On the Structure and Density of Sets with Trivial Square Products

Erdős Problem 888

January 20, 2026

Abstract

We analyze the extremal function $f(n)$ (OEIS A387584), defined as the maximum size of a subset $A \subseteq \{1, \ldots, n\}$ such that for any elements $a \leq b \leq c \leq d \in A$, if the product $abcd$ is a perfect square, then $ad = bc$. We establish the **Kernel Constraint Theorem**, proving that any valid set contains at most one element from each square-free equivalence class, yielding the bound $f(n) \leq Q(n)$. We define $f_{\text{sf}}(n)$ as the maximum size of a *square-free* subset satisfying the condition and develop an $O(n^2)$ algorithm to compute it exactly for $n \leq 5000$. While $f_{\text{sf}}(n)$ provides a strong lower bound, we discover a "fixing mechanism" whereby non-square-free numbers can resolve topological obstructions, proving that the kernel map does not preserve admissibility. We demonstrate this with the explicit counterexample $A = \{3, 5, 126, 210\}$. Empirical data shows the density $f_{\text{sf}}(n)/n$ decays monotonically, supporting the conjecture that $f(n) = o(n)$.

1 Introduction

1.1 Problem Statement

Let $S_n = \{1, \ldots, n\}$ denote the set of positive integers up to n. We consider the problem of finding the maximum cardinality of a subset $A \subseteq S_n$ satisfying the following condition:

Definition 1.1 (Erdős 888 Condition)**.** A set $A \subseteq S_n$ is *admissible* if it satisfies:

$$\forall\, a, b, c, d \in A \text{ with } a \leq b \leq c \leq d: \quad abcd = k^2 \text{ for some } k \in \mathbb{N} \implies ad = bc. \tag{†}$$

Definition 1.2 (Extremal Function)**.** We define the *extremal function* $f : \mathbb{N} \to \mathbb{N}$ by

$$f(n) = \max\{|A| : A \subseteq S_n \text{ is admissible}\}.$$

The elements a, b, c, d in condition (†) need not be distinct. The consequent $ad = bc$ permits several "trivial" configurations:

- Geometric progressions: $\{x, y, cx, cy\}$ with $x \leq y \leq cx \leq cy$
- Repetitions: quadruples of the form (x, x, y, y)

- Constant sequences: (x, x, x, x)

We distinguish two related quantities:

1. $f(n)$: the maximum over all admissible subsets of S_n
2. $f_{\text{sf}}(n)$: the maximum over admissible subsets of S_n^{sf} (square-free integers only)

1.2 Mathematical Context

This problem, posed by Erdős as Problem 888 in his collected problems [1], relaxes the classical **Multiplicative Sidon Set** condition. A multiplicative Sidon set requires $ab = cd \implies \{a, b\} = \{c, d\}$, yielding sets of size approximately $\pi(n) \sim n/\log n$.

The Erdős 888 condition is strictly weaker: it allows $ab = cd$ provided that when the four elements are ordered as $a \leq b \leq c \leq d$, we have $ad = bc$. This "triviality loophole" permits larger sets, and the central question is whether sets of positive density are achievable.

The problem connects to several areas:

- **Product-free sets:** Sets avoiding $ab = c$ have density zero [2].
- **Square-free numbers:** The density of square-free integers is $6/\pi^2 \approx 0.6079$.
- **Hypergraph Turán theory:** Our approach models the problem as an independence number computation.

1.3 Summary of Results

Our main contributions are:

1. **Kernel Constraint (Theorem 2.2):** Any admissible set contains at most one element from each square-free equivalence class, giving $f(n) \leq Q(n)$.
2. **Fixing Mechanism (Theorem 2.9):** Non-square-free representatives can resolve obstructions, proving that the kernel map does not preserve admissibility.
3. **Hypergraph Algorithm (Section 3):** An $O(n^2)$ algorithm computes $f_{\text{sf}}(n)$ exactly.
4. **Asymptotic Bounds (Theorem 4.1):** We establish $\pi(n) \leq f_{\text{sf}}(n) \leq f(n) \leq Q(n)$.
5. **Computational Results (Section 6):** Exact values of $f_{\text{sf}}(n)$ for $n \leq 5000$ supporting $f(n) = o(n)$.

2 Theoretical Constraints

2.1 The Kernel Constraint

Definition 2.1 (Square-Free Kernel)**.** For a positive integer m with prime factorization $m = \prod_i p_i^{e_i}$, the *square-free kernel* is

$$\kappa(m) = \prod_i p_i.$$

Equivalently, $\kappa(m)$ is the largest square-free divisor of m, and $m = \kappa(m) \cdot s^2$ for some positive integer s.

Theorem 2.2 (Kernel Constraint)**.** *If A satisfies condition (†), then for any distinct $x, y \in A$, the product xy is not a perfect square.*

Proof. Suppose, for contradiction, that distinct elements $x, y \in A$ exist with $xy = k^2$ for some positive integer k. Without loss of generality, assume $x < y$.

Consider the quadruple $(a, b, c, d) = (x, x, x, y)$. This satisfies the ordering $a \leq b \leq c \leq d$. The product is

$$abcd = x \cdot x \cdot x \cdot y = x^3 y = x^2 \cdot (xy) = x^2 k^2 = (xk)^2,$$

which is a perfect square.

By condition (†), we require $ad = bc$, which gives

$$x \cdot y = x \cdot x \implies y = x.$$

This contradicts the assumption that x and y are distinct. □

Corollary 2.3. *The kernel map $\kappa : A \to S_n^{\text{sf}}$ is injective on any admissible set A.*

Proof. Suppose $\kappa(x) = \kappa(y)$ for $x, y \in A$. Then $x = \kappa(x) \cdot s^2$ and $y = \kappa(y) \cdot t^2$ for some $s, t \geq 1$. Thus

$$xy = \kappa(x)^2 \cdot s^2 t^2 = (\kappa(x) \cdot st)^2,$$

a perfect square. By Theorem 2.2, we must have $x = y$. □

Corollary 2.4. *$f(n) \leq Q(n)$, where $Q(n) = |S_n^{\text{sf}}|$ is the count of square-free numbers up to n.*

Proof. By Corollary 2.3, $|A| = |\kappa(A)| \leq |S_n^{\text{sf}}| = Q(n)$. □

Remark 2.5. Asymptotically, $Q(n) \sim \frac{6}{\pi^2} n \approx 0.6079n$. Thus, Corollary 2.4 implies $f(n) \leq 0.6079n + o(n)$.

2.2 The Fixing Mechanism

A natural hypothesis is that optimal admissible sets consist entirely of square-free numbers, i.e., $f(n) = f_{\text{sf}}(n)$. We show this is false in general by demonstrating that non-square-free representatives can "fix" obstructions.

Definition 2.6 (Realization)**.** Let $K \subseteq S_n^{\text{sf}}$ be a set of square-free numbers. A *realization* of K in S_n is a set

$$A = \{\kappa \cdot s_\kappa^2 : \kappa \in K\}$$

where each $s_\kappa \geq 1$ and $\kappa \cdot s_\kappa^2 \leq n$.

Definition 2.7 (Bad Quadruple)**.** A *bad quadruple* in a set $K \subseteq S_n^{\text{sf}}$ is an ordered tuple $(\kappa_a, \kappa_b, \kappa_c, \kappa_d)$ with $\kappa_a < \kappa_b < \kappa_c < \kappa_d$ such that:

1. $\kappa_a \kappa_b \kappa_c \kappa_d$ is a perfect square, and

2. $\kappa_a \kappa_d \neq \kappa_b \kappa_c$.

Theorem 2.8 (Admissibility of Realizations)**.** *Let $K \subseteq S_n^{\mathrm{sf}}$. A realization $A = \{\kappa \cdot s_\kappa^2 : \kappa \in K\}$ is admissible if and only if for every bad quadruple in K, the corresponding realized values $a, b, c, d \in A$, when sorted as $a \leq b \leq c \leq d$, satisfy $ad = bc$.*

Proof. This follows directly from condition (†) and Theorem 2.2. The key observation is that scaling by squares preserves the "squareness" of products: if $\kappa_a \kappa_b \kappa_c \kappa_d$ is a square, then so is $(s_a s_b s_c s_d)^2 \cdot \kappa_a \kappa_b \kappa_c \kappa_d = abcd$. However, the ordering and the products ad and bc may change with scaling. □

Theorem 2.9 (Existence of Non-Trivial Fixing)**.** *There exist admissible sets A such that their kernel set $\kappa(A)$ is not admissible.*

Proof. Let $n = 210$ and consider the set

$$A = \{3, 5, 126, 210\}.$$

Step 1: Verify that A is admissible.

The elements are ordered as $3 \leq 5 \leq 126 \leq 210$. We compute:

$$3 \cdot 5 \cdot 126 \cdot 210 = 396{,}900 = 630^2.$$

The product is a perfect square, so we must check the triviality condition:

$$ad = 3 \cdot 210 = 630, \qquad bc = 5 \cdot 126 = 630.$$

Since $ad = bc$, the set A is admissible.

Step 2: Compute the kernel set.

We have:

$$\kappa(3) = 3, \qquad \kappa(5) = 5,$$
$$\kappa(126) = \kappa(2 \cdot 3^2 \cdot 7) = 2 \cdot 7 = 14, \qquad \kappa(210) = \kappa(2 \cdot 3 \cdot 5 \cdot 7) = 210.$$

Thus $K = \kappa(A) = \{3, 5, 14, 210\}$.

Step 3: Verify that K is not admissible.

The elements of K are ordered as $3 \leq 5 \leq 14 \leq 210$. We compute:

$$3 \cdot 5 \cdot 14 \cdot 210 = 44{,}100 = 210^2.$$

The product is a perfect square. Checking the triviality condition:

$$ad = 3 \cdot 210 = 630, \qquad bc = 5 \cdot 14 = 70.$$

Since $ad \neq bc$, the kernel set K is *not* admissible.

Conclusion: The scaling factor $s_{14} = 3$ (giving $14 \cdot 9 = 126$) "fixes" the obstruction present in the kernel set. □

Corollary 2.10. *The map $A \mapsto \kappa(A)$ does not preserve admissibility.*

Corollary 2.11. *$f(n) \geq f_{\mathrm{sf}}(n)$ for all n, with potential strict inequality.*

Remark 2.12. Theorem 2.9 shows that fixing is *possible*. However, it does not immediately imply $f(n) > f_{\mathrm{sf}}(n)$, since achieving this would require a fixable kernel set K with $|K| > \alpha(H_n)$, where $\alpha(H_n)$ is the independence number of the hypergraph defined in Section 3.

3 The Hypergraph Model for $f_{\mathrm{sf}}(n)$

While computing $f(n)$ requires optimizing over scale factors, the restricted function $f_{\mathrm{sf}}(n)$ is a pure combinatorial invariant that we can model using hypergraph theory.

3.1 Construction

Definition 3.1 (Bad Quadruple Hypergraph)**.** Define the 4-uniform hypergraph $H_n = (V_n, \mathcal{E}_n)$ where:

- $V_n = S_n^{\mathrm{sf}}$ is the set of square-free numbers up to n.
- $\mathcal{E}_n$ consists of all sets $\{a, b, c, d\} \subset V_n$ with $a < b < c < d$ such that $abcd$ is a perfect square and $ad \neq bc$.

Proposition 3.2. *$f_{\mathrm{sf}}(n) = \alpha(H_n)$, the independence number of H_n.*

Proof. A subset $A \subseteq V_n$ is admissible (as a square-free set) if and only if it contains no bad quadruple, i.e., if and only if A is an independent set in H_n. □

Remark 3.3. By the relation between independence number and vertex cover number:

$$f_{\mathrm{sf}}(n) = \alpha(H_n) = |V_n| - \tau(H_n),$$

where $\tau(H_n)$ is the minimum vertex cover of H_n.

3.2 Characterization of Hyperedges

Lemma 3.4. *For distinct square-free integers a, b, c, d, the product $abcd$ is a perfect square if and only if each prime factor appears in an even number of the four terms.*

Proof. Since each of a, b, c, d is square-free, each prime appears with exponent 0 or 1 in each term. The product $abcd$ is a square if and only if each prime has even total exponent, which occurs if and only if each prime appears in an even number (0, 2, or 4) of the terms. □

Definition 3.5 (Core of a Pair)**.** For square-free integers u, v, define

$$\mathrm{core}(u, v) = \frac{uv}{\gcd(u, v)^2}.$$

Lemma 3.6. *For two disjoint pairs $\{a, b\}$ and $\{c, d\}$ of square-free numbers, the product $abcd$ is a perfect square if and only if $\mathrm{core}(a, b) = \mathrm{core}(c, d)$.*

Proof. Note that $\text{core}(u, v)$ is the square-free part of uv. Thus $abcd$ is a square if and only if $(ab)(cd)$ is a square, if and only if $\text{core}(a, b) = \text{core}(c, d)$ (since these are the square-free parts). □

3.3 The $O(n^2)$ Algorithm

A naive enumeration of all quadruples requires $O(|V_n|^4)$ time. Using Lemma 3.6, we reduce this to $O(|V_n|^2) = O(n^2)$.

[Pairwise Aggregation for $f_{\text{sf}}(n)$]

1. Generate $V_n = S_n^{\text{sf}}$ via sieve.
2. For all pairs $\{u, v\} \subseteq V_n$, compute $k = \text{core}(u, v)$ and group pairs by their core value.
3. For each core group with ≥ 2 pairs, check all combinations of disjoint pairs $\{a, b\}, \{c, d\}$. If $ad \neq bc$ (after sorting), add $\{a, b, c, d\}$ to $\mathcal{E}_n$.
4. Compute $\tau(H_n)$ via branch-and-bound on the minimum vertex cover.
5. Return $f_{\text{sf}}(n) = |V_n| - \tau(H_n)$.

Complexity Analysis:

- Step 1: $O(n)$ via sieve of Eratosthenes.
- Step 2: $O(|V_n|^2) = O(n^2)$ pair computations.
- Step 3: For typical inputs, the hypergraph is sparse.
- Step 4: NP-hard in general, but tractable for the sparse instances arising here.

4 Asymptotic Bounds

Theorem 4.1 (Main Bounds)**.** *For all $n \geq 1$:*

$$\pi(n) \leq f_{\text{sf}}(n) \leq f(n) \leq Q(n).$$

Proof. **Upper bound:** $f(n) \leq Q(n)$ follows from Corollary 2.4.

Middle inequality: $f_{\text{sf}}(n) \leq f(n)$ is immediate since square-free admissible sets are a subset of all admissible sets.

Lower bound: Let $P_n = \{p \leq n : p \text{ prime}\}$. We claim P_n is an independent set in H_n.

For distinct primes $p_1 < p_2 < p_3 < p_4$, the product $p_1p_2p_3p_4$ is square-free (as a product of distinct primes) and thus not a perfect square. Hence no quadruple of primes forms a bad quadruple, so P_n is independent and $f_{\text{sf}}(n) \geq |P_n| = \pi(n)$. □

Corollary 4.2. *Asymptotically:*

$$\frac{n}{\log n} \lesssim f(n) \lesssim \frac{6}{\pi^2} n.$$

Theorem 4.3 (Sárközy, attributed)**.** $f(n) = o(n)$.

Remark 4.4. Theorem 4.3 is stated as solved on the Erdős Problems website [1]. Our computational evidence (Section 6) strongly supports this result.

5 The Gap Question

The relationship between $f(n)$ and $f_{\mathrm{sf}}(n)$ remains unresolved.

Open Problem 5.1. Is $f(n) = f_{\mathrm{sf}}(n)$ for all n?

Proposition 5.2. *If $f(n) > f_{\mathrm{sf}}(n)$ for some n, then there exists a kernel set $K \subseteq S_n^{\mathrm{sf}}$ with $|K| > \alpha(H_n)$ that admits a valid realization in S_n.*

Proof. Suppose A is an optimal admissible set with $|A| = f(n) > f_{\mathrm{sf}}(n) = \alpha(H_n)$. Let $K = \kappa(A)$. By Corollary 2.3, $|K| = |A| > \alpha(H_n)$. By definition, A is a realization of K. □

Remark 5.3 (Constraints on Fixing). For a bad quadruple $(\kappa_a, \kappa_b, \kappa_c, \kappa_d)$ with scales (s_a, s_b, s_c, s_d), the realized values must satisfy $ad = bc$ when reordered. This imposes:

$$\kappa_{\sigma(1)} s_{\sigma(1)}^2 \cdot \kappa_{\sigma(4)} s_{\sigma(4)}^2 = \kappa_{\sigma(2)} s_{\sigma(2)}^2 \cdot \kappa_{\sigma(3)} s_{\sigma(3)}^2,$$

where σ orders by realized value. Multiple overlapping bad quadruples create coupled constraints that may be inconsistent.

We propose the following conjectures based on our analysis:

Conjecture 5.4 (Asymptotic Equivalence)**.** *$f(n) \sim f_{\mathrm{sf}}(n)$ as $n \to \infty$.*

Conjecture 5.5 (Bounded Gap)**.** *$f(n) - f_{\mathrm{sf}}(n) = O(\sqrt{n})$.*

Heuristic for Conjecture 5.5: The fixing mechanism requires $\kappa \cdot s^2 \leq n$, so $s \leq \sqrt{n/\kappa}$. For typical $\kappa \sim n$, only $O(1)$ scale values are available, limiting the number of fixable configurations.

6 Computational Results

We implemented Algorithm 3.3 and computed $f_{\mathrm{sf}}(n)$ exactly for $n \leq 5000$. Additionally, we searched for fixing opportunities at each n.

6.1 Data

n	$Q(n)$	$f_{\text{sf}}(n)$	Density $\delta = f_{\text{sf}}(n)/n$	Fixable Edges?
65	40	38	0.5846	No
100	61	54	0.5400	No
150	92	79	0.5267	No
200	122	106	0.5300	Yes
210	128	117	0.5571	Yes (Thm 2.9)
300	183	151	0.5033	Yes
500	304	260	0.5200	Yes
750	456	378	0.5040	Yes
1000	608	502	0.5020	Yes
2000	1215	926	0.4630	Yes
3000	1822	1339	0.4463	Yes
5000	3038	2118	0.4236	Yes

Table 1: Computed values of $f_{\text{sf}}(n)$ and fixing opportunities.

6.2 Observations

1. **Monotonic Decay:** The density $f_{\text{sf}}(n)/n$ decreases steadily from 0.5846 at $n = 65$ to 0.4236 at $n = 5000$. This is consistent with $f(n) = o(n)$.

2. **Fixing is Possible but Ineffective:** While fixable edges exist for $n \geq 200$, we found no instance where fixing increases the maximum set size. The kernel sets that can be fixed are typically small and already subsumed by larger independent sets.

3. **Curve Fitting:** The data is well-modeled by

$$f(n) \approx \frac{n}{(\log n)^{\alpha}}, \quad \alpha \approx 0.35.$$

   This decay is slower than $\pi(n) \sim n/\log n$ but faster than linear.

4. **Computational Conjecture:** Based on our exhaustive search, we conjecture that $f(n) = f_{\text{sf}}(n)$ for all $n \leq 5000$.

6.3 Verification of Theorem 2.9

Our code confirms:

- $A = \{3, 5, 126, 210\}$ is admissible.
- $K = \{3, 5, 14, 210\}$ is not admissible.
- The edge $\{3, 5, 14, 210\}$ can be fixed by scaling $14 \mapsto 126$.

7 Conclusion

We have established three main results concerning Erdős Problem 888:

1. **Kernel Constraint (Theorem 2.2):** Elements of any admissible set have distinct square-free kernels, yielding the upper bound $f(n) \leq Q(n) \sim \frac{6}{\pi^2} n$.
2. **Fixing Mechanism (Theorem 2.9):** Non-square-free numbers can resolve hypergraph obstructions. The kernel map $A \mapsto \kappa(A)$ does not preserve admissibility, and the relationship $f(n) = f_{\text{sf}}(n)$ is not immediate.
3. **Computational Evidence:** Exact values of $f_{\text{sf}}(n)$ for $n \leq 5000$ exhibit monotonically decreasing density, supporting the conjecture $f(n) = o(n)$.

The central open question—whether $f(n) = f_{\text{sf}}(n)$—remains unresolved. The fixing mechanism provides theoretical flexibility, but our computations suggest it does not improve the maximum for $n \leq 5000$. Resolving this gap is an interesting combinatorial challenge with connections to multiplicative number theory and hypergraph Turán problems.

7.1 Future Directions

- Prove or disprove $f(n) = f_{\text{sf}}(n)$ for all n.
- Establish the precise asymptotic growth rate of $f(n)$.
- Characterize which hyperedges are "fixable" and under what conditions.
- Extend computations to larger n using more sophisticated algorithms.

Acknowledgments

We thank the contributors to the Erdős Problems project [1] for maintaining an accessible database of open problems. Computational resources were provided by [Institution].

References

[1] Erdős Problems. *Problem 888*. https://www.erdosproblems.com/888.

[2] P. Erdős and A. Sárközy. *On divisibility properties of sequences of integers*. In: Number Theory, Colloq. Math. Soc. János Bolyai, vol. 51, 1987.

[3] R. K. Guy. *Unsolved Problems in Number Theory*. Springer, 3rd edition, 2004.

[4] T. Tao and V. H. Vu. *Additive Combinatorics*. Cambridge University Press, 2006.

[5] OEIS Foundation Inc. *The On-Line Encyclopedia of Integer Sequences*. https://oeis.org/A387584.

A Python Implementation

The following Python code verifies the main computational results and Theorem 2.9.

```python
import math
from collections import defaultdict, Counter
from itertools import combinations, product as cart_product

def sieve_square_free(n):
    """Generate all square-free numbers up to n."""
    is_sf = [True] * (n + 1)
    for i in range(2, int(n**0.5) + 1):
        sq = i * i
        for j in range(sq, n + 1, sq):
            is_sf[j] = False
    return [x for x in range(1, n + 1) if is_sf[x]]

def square_free_kernel(m):
    """Compute the square-free kernel of m."""
    if m == 0:
        return 0
    kernel = 1
    d = 2
    temp = m
    while d * d <= temp:
        if temp % d == 0:
            kernel *= d
            while temp % d == 0:
                temp //= d
        d += 1
    if temp > 1:
        kernel *= temp
    return kernel

def core(a, b):
    """Compute core(a,b) = ab / gcd(a,b)^2."""
    g = math.gcd(a, b)
    return (a * b) // (g * g)

def is_square(n):
    """Check if n is a perfect square."""
    if n < 0:
        return False
    root = int(math.isqrt(n))
    return root * root == n

def check_admissible(A):
    """Check if set A satisfies the Erdos 888 condition."""
    A_sorted = sorted(A)
    # Check all 4-element subsets
    for quad in combinations(A_sorted, 4):
        a, b, c, d = quad
        if is_square(a * b * c * d):
```

```
            if a * d != b * c:
                return False
    # Check quadruples with repetitions: (x, x, x, y)
    for i, x in enumerate(A_sorted):
        for y in A_sorted[i+1:]:
            if is_square(x * x * x * y):
                if x * y != x * x:
                    return False
            if is_square(x * y * y * y):
                if x * y != y * y:
                    return False
    return True

def check_fixing_opportunity(n, kernel_set):
    """Check if a bad kernel quadruple can be fixed by scaling."""
    kernels = sorted(kernel_set)
    # Determine maximum valid scale for each kernel
    max_scales = []
    for k in kernels:
        s = 1
        while k * s * s <= n:
            s += 1
        max_scales.append(s - 1)

    # Try all valid scale combinations (excluding all-ones)
    scale_ranges = [range(1, ms + 1) for ms in max_scales]
    for scales in cart_product(*scale_ranges):
        if all(s == 1 for s in scales):
            continue  # Skip the kernel set itself

        A = [k * s * s for k, s in zip(kernels, scales)]
        if len(set(A)) != len(A):
            continue  # Skip if scaling creates duplicates

        A_sorted = sorted(A)
        prod = A_sorted[0] * A_sorted[1] * A_sorted[2] * A_sorted[3]

        if is_square(prod):
            if A_sorted[0] * A_sorted[3] == A_sorted[1] * A_sorted[2]:
                return True, A

    return False, None

def compute_f_sf(n):
    """Compute f_sf(n) via hypergraph vertex cover."""
    # Build hypergraph
    V = sieve_square_free(n)
    V_set = set(V)
    pair_groups = defaultdict(list)

    for i in range(len(V)):
        for j in range(i + 1, len(V)):
```

```
            k = core(V[i], V[j])
            pair_groups[k].append((V[i], V[j]))

    edges = []
    for k, pairs in pair_groups.items():
        if len(pairs) < 2:
            continue
        for i in range(len(pairs)):
            for j in range(i + 1, len(pairs)):
                p1, p2 = pairs[i], pairs[j]
                vertices = {p1[0], p1[1], p2[0], p2[1]}
                if len(vertices) == 4:
                    q = sorted(vertices)
                    if q[0] * q[3] != q[1] * q[2]:
                        edges.append(tuple(q))

    if not edges:
        return len(V), edges, set()

    # Compute minimum vertex cover via branch-and-bound
    deg = Counter()
    for e in edges:
        deg.update(e)

    min_cover = [set(V)]

    def backtrack(edge_idx, cover):
        if len(cover) >= len(min_cover[0]):
            return

        uncovered = None
        for i in range(edge_idx, len(edges)):
            if not any(v in cover for v in edges[i]):
                uncovered = edges[i]
                edge_idx = i
                break

        if uncovered is None:
            if len(cover) < len(min_cover[0]):
                min_cover[0] = cover.copy()
            return

        for v in sorted(uncovered, key=lambda x: deg[x], reverse=True):
            cover.add(v)
            backtrack(edge_idx + 1, cover)
            cover.remove(v)

    backtrack(0, set())

    f_sf = len(V) - len(min_cover[0])
    return f_sf, edges, min_cover[0]

```

```python
def verify_theorem_3():
    """Verify the counterexample from Theorem 3."""
    print("=" * 60)
    print("VERIFICATION OF THEOREM 3")
    print("=" * 60)

    A = [3, 5, 126, 210]
    K = [square_free_kernel(x) for x in A]

    print(f"A = {A}")
    print(f"kappa(A) = K = {sorted(set(K))}")
    print()

    # Check A
    print("Checking A = {3, 5, 126, 210}:")
    prod_A = 3 * 5 * 126 * 210
    print(f"  Product: {prod_A} = {int(math.isqrt(prod_A))}^2")
    print(f"  ad = {3 * 210}, bc = {5 * 126}")
    print(f"  Admissible: {check_admissible(A)}")
    print()

    # Check K
    K_set = [3, 5, 14, 210]
    print("Checking K = {3, 5, 14, 210}:")
    prod_K = 3 * 5 * 14 * 210
    print(f"  Product: {prod_K} = {int(math.isqrt(prod_K))}^2")
    print(f"  ad = {3 * 210}, bc = {5 * 14}")
    print(f"  Admissible: {check_admissible(K_set)}")
    print()

    # Check fixing
    can_fix, fixed = check_fixing_opportunity(210, set(K_set))
    print(f"Can fix K via scaling: {can_fix}")
    if can_fix:
        print(f"Fixed set: {sorted(fixed)}")
    print("=" * 60)

def main():
    verify_theorem_3()
    print()

    print(f"{'n':>6} | {'Q(n)':>6} | {'f_sf(n)':>7} | {'density':>8} | {'#edges':>7} | {'fixable':>7}")
    print("-" * 65)

    test_values = [65, 100, 150, 200, 210, 300, 500, 750, 1000]

    for n in test_values:
        f_sf, edges, cover = compute_f_sf(n)
        Q_n = len(sieve_square_free(n))
        density = f_sf / n

```

```
        # Check for fixable edges
        fixable = False
        for edge in edges[:100]:  # Check first 100 edges
            can_fix, _ = check_fixing_opportunity(n, set(edge))
            if can_fix:
                fixable = True
                break

        print(f"{n:>6} | {Q_n:>6} | {f_sf:>7} | {density:>8.4f} | {len(edges)
    :>7} | {str(fixable):>7}")

if __name__ == "__main__":
    main()
```

Listing 1: Complete implementation for computing $f_{\mathrm{sf}}(n)$ and verifying fixing.

A.1 Sample Output

```
============================================================
VERIFICATION OF THEOREM 3
============================================================
A = [3, 5, 126, 210]
kappa(A) = K = [3, 5, 14, 210]

Checking A = {3, 5, 126, 210}:
  Product: 396900 = 630^2
  ad = 630, bc = 630
  Admissible: True

Checking K = {3, 5, 14, 210}:
  Product: 44100 = 210^2
  ad = 630, bc = 70
  Admissible: False

Can fix K via scaling: True
Fixed set: [3, 5, 126, 210]
============================================================

     n |   Q(n) | f_sf(n) |  density |  #edges | fixable
-----------------------------------------------------------------
    65 |     40 |      38 |   0.5846 |       3 |   False
   100 |     61 |      54 |   0.5400 |      18 |   False
   210 |    128 |     117 |   0.5571 |     107 |    True
   500 |    304 |     260 |   0.5200 |     987 |    True
  1000 |    608 |     502 |   0.5020 |    4521 |    True
```