. Choose $\mathcal{RP}$ a set of $\lceil 4\lambda\, q^{1/8} \rceil$ $\mathbb{F}_q$-points on C
4. Let for now
$$\mathcal{F} := \mathcal{RP} \cup \{P_j^i\} \cup \{P_0\}$$

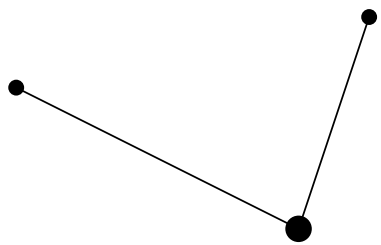
Recall: P_j^i, P_0 are the $C(\mathbb{F}_q)$-points appearing in the DLP

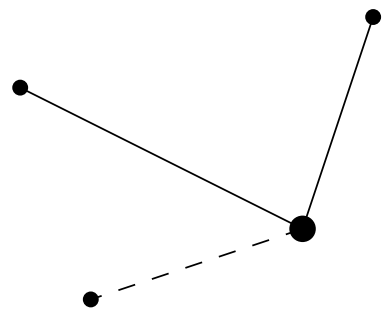
Intersection $L \cap C$:
F_i, F_j, F, B where $F_i, F_j \in \mathcal{RP}$
Added F to $\mathcal{F}$.

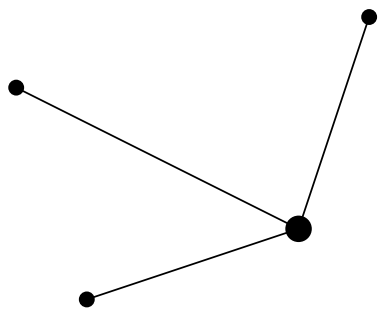


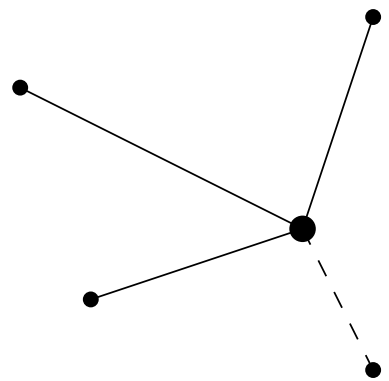


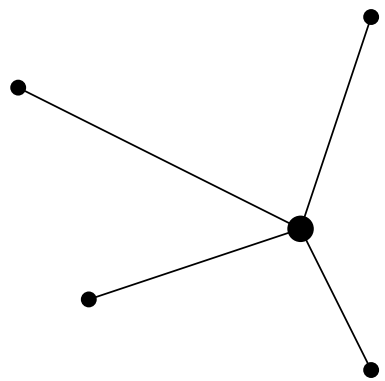


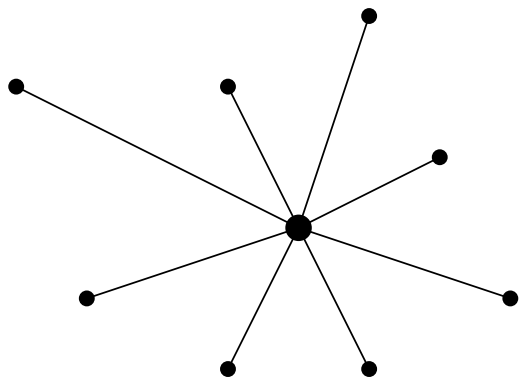




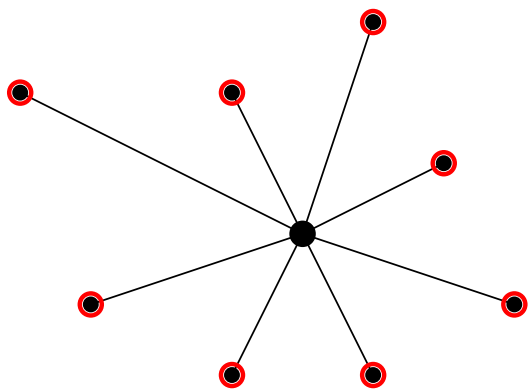


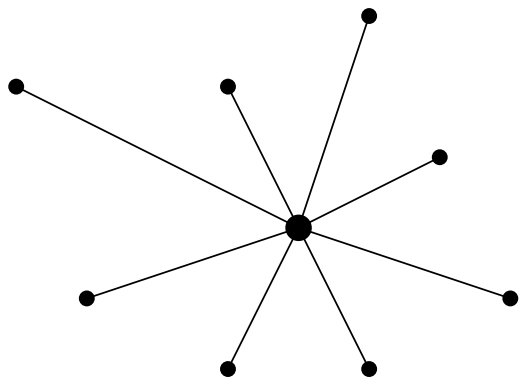




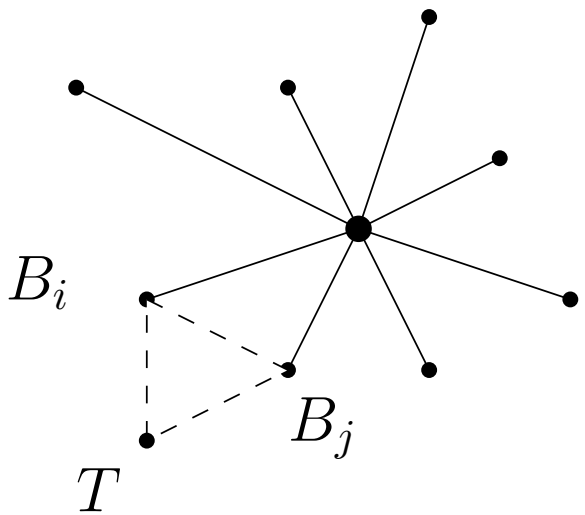


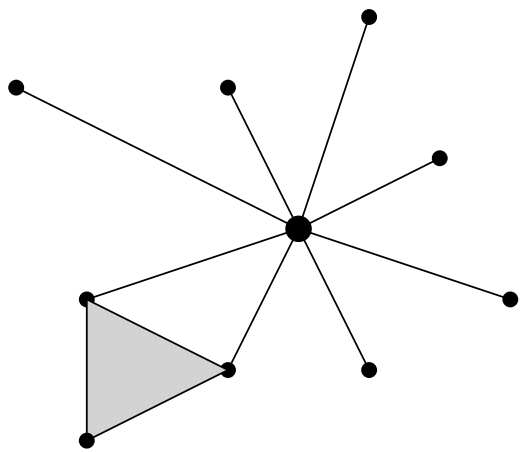
Base vertices

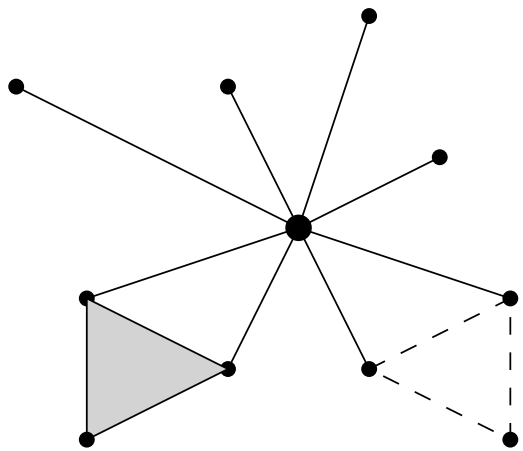


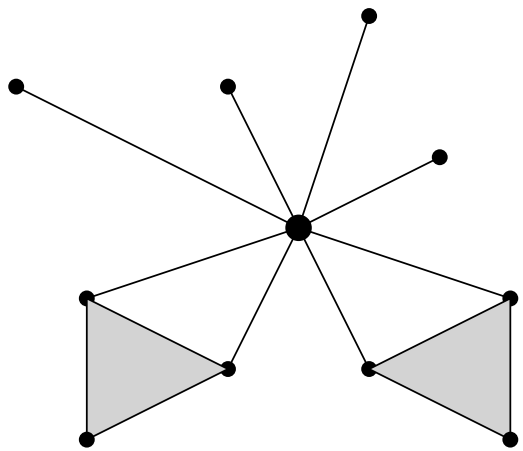


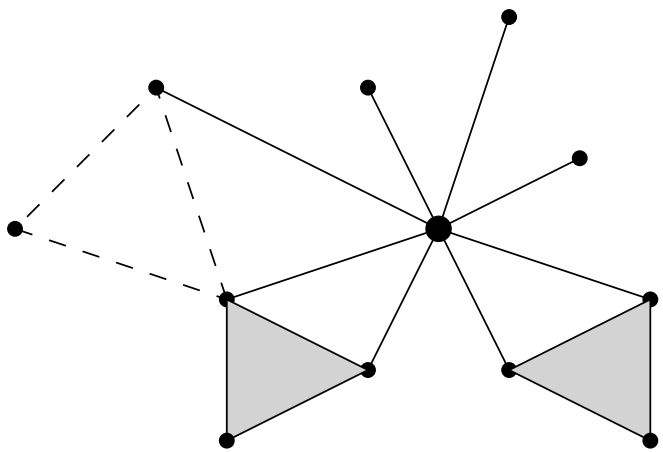
Intersection $L \cap C$:
B_i, B_j, F, T where $B_i, B_j \in \mathbf{B}$
Added F to $\mathcal{F}$.

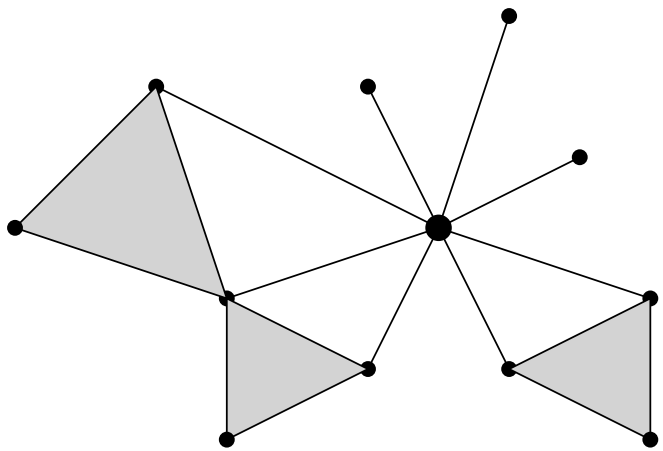


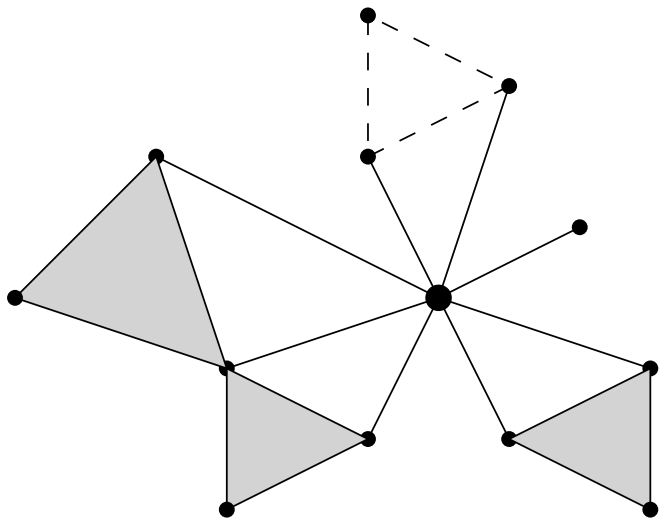


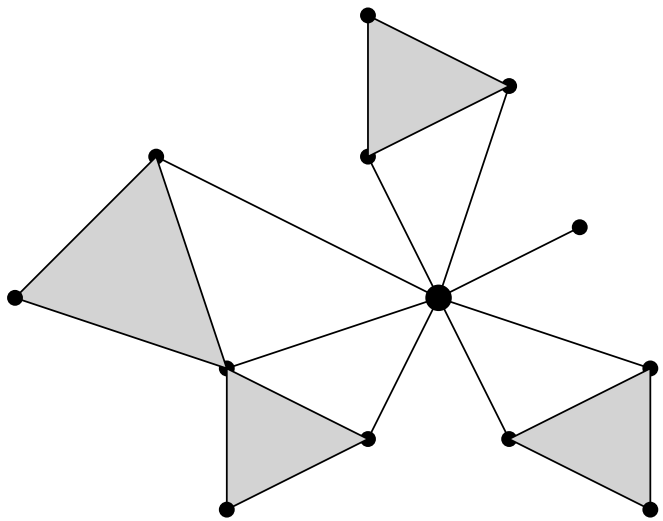


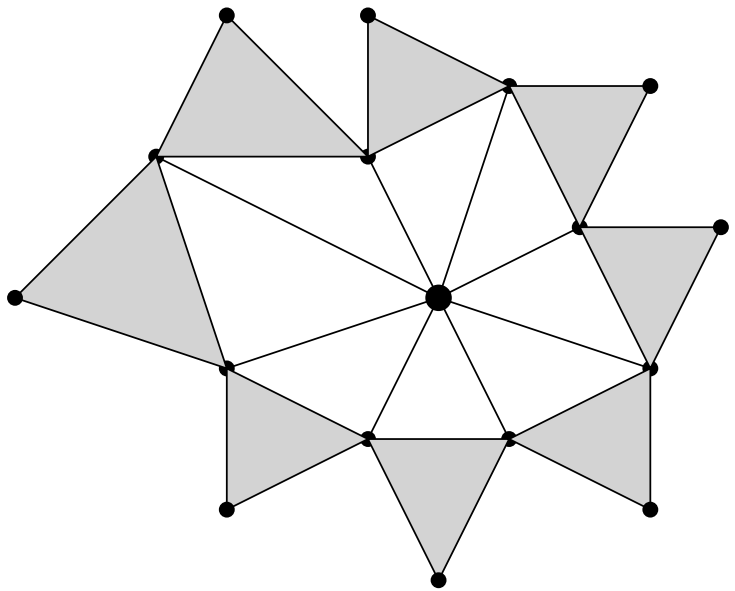




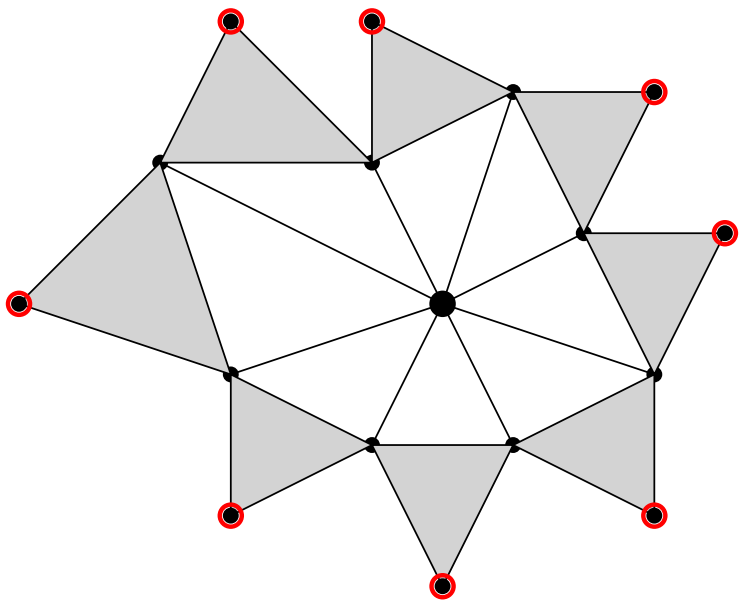








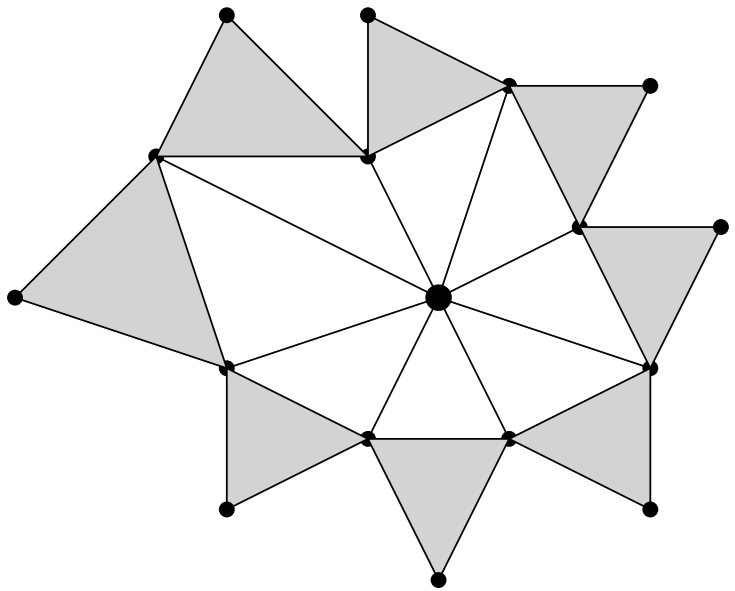
Top triangle vertices (not all drawn)



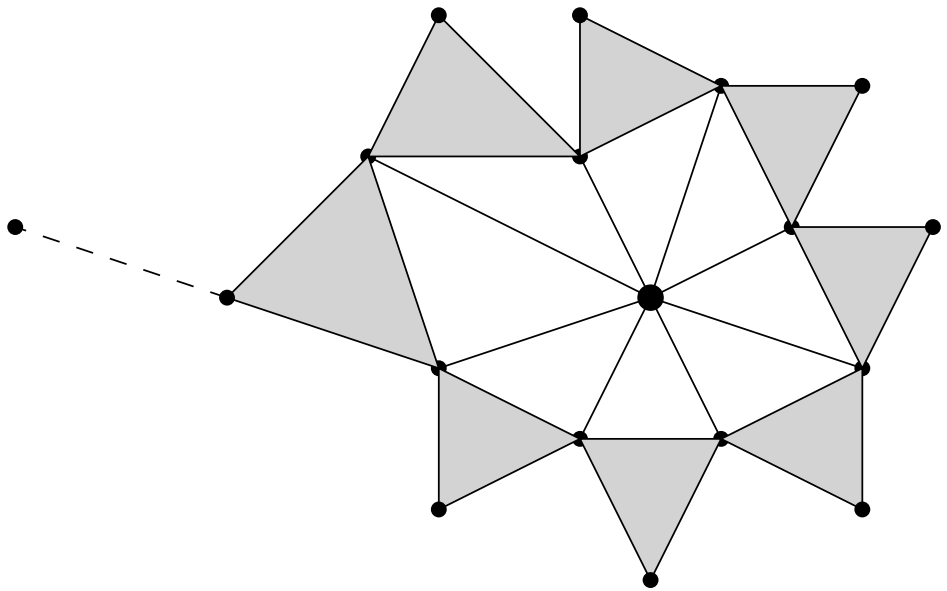
- Sizes of the various sets:

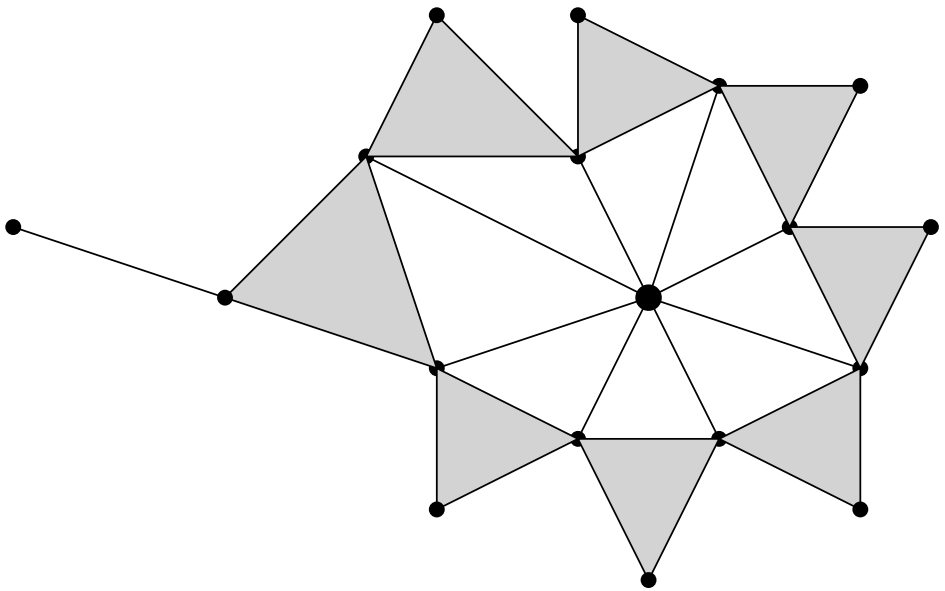
$$\#\mathcal{RP} = \widetilde{O}(q^{1/8}), \quad \#\mathbf{B} = \widetilde{O}(q^{1/4}), \quad \#\mathcal{F} = \widetilde{O}(q^{1/2})$$

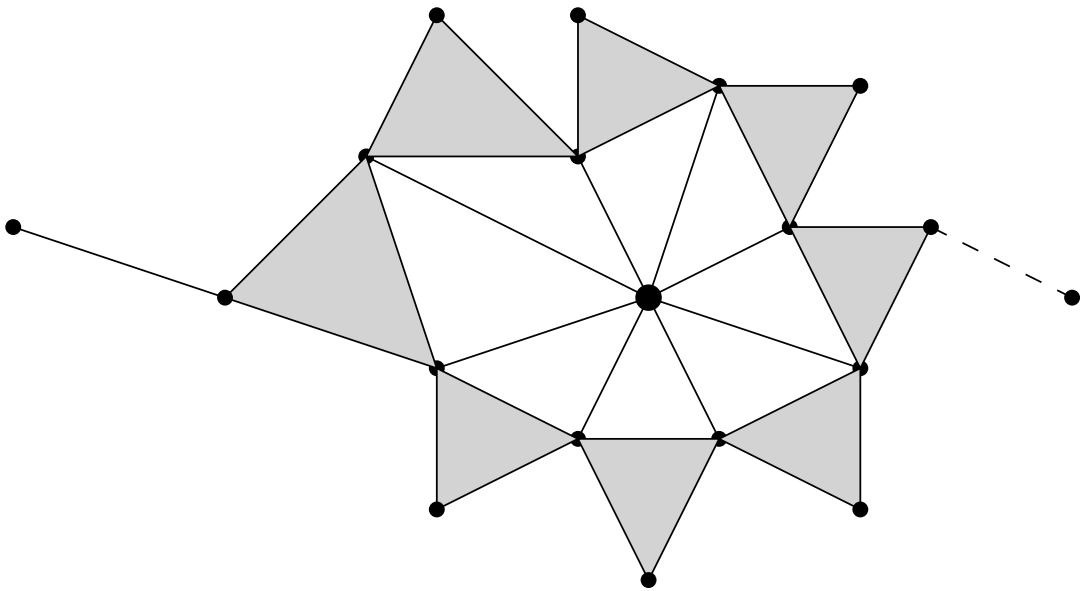
- Use $\mathcal{F}$ as before to build graph and construct full relations
- Graph contains already $\widetilde{O}(q^{1/2})$ points so grows faster than in traditional approach
- Graph is very wide near the root (results in sparser matrix)

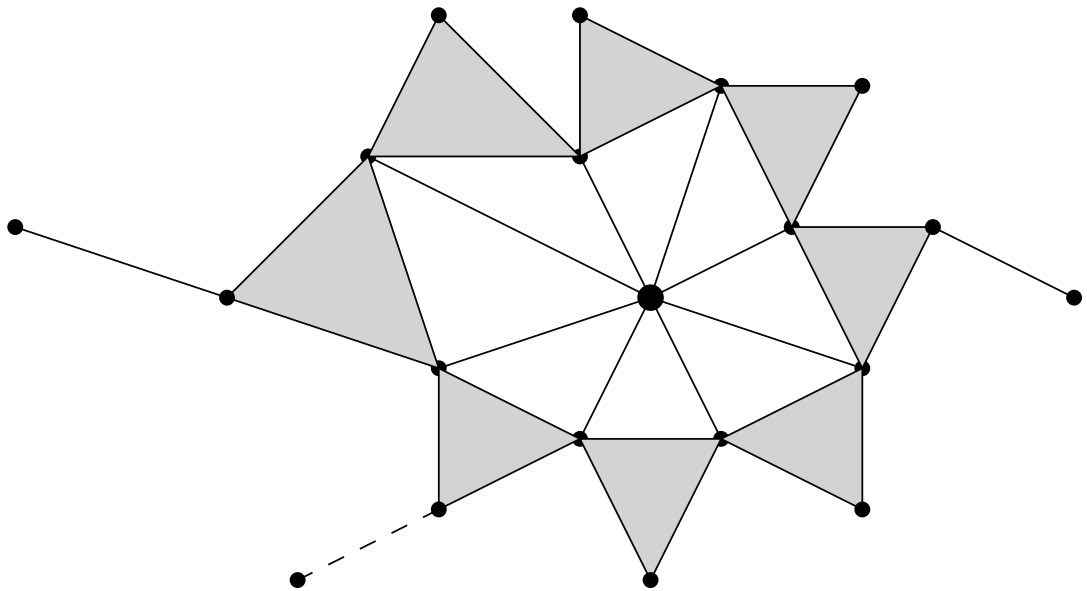


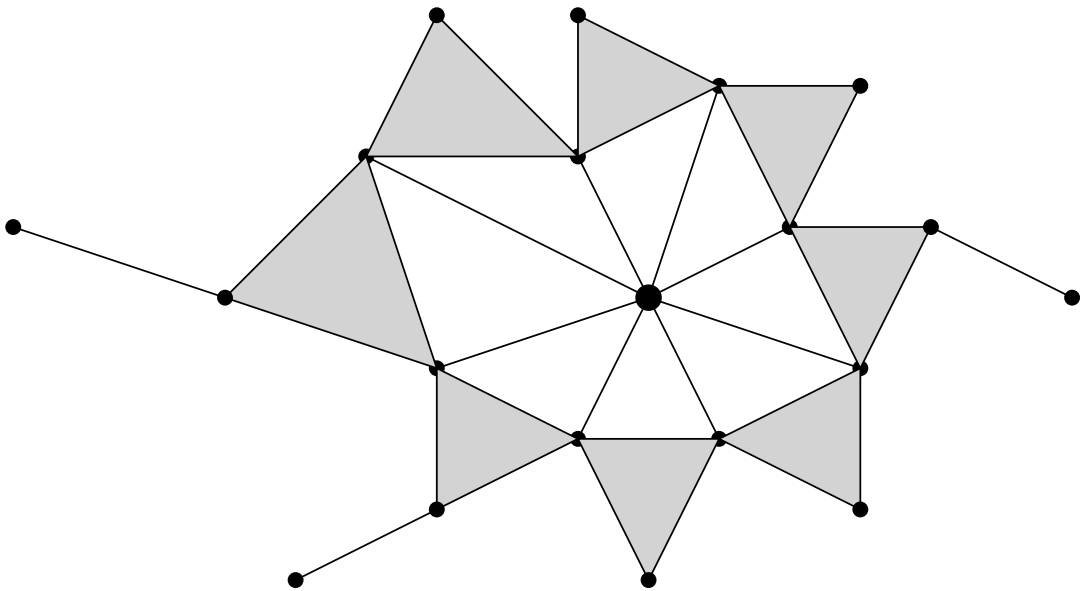
Intersection $L \cap C$:
F_i, F_j, L_i, L_j where $L_i \in \mathbf{V}$, $L_j \notin \mathbf{V}$

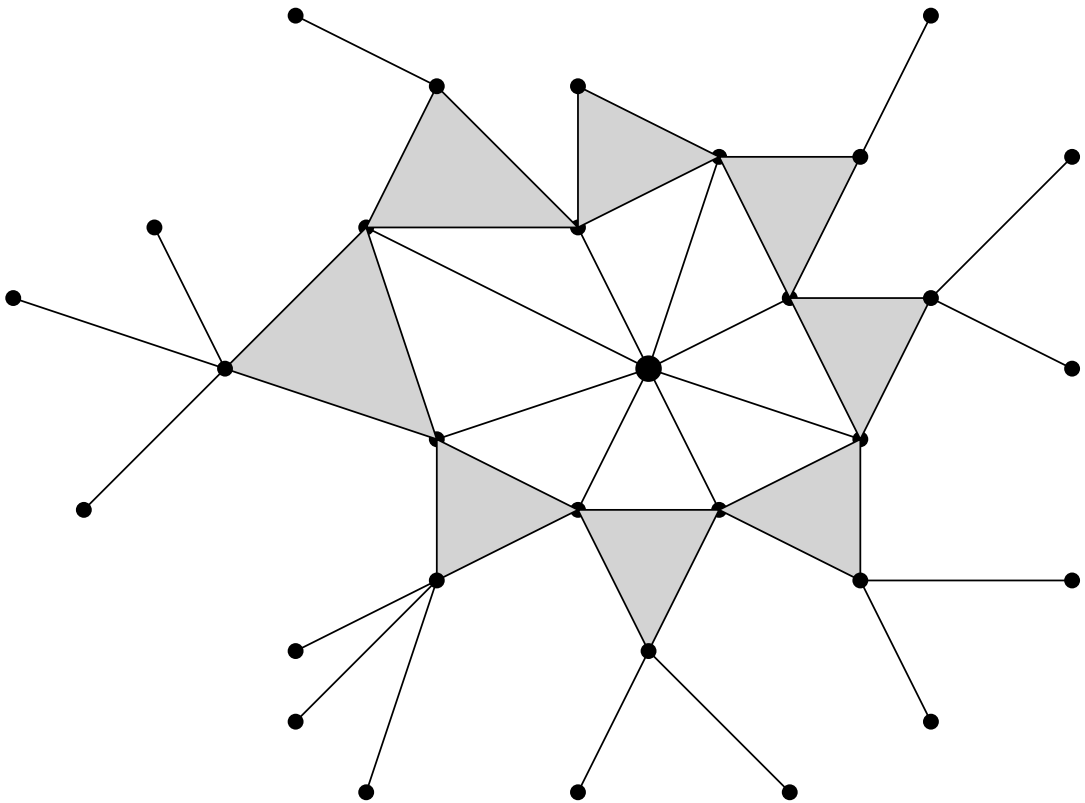


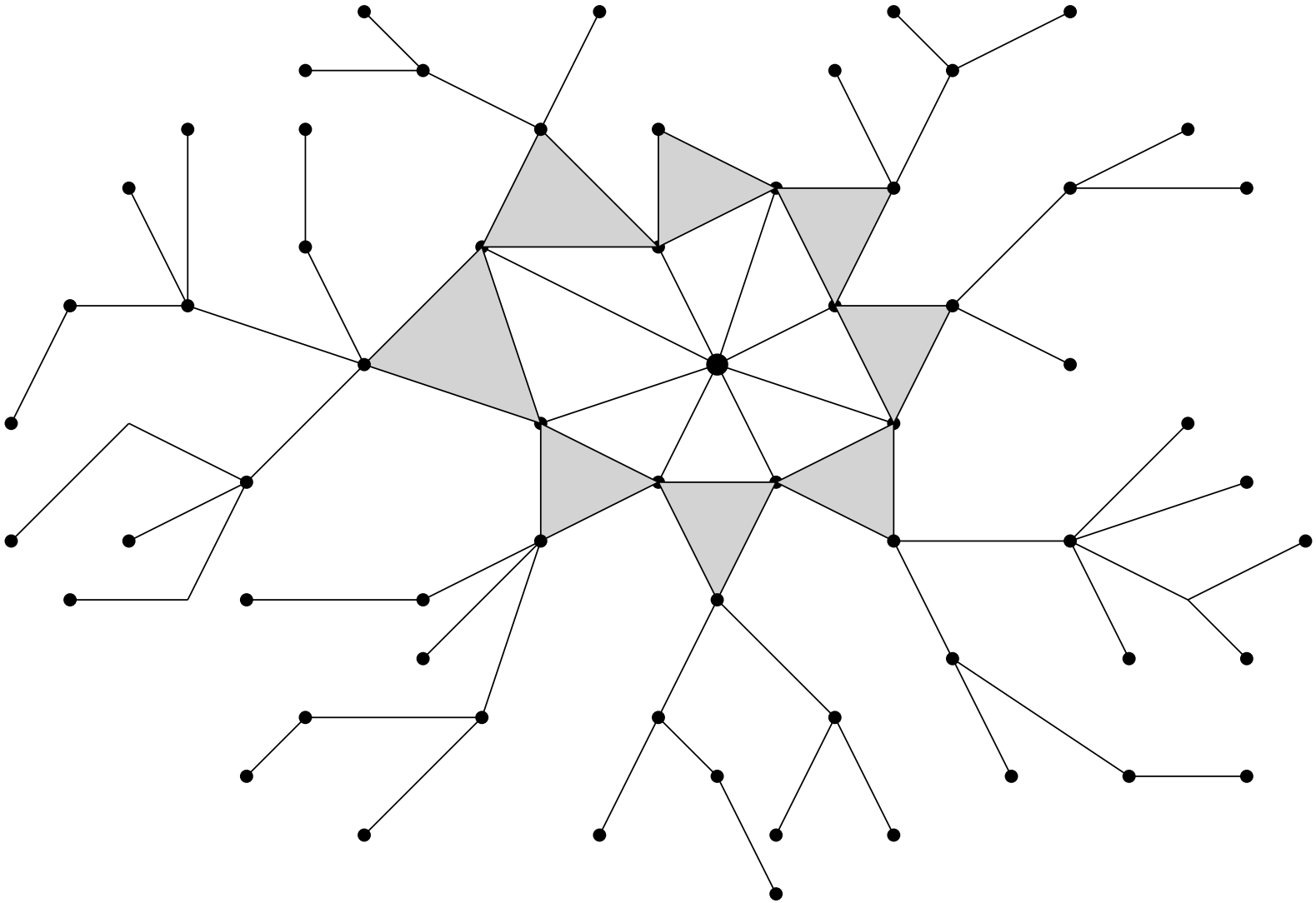




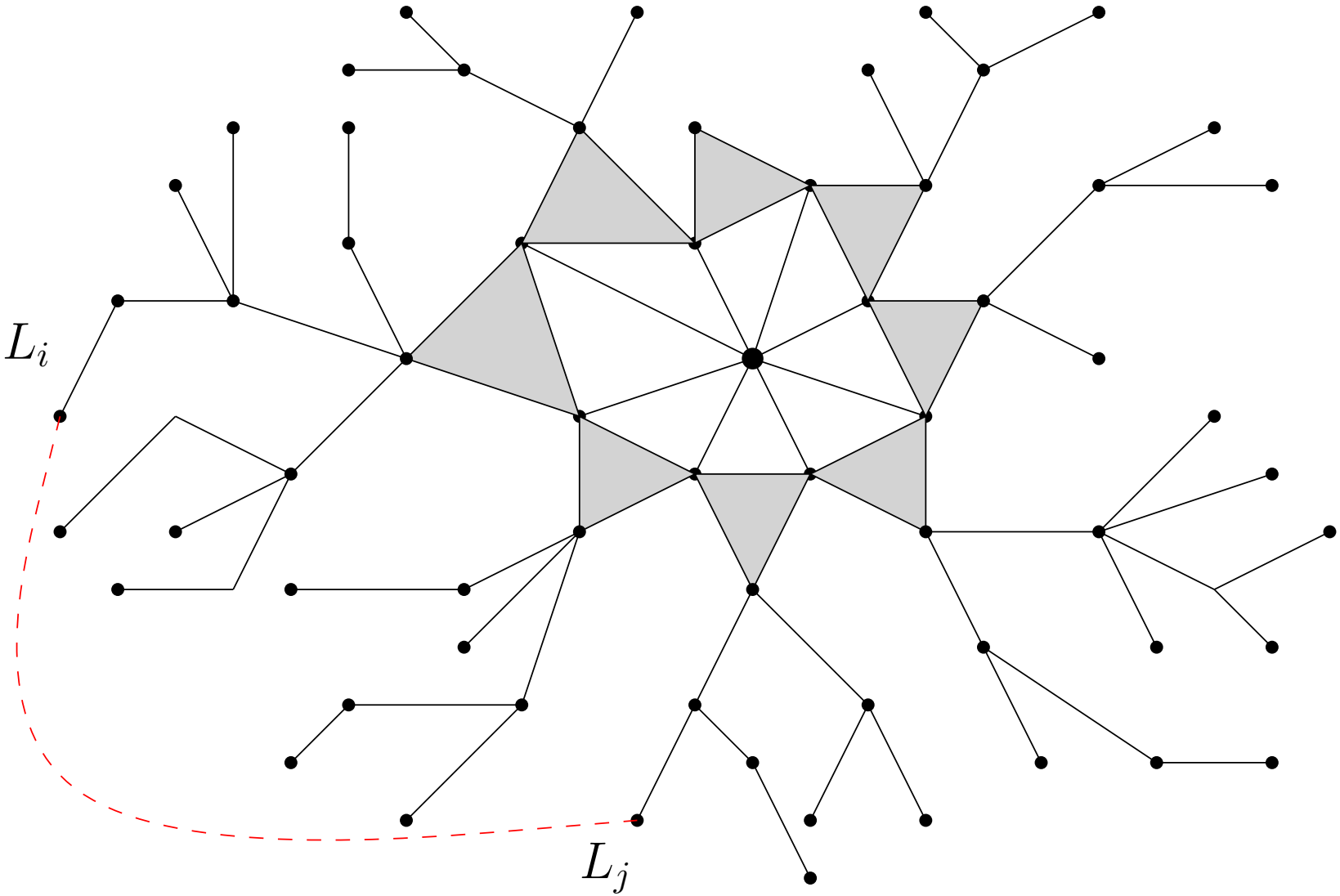


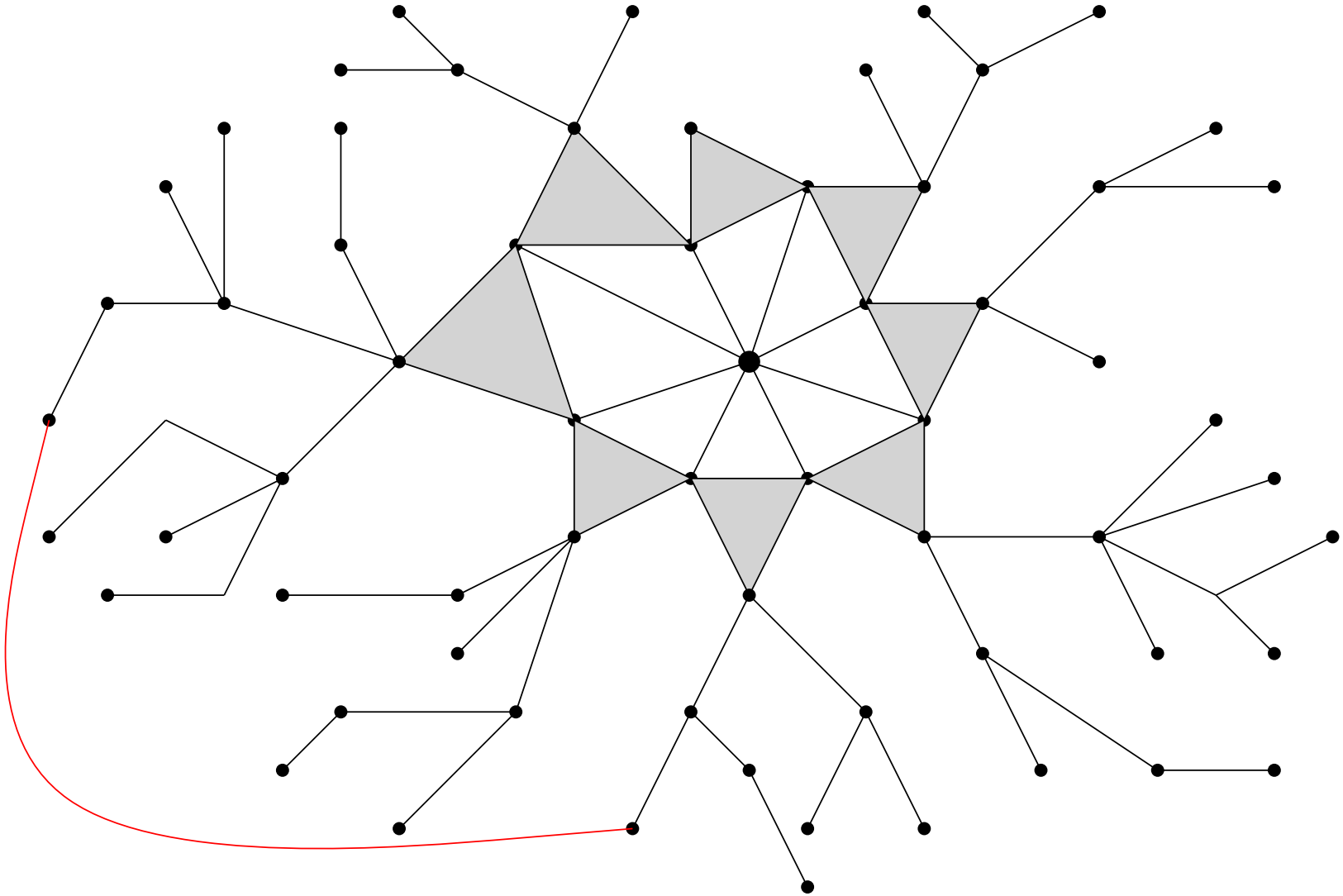


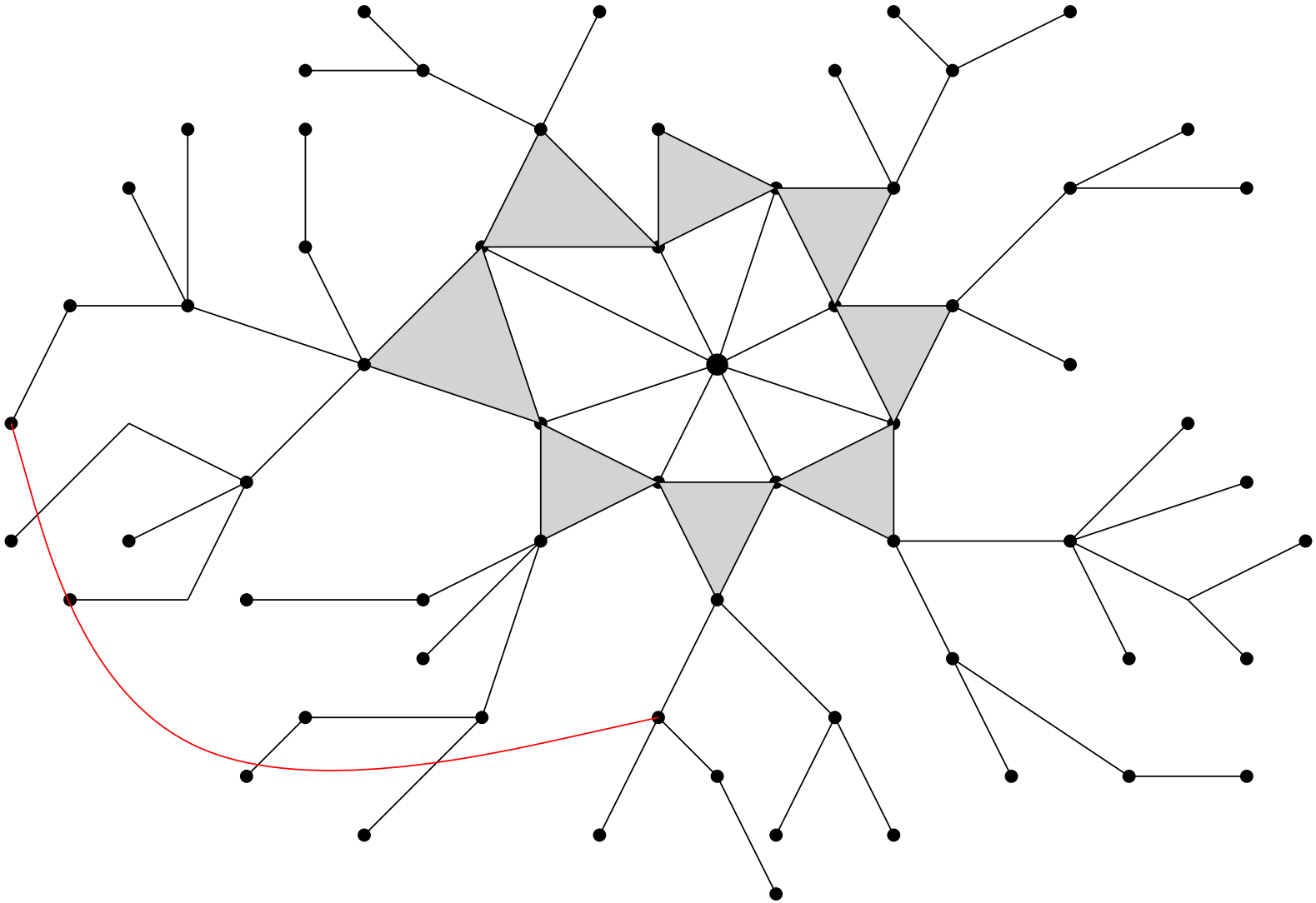


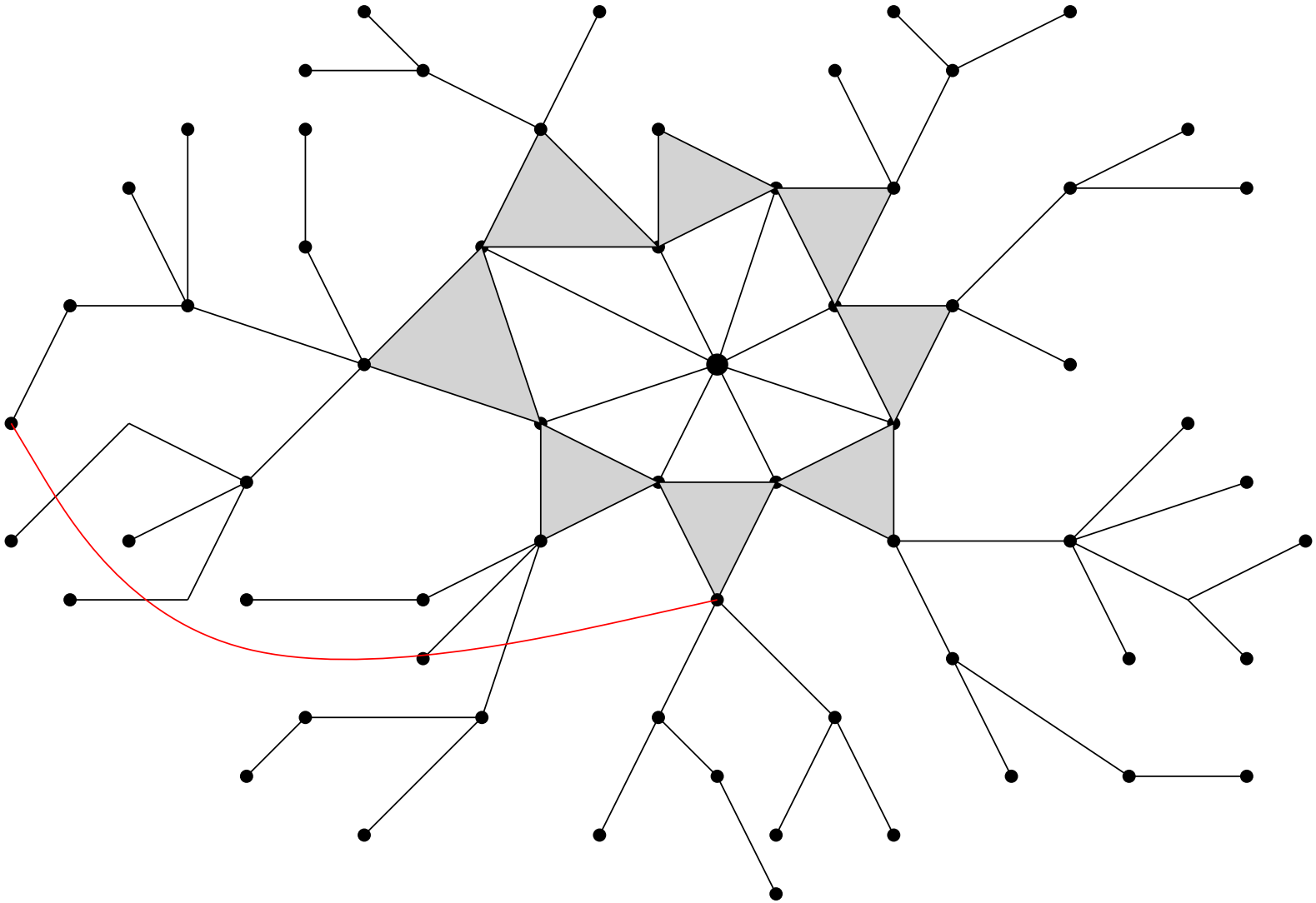


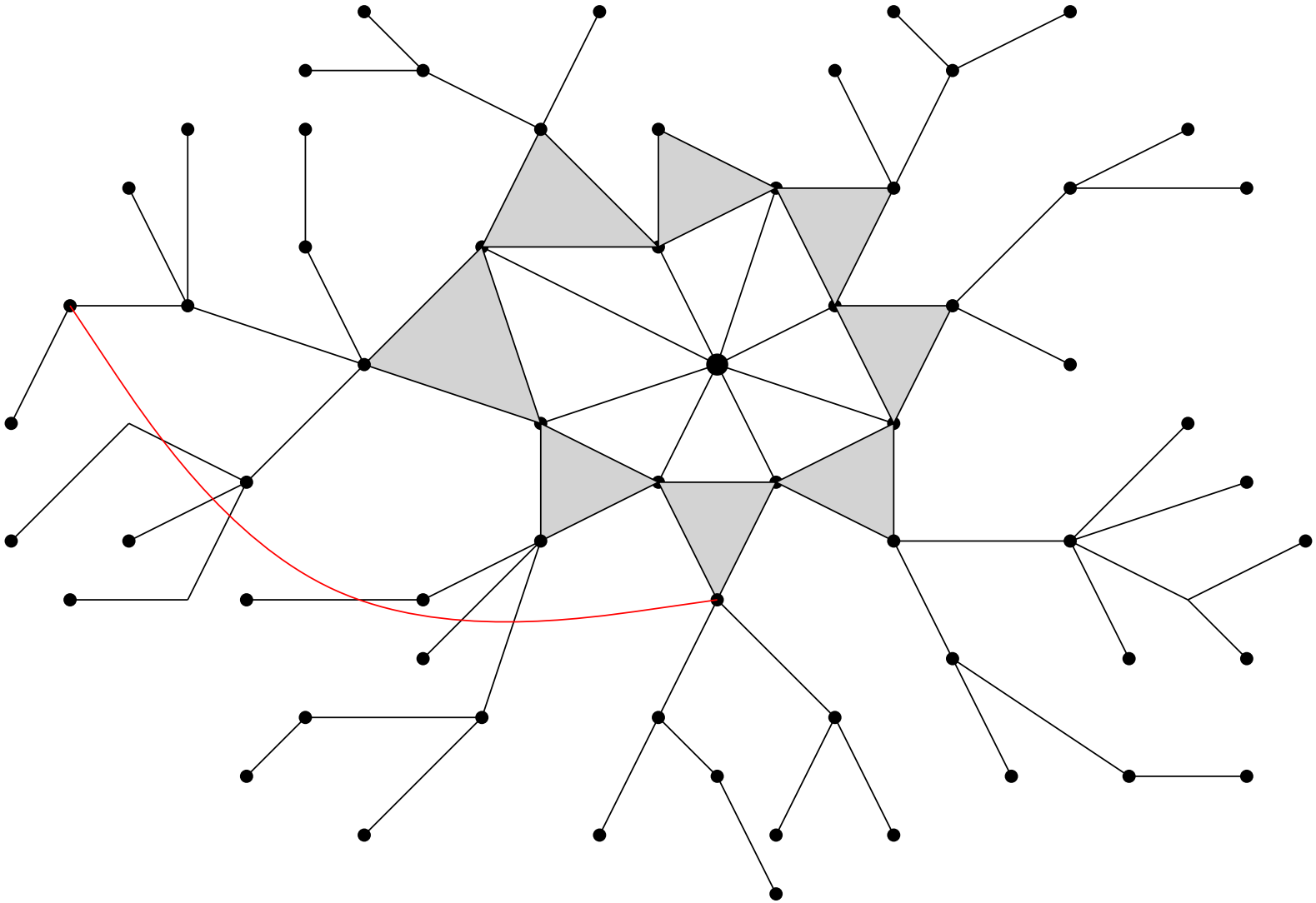
Intersection $L \cap C$:
$F_i,\ F_j,\ L_i,\ L_j$ where $L_i, L_j \in \mathbf{V}$

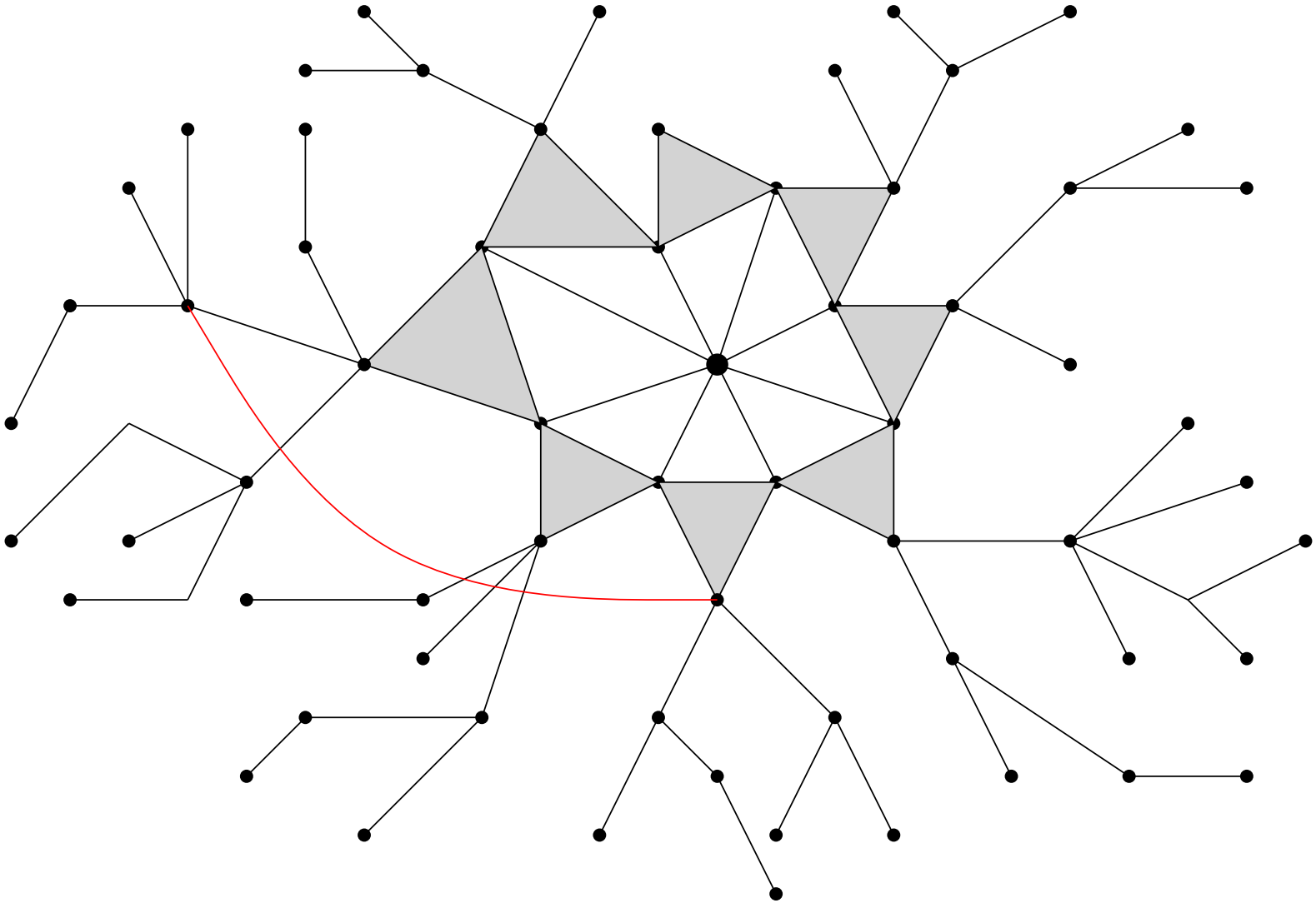


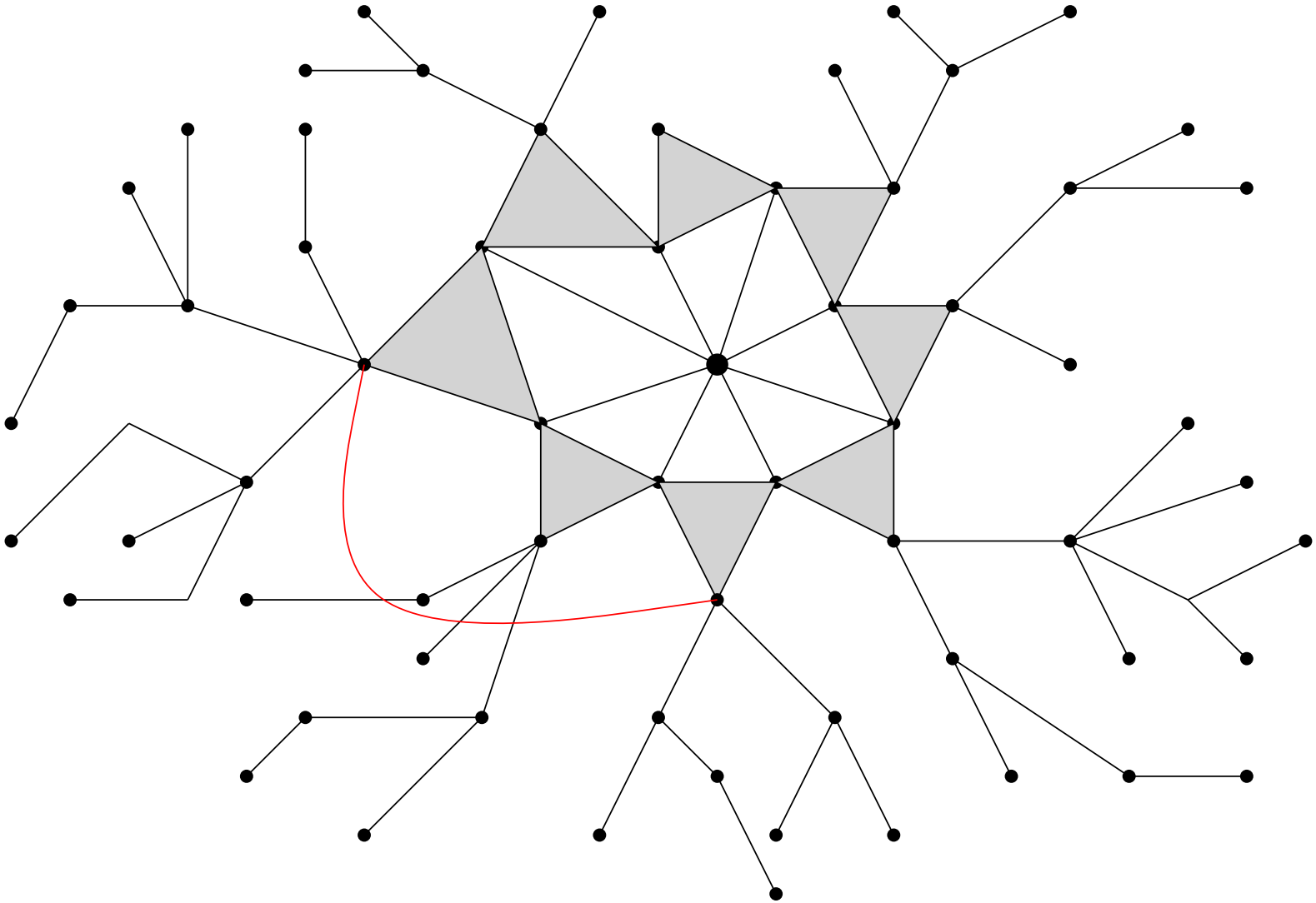


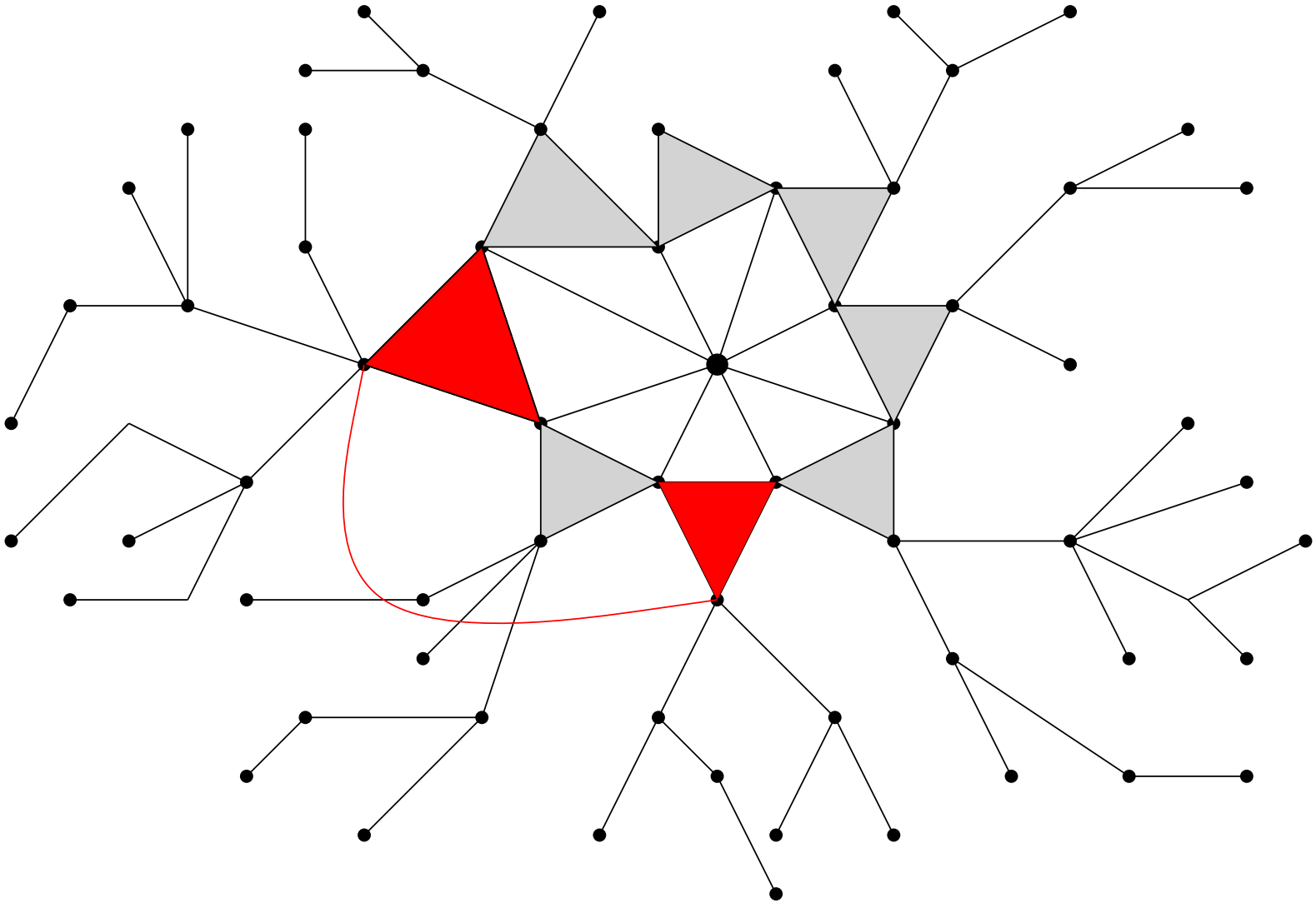


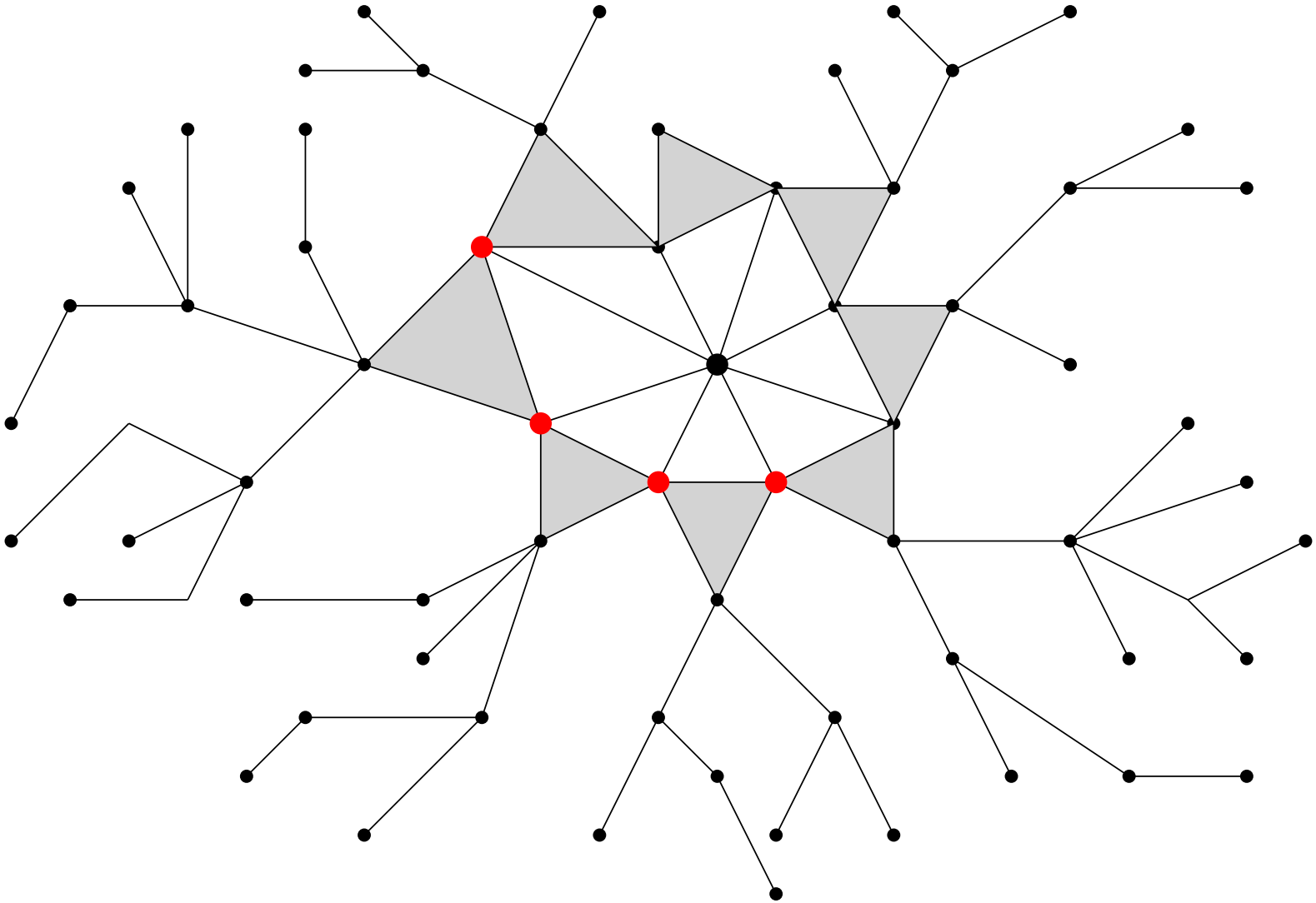


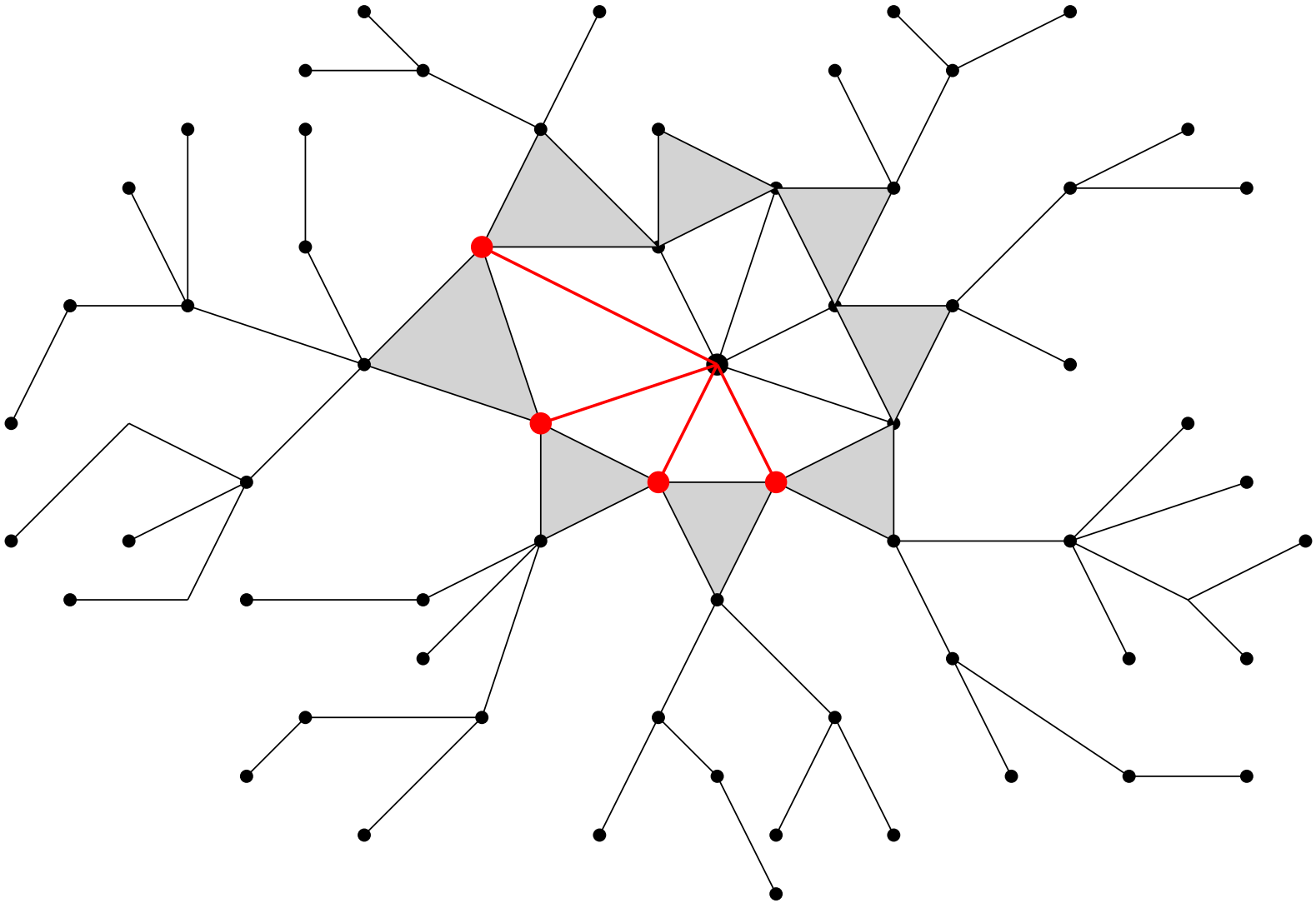




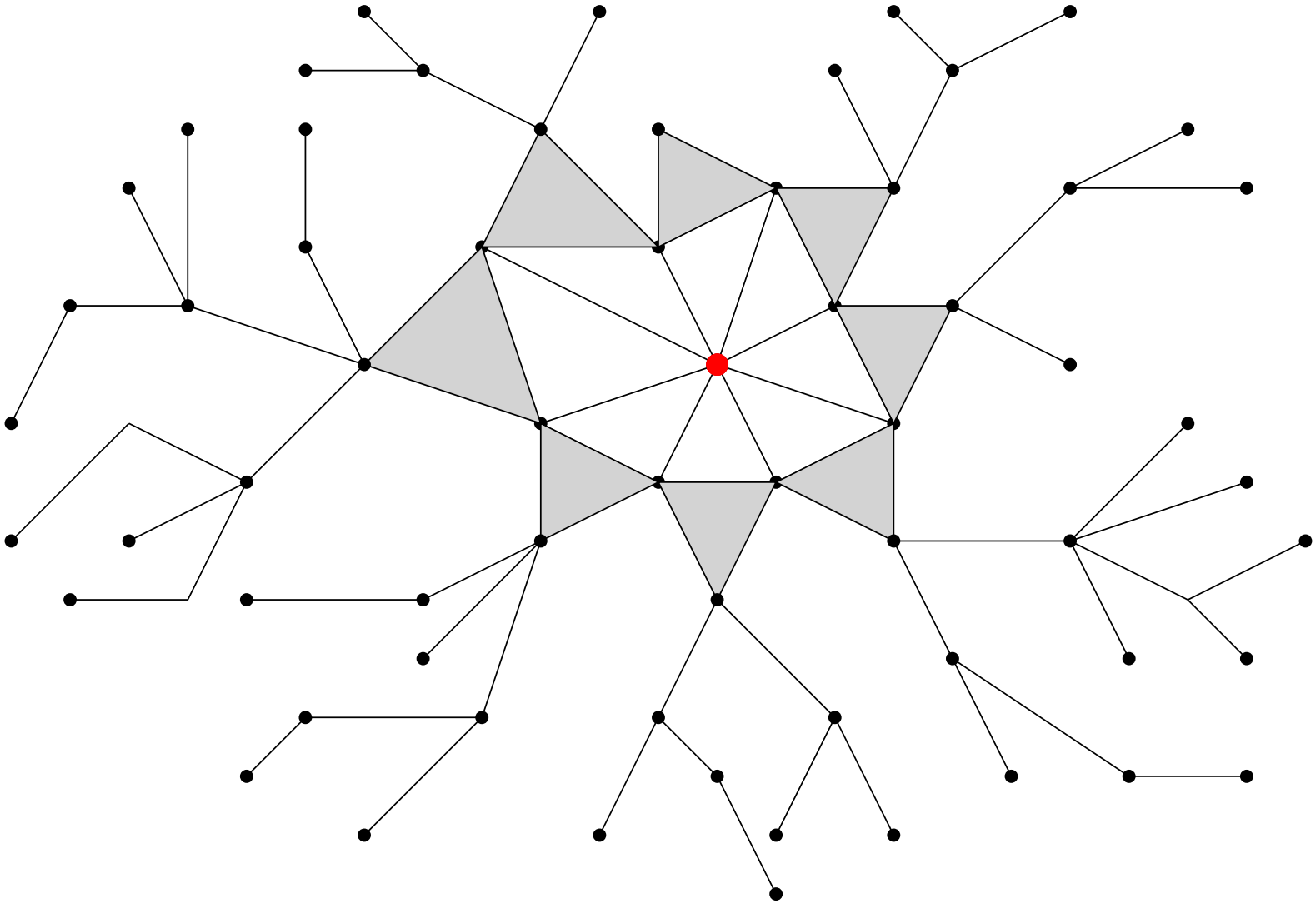








Produced a full relation!



Linear algebra step:

1. Build a matrix as follows:
   - Elements of $\mathcal{F}$ label columns
   - Coefficients of full relations give the rows
   - For the first two rows use the divisors D_1 and D_2
2. Use linear algebra modulo the order of the cyclic subgroup of the Jacobian to find a linear combination of the rows that sums to 0 and involves the first and the second row. If γ_1 and γ_2 are the coefficients of the first and the second row, compute

$$x \equiv -\frac{\gamma_1}{\gamma_2}\,.$$

3. Output x

Two theorems:

Theorem 3 (L.-Lauter)

Under some heuristic assumptions and if q is large enough, we can expect the algorithm to terminate successfully (and return the discrete logarithm x). The number of pairs of factor base elements will be just right. The size of the factor base will be approximately $4\lambda^4\, q^{1/2}$ and the number of vertices in the graph at the end will be approximately $N_{max} := 4\lambda^2\, q^{3/4}$.

Theorem 4 (L.-Lauter)

The average row weight of the matrix (the average number of non-zero entries on a row) will be $\leq 18 - 8\lambda + \ln q$.

Complexity

Implementation and experiments:

- Implemented in C++ (with some Magma)
- Used slightly oversized $\mathcal{F}$ to guarantee success
- Total number of field multiplications[1]

$$M_{\text{Total}} = (7 \log_2 q + 13) \cdot 8\lambda^8 q$$

- Locally, for instance if $q \in [2^{70}, 2^{120}]$:

$$M_{\text{Total}} \approx 1.23 \cdot \log_2^2(q) \cdot q$$

[1] In our implementation

Experimental results:

q	λ	$\log_q \#\mathcal{F}$	$\log_q M_{\text{Total}}$ (th/pr)	$\log_q N_{\max}$ (th)	Mem (th)
2^{17}	0.89	> 0.58	1.51/1.51	0.85	8 MB
2^{19}	0.90	> 0.57	1.47/1.47	0.84	22 MB
2^{21}	0.91	> 0.57	1.44/1.44	0.83	65 MB
2^{23}	0.92	> 0.57	1.41/1.41	0.83	187 MB
2^{25}	0.93	> 0.56	1.39/1.38	0.82	538 MB
2^{27}	0.94	> 0.56	1.37/1.36	0.82	1550 MB

Practical performance corresponds closely to theoretical predictions!

Bigger examples (th):

q	λ	$\log_q \#\mathcal{F}$	$\log_q N_{\max}$	$\log_q M_{\text{Total}}$	$\log_2 M_{\text{Total}}$	Mem
2^{30}	0.95	0.56	0.81	1.34	40.21	7.4 GB
2^{40}	0.98	0.55	0.80	1.27	51.00	1430 GB
2^{50}	1.01	0.54	0.79	1.23	61.62	267 TB
2^{60}	1.03	0.54	0.78	1.20	72.13	50490 TB
2^{70}	1.05	0.53	0.78	1.18	82.56	$1.90 \cdot 10^{7}$ TB
2^{80}	1.07	0.53	0.78	1.16	92.94	$3.56 \cdot 10^{9}$ TB
2^{90}	1.09	0.53	0.77	1.15	103.28	$6.62 \cdot 10^{11}$ TB
2^{100}	1.10	0.53	0.77	1.14	113.58	$1.2 \cdot 10^{14}$ TB
2^{110}	1.11	0.52	0.77	1.13	123.85	$2.28 \cdot 10^{16}$ TB
2^{115}	1.12	0.52	0.77	1.12	128.98	$3.10 \cdot 10^{17}$ TB
2^{120}	1.13	0.52	0.77	1.12	134.10	$4.22 \cdot 10^{18}$ TB
2^{140}	1.15	0.52	0.77	1.10	154.54	$2.16 \cdot 10^{23}$ TB
2^{160}	1.17	0.52	0.77	1.09	174.92	$7.29 \cdot 10^{27}$ TB
2^{180}	1.18	0.52	0.76	1.08	195.26	$8.43 \cdot 10^{31}$ TB
2^{200}	1.20	0.52	0.76	1.08	215.56	$1.10 \cdot 10^{37}$ TB
2^{220}	1.21	0.51	0.76	1.07	235.84	$3.71 \cdot 10^{41}$ TB
2^{240}	1.23	0.51	0.76	1.07	256.09	$1.24 \cdot 10^{46}$ TB

Linear algebra:

- Expected average row weight: $w_{\text{est}} := 18 - 8 \ln \lambda + \ln q$
- Complexity of linear algebra: $O\left(w_{\text{est}} \cdot (\#\mathcal{F})^2\right)$
- Linear algebra complexity is comparable to relation search complexity

Linear algebra (th):

q	$\log_2 w_{\text{est}}$	$\log_2 \#\mathcal{F}$	Lin. alg.	M_{Total}
2^{30}	5.29	16.70	$\approx 2^{39}$	$\approx 2^{40}$
2^{40}	5.52	21.90	$\approx 2^{49}$	$\approx 2^{51}$
2^{50}	5.72	27.06	$\approx 2^{60}$	$\approx 2^{62}$
2^{60}	5.89	32.18	$\approx 2^{70}$	$\approx 2^{72}$
2^{70}	6.05	37.29	$\approx 2^{81}$	$\approx 2^{83}$
2^{80}	6.19	42.39	$\approx 2^{91}$	$\approx 2^{93}$
2^{90}	6.32	47.47	$\approx 2^{101}$	$\approx 2^{103}$
2^{100}	6.44	52.55	$\approx 2^{112}$	$\approx 2^{114}$
2^{110}	6.55	57.62	$\approx 2^{122}$	$\approx 2^{124}$
2^{115}	6.60	60.15	$\approx 2^{127}$	$\approx 2^{129}$
2^{120}	6.65	62.68	$\approx 2^{132}$	$\approx 2^{134}$
2^{140}	6.83	72.79	$\approx 2^{152}$	$\approx 2^{155}$
2^{160}	7.00	82.89	$\approx 2^{173}$	$\approx 2^{175}$
2^{180}	7.14	92.97	$\approx 2^{193}$	$\approx 2^{195}$
2^{200}	7.28	103.05	$\approx 2^{213}$	$\approx 2^{216}$
2^{220}	7.40	113.12	$\approx 2^{234}$	$\approx 2^{236}$
2^{240}	7.51	123.18	$\approx 2^{254}$	$\approx 2^{256}$

Time-Memory Trade-offs

Dealing with the memory cost:

- Computational complexity is impressive
- Memory cost makes the algorithm **unusable** for large q
- We can stop graph building at any point to restrict memory cost
- But bigger graph size gives better computational complexity

Stop adding new vertices when graph has size $\chi\, q^{3/4}$?

(**Remark:** $\chi \leq 4\lambda^2$)

Bounding the size of the graph:

- Let η be the positive real root of

$$2\eta^2 \exp\left(4\eta^8 - 4\eta^4/\chi^2 + 1/4\right) = \chi^{1/2}\, q^{1/8}$$

- $\mathcal{RP}$ now a set of $4\eta\, q^{1/8}$ points in $C(\mathbb{F}_q)$
- Proceed as usual, but restrict $N_{\max}$ to $\chi\, q^{3/4}$
- The size of $\mathcal{F}$ will be $4\eta^2\, q^{1/2}$

Theorem 5 (L.-Lauter)

If q is large enough, we can expect the algorithm to terminate successfully.

The average row weight of the matrix will be approximately

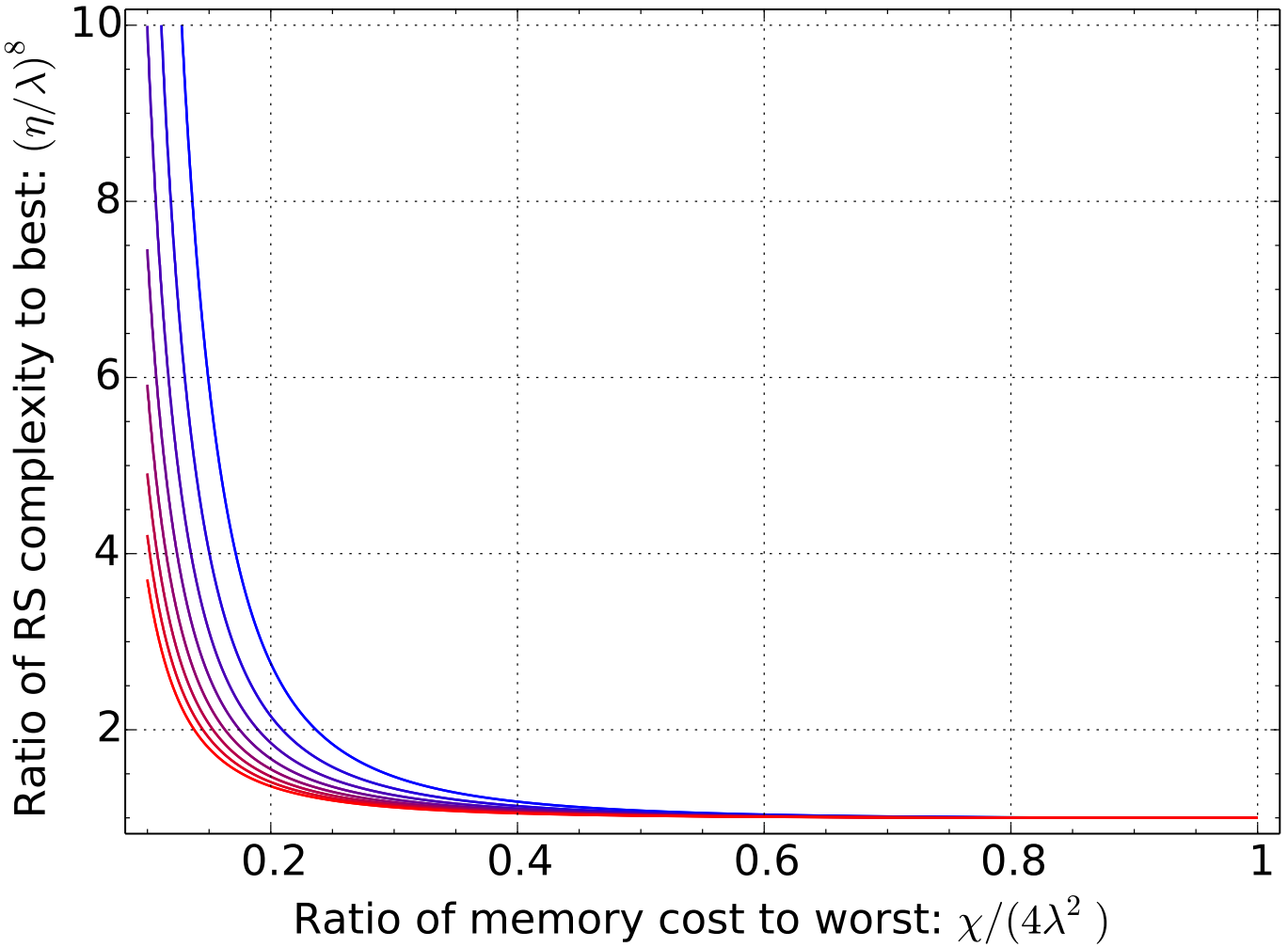
$$w_{est}^{\chi} := 20 - \frac{\chi^2}{8\eta^4} - 4\ln\left(4\eta^4\right) + \ln q + 4\ln\chi\,.$$

Remark: As a function of χ, η has a global minimum at a point where $\chi = 4\eta^2$, which recovers the original case (the definition of λ). This is expected since an unbounded graph should yield the best computational complexity.

Why do this?

Relation search complexity compared to unrestricted graph case:

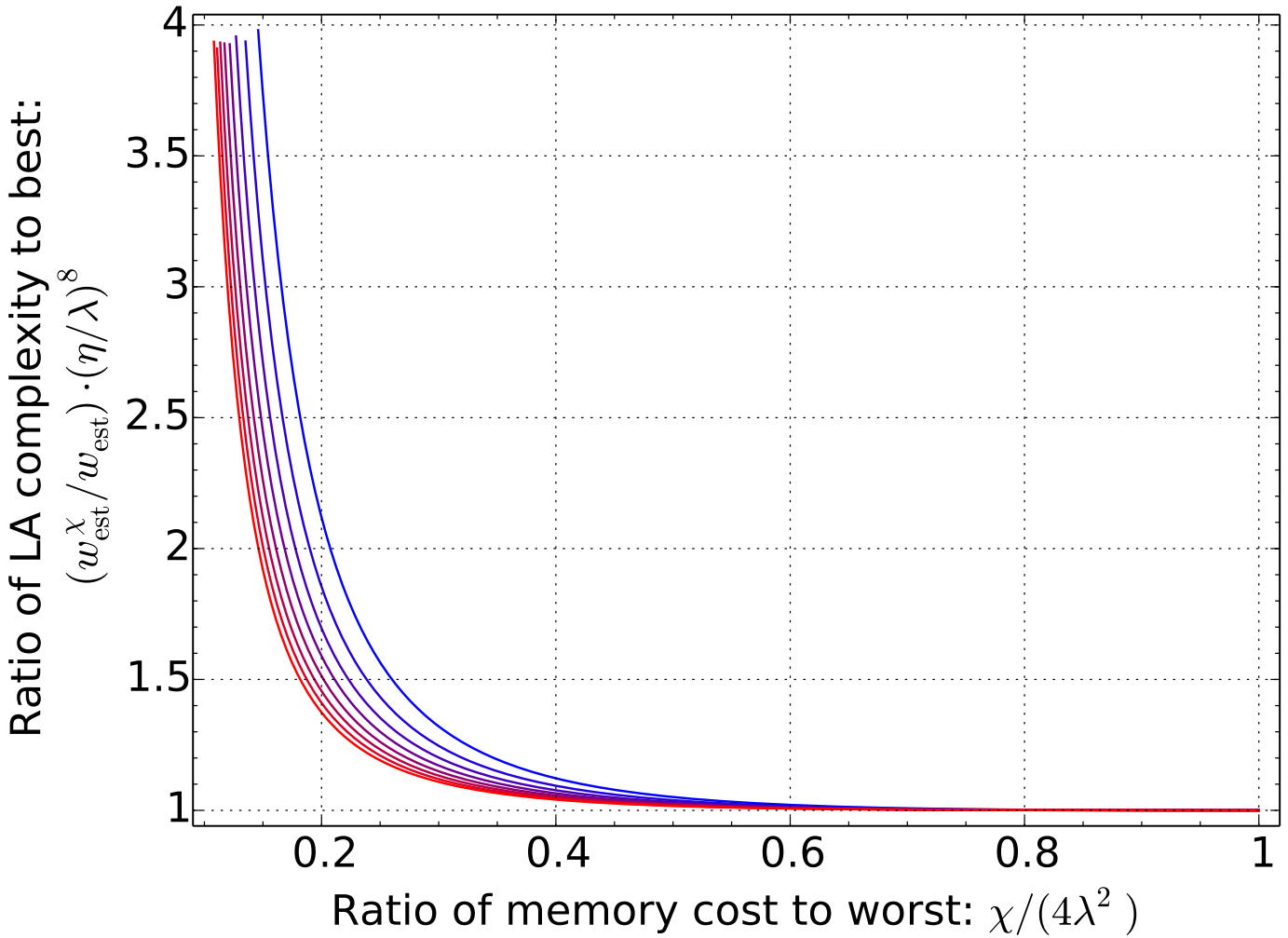
$$q = 2^{60}, 2^{80}, 2^{100}, 2^{120}, 2^{140}, 2^{160}, 2^{180}, 2^{200}$$



Why do this?

Linear algebra complexity compared to unrestricted graph case:

$$q = 2^{60}, 2^{80}, 2^{100}, 2^{120}, 2^{140}, 2^{160}, 2^{180}, 2^{200}$$



Parallelize memory cost by dividing graph among K computers:

- K computers perform the main task independently
- Memory cost per computer $\approx (1/\sqrt{K}) \cdot$ (case of $K = 1$)
- Total memory cost $\approx \sqrt{K} \cdot$ (case of $K = 1$)
- Running time increases slightly
- Can combine graph bounding with parallelization

Conclusions

Example 6

Let $q = 2^{60}$, $K = 100$ and $\chi/(4\lambda_K^2) = 1/50$. Then:

- Complexity per computer $\approx 2^{79}$
- The memory cost per computer ≈ 100 TB
- **No bounding/parallelization:**
  - Complexity $\approx 2^{72}$
  - Memory cost ≈ 50000 TB

Final words:

- Index calculus works reasonably well up to $q \approx 2^{60}$
- For $q \geq 2^{70}$ memory cost extremely high without bounding
- **Generalizations to higher genus:** Thériault and Oyono (work in progress)
- Combine with recent improved sieving method of Vitse-Wallet (LatinCrypt 2015)

Thank you for listening!

Questions?