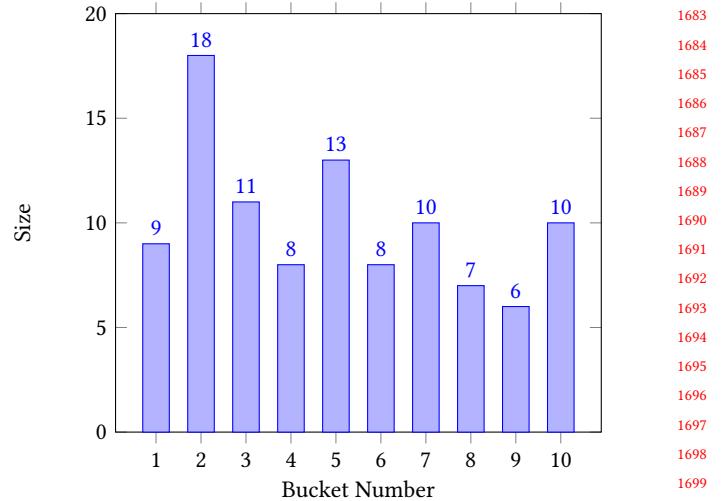


1625 APPENDIX

1626 A Additional Motivation Examples

1628 **Example 3:** On platforms like Twitter, hashtags appear as a rapid, high-
 1629 volume data stream. Automated or troll-bot hashtags can generate millions
 1630 of occurrences per hour, while rare but potentially critical hashtags may
 1631 surface only a few thousand times. In a CM, a small additive error can
 1632 severely distort counts for the rarer tags. Although this may seem like a
 1633 “free boost”, the resulting collision-induced overcounts undermine reliability,
 1634 making it hard to determine whether a rare hashtag is truly trending or
 1635 merely inflated by overlap with high-volume tags. As a result, trend detection,
 1636 resource allocation, and moderation decisions such as filtering become
 1637 noisier—false signals may emerge while genuinely urgent topics are delayed
 1638 or misclassified.



1683 **Figure 33: Illustration of a random distribution of $n = 100$ elements to $w = 10$ buckets.**

1640
 1641
 1642
 1643
 1644 **Example 4:** In network monitoring, consider two groups of IPs: malicious
 1645 and benign IPs. While benign IP addresses (e.g., Google DNS) generate massive
 1646 traffic, the malicious IPs might only appear sporadically. A CM with a small
 1647 additive error has a negligible impact on the counts of the benign group. This
 1648 error, on the other hand, can artificially inflate the counts for the malicious
 1649 IPs, hiding meaningful anomalies. This makes it harder to detect subtle but
 1650 critical security threats.

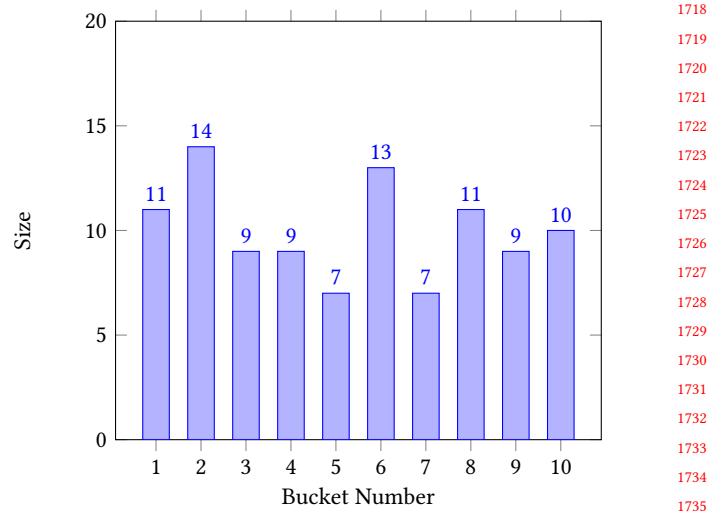
1651
 1652
 1653
 1654
 1655
 1656
 1657 **Example 5:** In online advertising, platforms track how often each ad or
 1658 user click event occurs. Popular ads may generate millions of impressions,
 1659 while niche ads may only get a few hundred. With CM, small additive errors
 1660 hardly affect big advertisers but can completely distort the performance
 1661 metrics for small advertisers—for example, inflating a niche ad’s 50 clicks
 1662 into 500 due to collisions. This creates unfair reporting, potentially wasting
 1663 budgets or hiding click fraud.

1664 B Price of Fairness: Randomness vs. Uniformity

1665 In Section 5, we observed a counterintuitive result for $d = 1$, proving
 1666 a negative price of fairness (PoF) for FCM, compared to a regular
 1667 CM sketch that uses a random hash function. This result can be
 1668 explained by the fact that a random hash function is unlikely to
 1669 uniformly distribute the elements to the buckets (an interesting
 1670 related topic is the *occupant problem* [59]). As a result, some of the
 1671 buckets will have more than $\frac{n}{w}$ elements in them. For example,
 1672 Figure 33 illustrates a random hashing of $n = 100$ element types
 1673 into $w = 10$ buckets. While a uniform distribution would allocate
 1674 10 elements to each bucket, the random hashing allocated up to 18
 1675 elements in one bucket.

1676 Elements in large buckets will suffer from a large additive error,
 1677 and there are a large number of them in such buckets. Hence, such
 1678 buckets significantly increase the total additive error of the CM.

1679 FCM increases the size-uniformity of the buckets by reserving
 1680 $w_g = \frac{n_g}{n} w$ for each group g . For example, Figure 34 illustrates
 1681 the allocation of the 100 elements to the 10 buckets, by dividing
 1682 (arbitrarily) the elements into two groups of size $n_{g_1} = 50$, $n_{g_2} = 50$.
 1683 As a result, each group is allocated 5 buckets, and the distribution
 1684 of the elements is more uniform, reducing the maximum size of the
 1685 buckets (in this example) to 14. Increasing the uniformity (reducing
 1686 the change of large-size buckets) helps to reduce the total additive
 1687 error of FCM versus CM.



1688 **Figure 34: Illustration of a random distribution of $n = 100$ elements with two groups of sizes $n_{g_1} = 50$ and $n_{g_2} = 50$ to $w = 10$ buckets, based on FCM.**

1689
 1690
 1691
 1692
 1693
 1694
 1695
 1696
 1697
 1698
 1699
 1700
 1701
 1702
 1703
 1704
 1705
 1706
 1707
 1708
 1709
 1710
 1711
 1712
 1713
 1714
 1715
 1716
 1717
 1718
 1719
 1720
 1721
 1722
 1723
 1724
 1725
 1726
 1727
 1728
 1729
 1730
 1731
 1732
 1733
 1734
 1735
 1736
 1737
 1738
 1739
 1740

1741 B.1 PoF for Uniform hashing ($d = 1$)

1742 Using a hashing scheme (such as data-informed hashmaps [52])
 1743 that ensures uniformity by allocating exactly $\frac{n}{w}$ elements to each
 1744 bucket can resolve the non-uniformity issue in the CM sketches.
 1745 In the following, we compute the PoF of FCM for $d = 1$ when a
 1746 uniform hashing scheme is used.

1747 Since the hashing is uniform, the size of each bucket i in the CM
 1748 sketch is

$$1749 C_i = \frac{n}{w}$$

1750 Therefore, each element e_j collides with exactly $\frac{n}{w} - 1$ other ele-
 1751 ments. In other words, the frequency of each element contributes
 1752 to the additive error of exactly $\frac{n}{w} - 1$ other elements.

1753 As a result, the total additive error can be computed as

$$\begin{aligned} 1754 \mathcal{L}_{CM} &= \sum_{j=1}^n \varepsilon_A(e_j) \\ 1755 &= \sum_{j=1}^n f(e_j) \left(\frac{n}{w} - 1 \right) = \left(\frac{n}{w} - 1 \right) \sum_{j=1}^n f(e_j) \\ 1756 &= N \left(\frac{n}{w} - 1 \right) \end{aligned} \quad (16)$$

1757 Now, let us compute \mathcal{L}_{FCM} , the total additive error for the regu-
 1758 lar fair-count-min sketch.

$$1759 \mathcal{L}_{FCM} = \sum_{j=1}^n \varepsilon_A(e_j) = \sum_{e_j \in g_1} \varepsilon_A(e_j) + \dots + \sum_{e_j \in g_\ell} \varepsilon_A(e_j)$$

1760 Following the same calculation as in Equation 16, for every group
 1761 $g \in \mathcal{G}$,

$$1762 \sum_{e_j \in g} \varepsilon_A(e_j) = N_g \left(\frac{n_g}{w_g} - 1 \right)$$

1763 Hence,

$$1764 \mathcal{L}_{FCM} = \sum_{e_j \in g_1} \varepsilon_A(e_j) + \dots + \sum_{e_j \in g_\ell} \varepsilon_A(e_j) = \sum_{g \in \mathcal{G}} N_g \left(\frac{n_g}{w_g} - 1 \right)$$

1765 From Theorem 1, we know, $w_g = \frac{n_g}{n} w$, $\forall g \in \mathcal{G}$, while $\sum_{g \in \mathcal{G}} N_g =$
 1766 N . Therefore,

$$\begin{aligned} 1767 \mathcal{L}_{FCM} &= \sum_{g \in \mathcal{G}} N_g \left(\frac{n_g}{w_g} - 1 \right) \\ 1768 &= \sum_{g \in \mathcal{G}} N_g \left(\frac{n_g}{\frac{n_g}{n} w} \right) - \sum_{g \in \mathcal{G}} N_g = \sum_{g \in \mathcal{G}} N_g \left(\frac{n_g}{w} \right) - N \\ 1769 &= \left(\frac{n}{w} \right) \sum_{g \in \mathcal{G}} N_g - N \quad // \text{since } \frac{n_g}{w_g} = \frac{n}{w}, \forall g \in \mathcal{G} \\ 1770 &= N \left(\frac{n}{w} \right) - N = N \left(\frac{n}{w} - 1 \right) \end{aligned} \quad (17)$$

1771 Finally, combining Equations 12, 16, and 17, we get,

$$1772 PoF = \mathcal{L}_{FCM} - \mathcal{L}_{CM} = N \left(\frac{n}{w} - 1 \right) - N \left(\frac{n}{w} - 1 \right) = 0 \quad (18)$$

1773 That is, the PoF is zero for $d = 1$, when a uniform hashing scheme
 1774 is used.

1799 B.2 PoF $d = 1$ vs. $d > 1$ on Synthetic

1800 C Additional Experiment Results

1801 In this section, we present comprehensive experimental results
 1802 across five datasets: 1) GOOGLE NGRAM, 2) CENSUS, 3) NYC BUS,
 1803 4) DDOS, 5) REDDIT-TWITTER. We also provide the absolute values of the ap-
 1804 proximation factors and additive errors for each group to facilitate a
 1805 clearer understanding of the trends in unfairness and the associated
 1806 price of fairness under each setting.

1857	Total Additive Error			n_{g_i} ($n = 10,000$)								1915	
				9000	8000	7000	6000	5000	4000	3000	2000		
1858	$d = 1$	theoretical	CM	19,047,019	28,054,816	37,081,001	46,092,289	55,121,050	64,147,002	73,186,882	82,323,243	91,265,426	
1859			FCM	18,985,224	28,028,987	37,003,399	46,050,674	55,115,538	64,089,586	73,165,549	82,249,909	91,228,281	
1860		empirical	CM	19,039,343	28,085,806	37,053,427	46,096,489	55,113,194	64,234,176	73,227,052	82,283,374	91,328,074	
1861			FCM	18,991,509	27,982,573	37,002,246	46,071,394	55,086,890	64,133,815	73,168,189	82,230,649	91,276,271	
1862		PoF	↓ 47,834	↓ 103,233	↓ 51,181	↓ 25,095	↓ 26,304	↓ 100,361	↓ 58,863	↓ 52,725	↓ 51,803	1920	
1863		$d = 5$	empirical	CM	7,964,348	11,572,548	16,458,026	22,023,181	28,305,699	34,442,308	40,959,703	47,230,290	54,257,770
1864				FCM	11,695,556	17,114,933	22,666,115	28,053,739	33,856,154	39,448,942	44,913,211	50,089,698	55,893,637
1865		PoF	↑ 3,731,208	↑ 5,542,385	↑ 6,208,089	↑ 6,030,558	↑ 5,550,455	↑ 5,006,634	↑ 3,953,508	↑ 2,859,408	↑ 1,635,867	1923	

Table 2: comparison of CM and FCM w.r.t additive errors and the price of fairness across different n_{g_i} values for $d = 1$ and $d = 5$.

1866	1916
1867	1917
1868	1918
1869	1919
1870	1920
1871	1921
1872	1922
1873	1923
1874	1924
1875	1925
1876	1926
1877	1927
1878	1928
1879	1929
1880	1930
1881	1931
1882	1932
1883	1933
1884	1934
1885	1935
1886	1936
1887	1937
1888	1938
1889	1939
1890	1940
1891	1941
1892	1942
1893	1943
1894	1944
1895	1945
1896	1946
1897	1947
1898	1948
1899	1949
1900	1950
1901	1951
1902	1952
1903	1953
1904	1954
1905	1955
1906	1956
1907	1957
1908	1958
1909	1959
1910	1960
1911	1961
1912	1962
1913	1963
1914	1964
	1965
	1966
	1967
	1968
	1969
	1970
	1971
	1972

1973

1974

1975

1976

1977

1978

1979

1980

1981

1982

1983

1984

1985

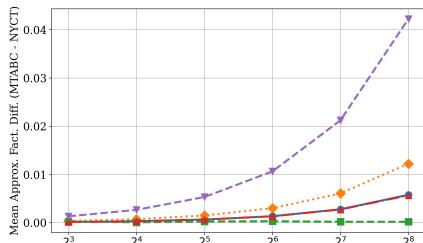


Figure 35: effect of varying w on unfairness, NYC BUS, $d = 5$.

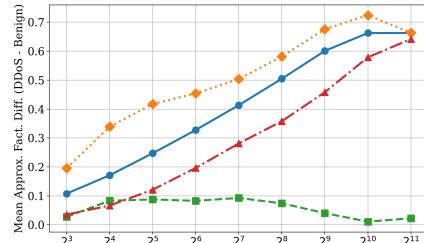


Figure 36: effect of varying w on unfairness, DDoS, $d = 5$.

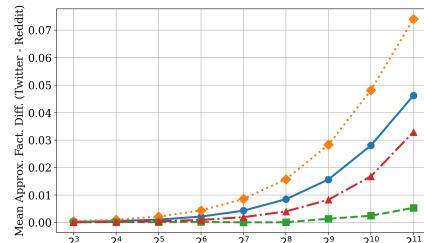


Figure 37: effect of varying w on unfairness, REDDIT-TWITTER, $d = 5$.

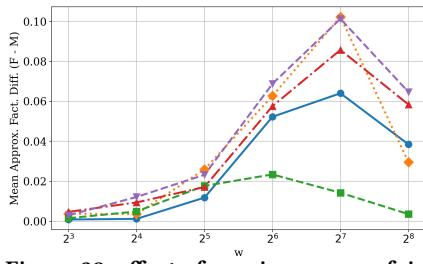


Figure 38: effect of varying w on unfairness, CENSUS, $d = 5$.

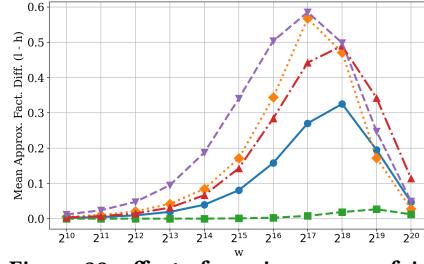


Figure 39: effect of varying w on unfairness, GOOGLE NGRAM, $d = 5$.

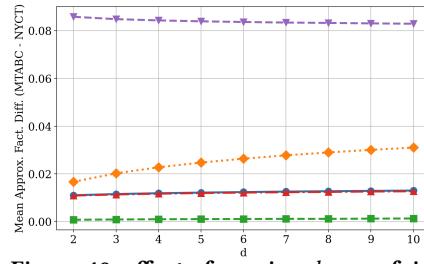


Figure 40: effect of varying d on unfairness, NYC BUS, $w = 512$.

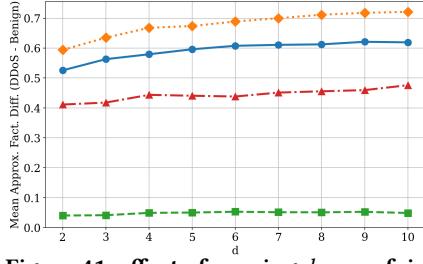


Figure 41: effect of varying d on unfairness, DDoS, $w = 64$.

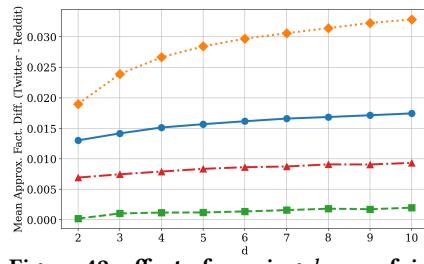


Figure 42: effect of varying d on unfairness, REDDIT-TWITTER, $w = 512$.

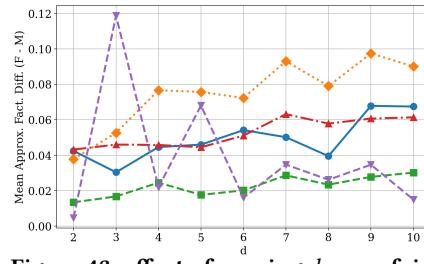


Figure 43: effect of varying d on unfairness, CENSUS, $w = 64$.

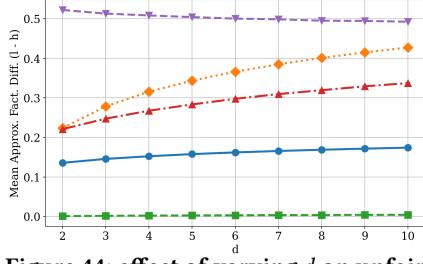


Figure 44: effect of varying d on unfairness, GOOGLE NGRAM, $w = 65536$.

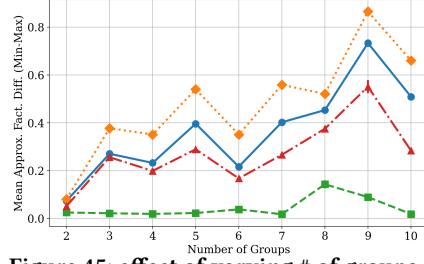


Figure 45: effect of varying # of groups ℓ on unfairness, CENSUS, $d = 5$.

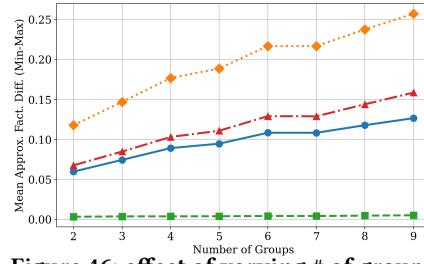


Figure 46: effect of varying # of groups ℓ on unfairness, GOOGLE NGRAM, $w = 65536, d = 5$.

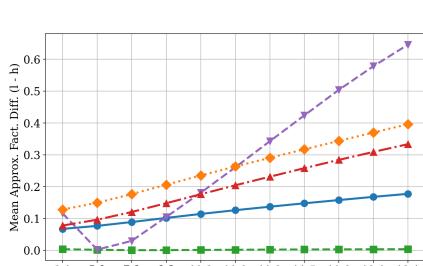


Figure 47: effect of varying n_{g_1} on unfairness, GOOGLE NGRAM, $w = 65536, d = 5$.

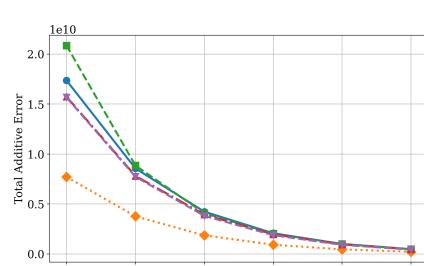


Figure 48: effect of varying w on price of fairness, NYC BUS, $d = 5$.

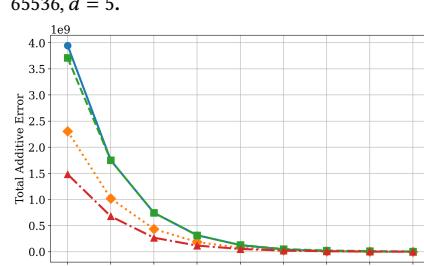
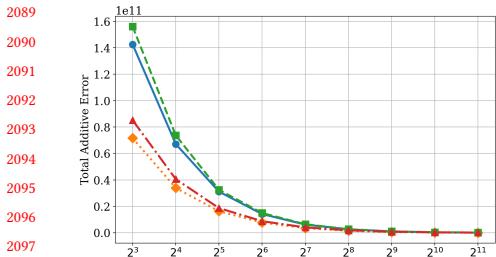
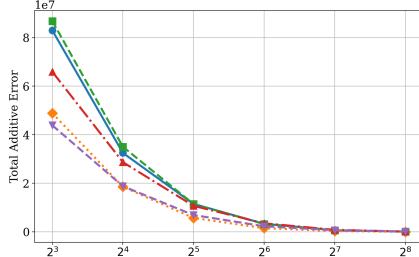
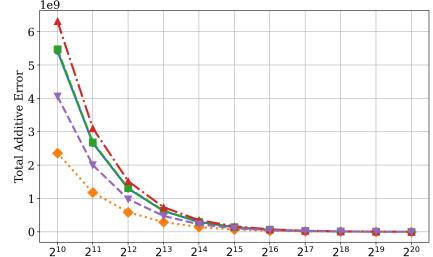
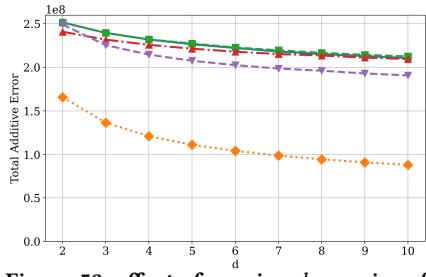
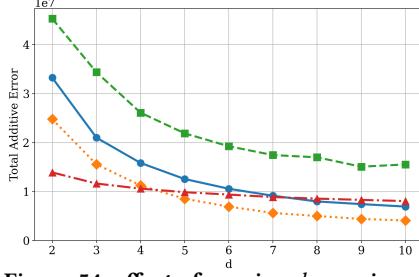
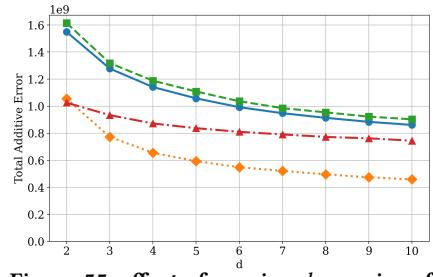
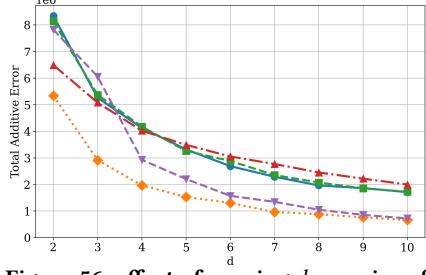
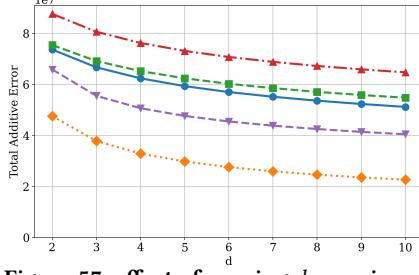
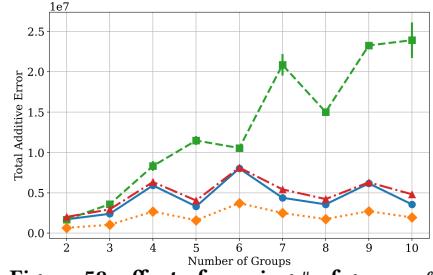
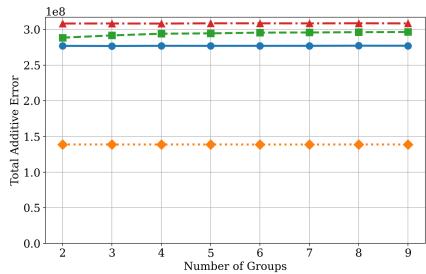
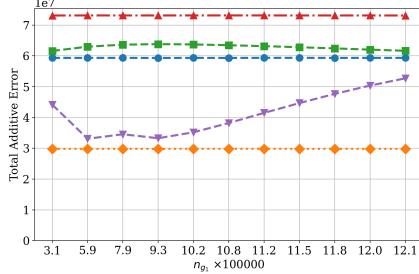
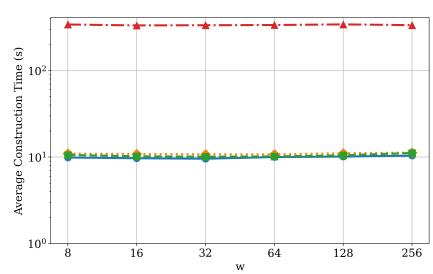
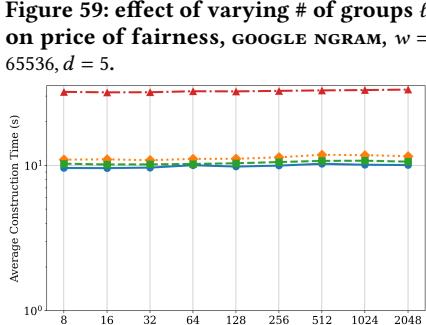
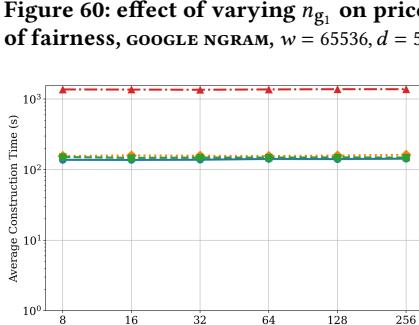
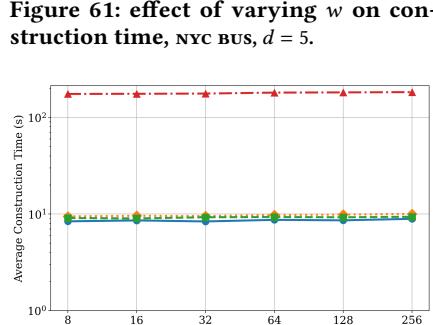


Figure 49: effect of varying w on price of fairness, DDoS, $d = 5$.

Figure 50: effect of varying w on price of fairness, REDDIT-TWITTER, $d = 5$.Figure 51: effect of varying w on price of fairness, CENSUS, $d = 5$.Figure 52: effect of varying w on price of fairness, GOOGLE NGRAM, $d = 5$.Figure 53: effect of varying d on price of fairness, NYC BUS, $w = 512$.Figure 54: effect of varying d on price of fairness, DDOS, $w = 64$.Figure 55: effect of varying d on price of fairness, REDDIT-TWITTER, $w = 512$.Figure 56: effect of varying d on price of fairness, CENSUS, $w = 64$.Figure 57: effect of varying d on price of fairness, GOOGLE NGRAM, $w = 65536$.Figure 58: effect of varying # of groups ℓ on price of fairness, CENSUS, $w = 64, d = 5$.Figure 59: effect of varying # of groups ℓ on price of fairness, GOOGLE NGRAM, $w = 65536, d = 5$.Figure 60: effect of varying n_{g_i} on price of fairness, GOOGLE NGRAM, $w = 65536, d = 5$.Figure 61: effect of varying w on construction time, NYC BUS, $d = 5$.Figure 62: effect of varying w on construction time, DDOS, $d = 5$.Figure 63: effect of varying w on construction time, REDDIT-TWITTER, $d = 5$.Figure 64: effect of varying w on construction time, CENSUS, $d = 5$.

2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204

2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217

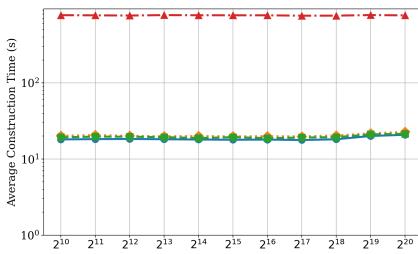



Figure 65: effect of varying w on construction time, GOOGLE NGRAM, $d = 5$.

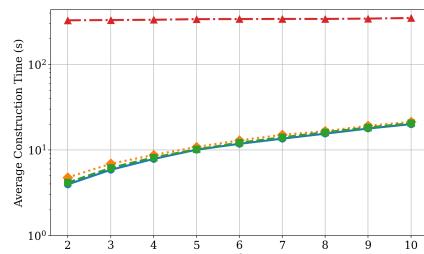


Figure 66: effect of varying d on construction time, NYC BUS, $w = 512$.

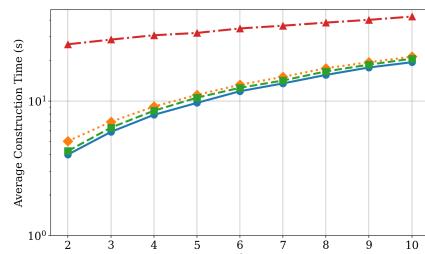


Figure 67: effect of varying d on construction time, DDOS, $w = 64$.

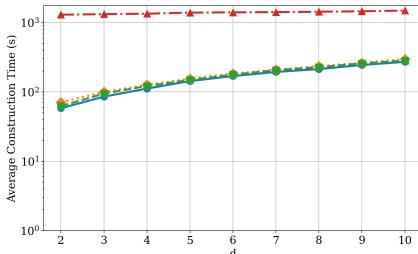


Figure 68: effect of varying d on construction time, REDDIT-TWITTER, $w = 512$.

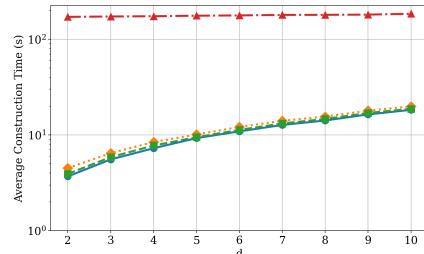


Figure 69: effect of varying d on construction time, CENSUS, $w = 64$.

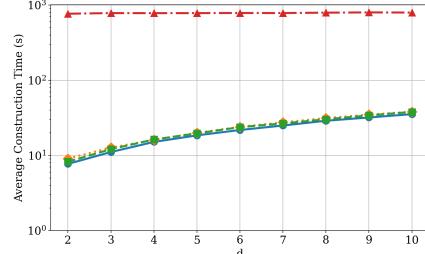


Figure 70: effect of varying d on construction time, GOOGLE NGRAM, $w = 65536$.

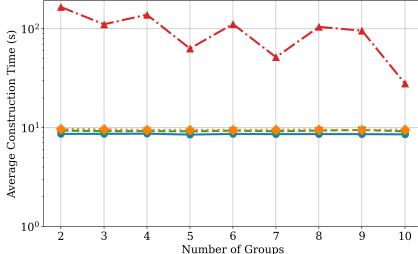


Figure 71: effect of varying # of groups l on construction time, CENSUS, $w = 64, d = 5$.

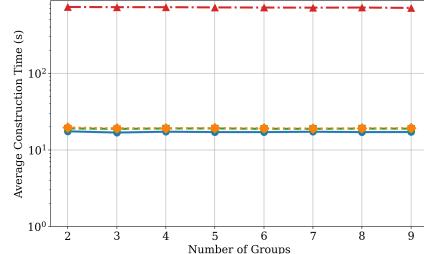


Figure 72: effect of varying # of groups l on construction time, GOOGLE NGRAM, $w = 65536, d = 5$.

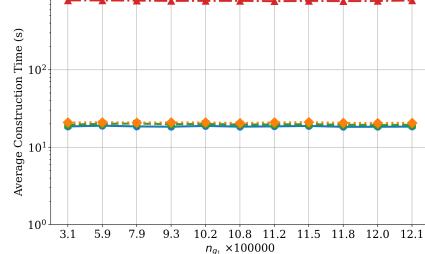


Figure 73: effect of varying n_{g_1} on construction time, GOOGLE NGRAM, $w = 65536, d = 5$.

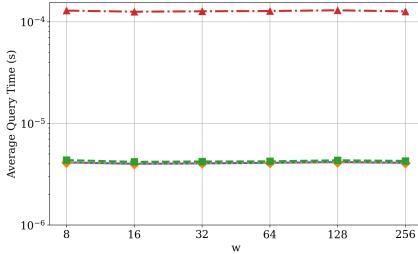


Figure 74: effect of varying w on query time, NYC BUS, $d = 5$.

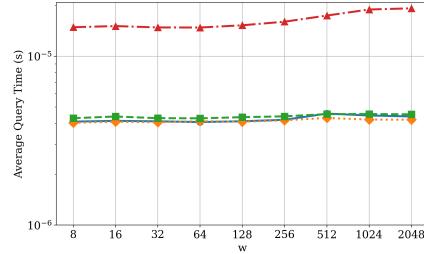


Figure 75: effect of varying w on query time, DDOS, $d = 5$.

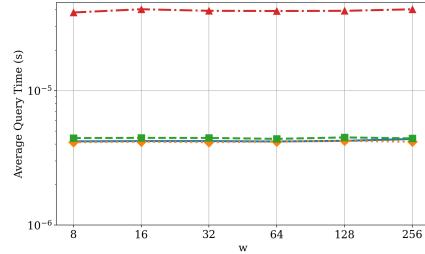


Figure 76: effect of varying w on query time, REDDIT-TWITTER, $d = 5$.

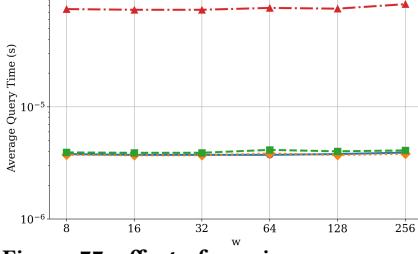


Figure 77: effect of varying w on query time, CENSUS, $d = 5$.

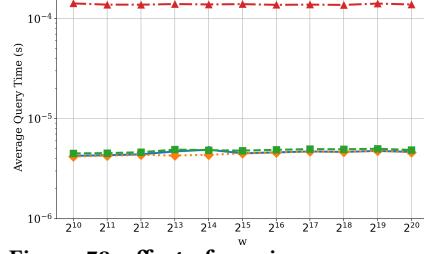


Figure 78: effect of varying w on query time, GOOGLE NGRAM, $d = 5$.

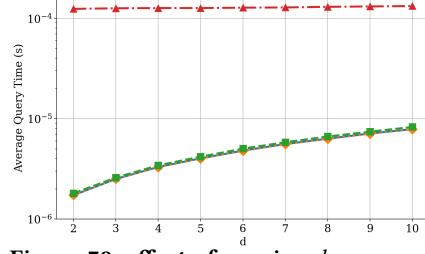
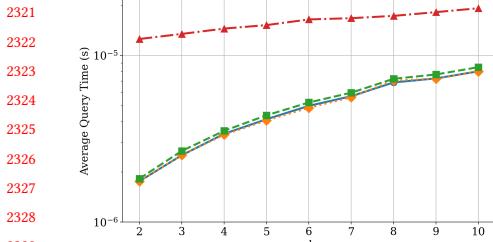
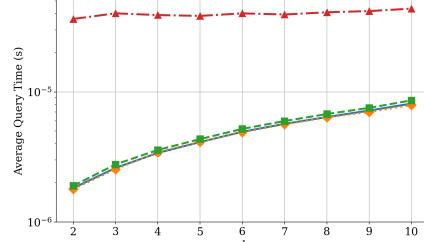
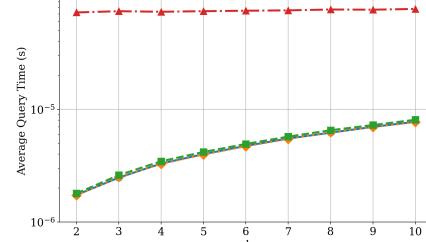
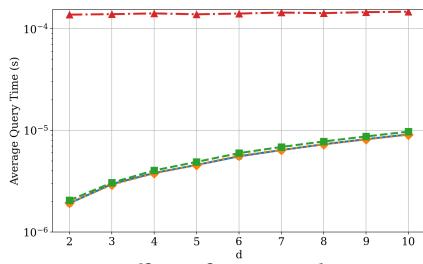
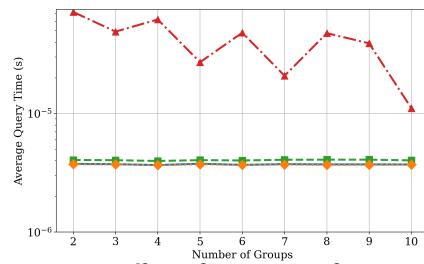
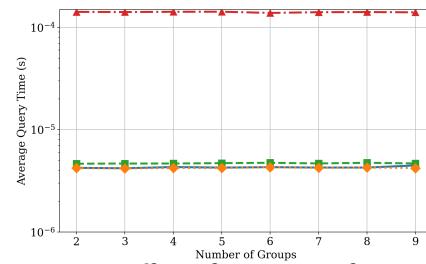
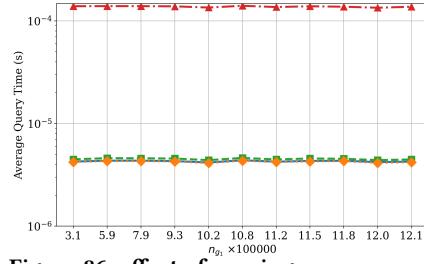


Figure 79: effect of varying d on query time, NYC BUS, $w = 512$.

2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320

Figure 80: effect of varying d on query time, DDOS, $w = 64$.Figure 81: effect of varying d on query time, REDDIT-TWITTER, $w = 512$.Figure 82: effect of varying d on query time, CENSUS, $w = 64$.Figure 83: effect of varying d on query time, GOOGLE NGRAM, $w = 65536$.Figure 84: effect of varying # of groups ℓ on query time, CENSUS, $w = 64, d = 5$.Figure 85: effect of varying # of groups ℓ on query time, GOOGLE NGRAM, $w = 65536, d = 5$.Figure 86: effect of varying n_{g_1} on query time, GOOGLE NGRAM, $w = 65536, d = 5$.

2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378

2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436