AMCS_Optimizer

Raymond Sun

2025-06-17

Helper Functions

```
import numpy as np
import itertools
from itertools import product
import random
def compute_t_G_W(H, W_block):
    # Compute t for nxn graphon, brute force n^|V(H)| calculations
    n = H.shape[0]
    edges = [(i, j) for i in range(n) for j in range(i + 1, n) if H[i, j] == 1 or H[j, i] ==
    num_blocks = W_block.shape[0]
    block_volume = 1.0 / num_blocks
    t = 0.0
    for assignment in product(range(num_blocks), repeat=n):
        prob = 1.0
        for (u, v) in edges:
            prob *= W_block[assignment[u], assignment[v]]
        t += prob * (block_volume ** n)
    return t

def stretch(W, p):
    """
    Given a 4x4 matrix W, return a new matrix where each entry is stretched further from the
    """
    W_stretched = W + (p - 1) * (W - np.mean(W))
    return W_stretched

def sidorenko_ratio(H, W_block):
    # Computes p^|E(H)| / t - 1
```

```
    t = compute_t_G_W(H, W_block)
    p = np.mean(W_block)
    num_edges = int(np.sum(H) // 2)
    return np.log(p ** num_edges / t)
```

Search Parameters and Logic

Here is some notes on my modification of Iurii's AMCS code:

1. Run the script with `python amcs_for_graphon_demo.py` , main inputs and parameters to adjust are `H` , `W_initial` , `max_depth`, `max_level`, `max_steps`, `cur_ep`, `min_length` .
2. Each perturbation step is a random 4*4 matrix with length (L2 norm) `cur_ep`. From the current best W, the algorithm makes `level` * `ep` random steps to search for an improvement, then `depth` + 1. When `depth` reaches `max_depth` , it resets to 0 and lets `level` + 1. When an improvement is found, `depth` is reset to 0 without increasing `level`. When `level` reaches `max_level`, `ep` becomes 80% of its original value (or 60% if there has been no improvement in the last cycle) since the current step may be too large.
3. Currently the weights stabilize after around 10-15 minutes, while the loop reaches an ending criteria after around an hour. It exits the loop when $\|W-1\|_\infty < \frac{1}{4m}$ or 3 full loops are run without any improvement. When the first condition is met, usually a Sidorenko ratio around $1.5 * 10^{-8}$ to $3 * 10^{-8}$ is reached.

   This is much smaller than that of a random matrix of similar distance to the uniform weight matrix (around order of 10^{-4}), but it's yet to be determined if there is some special structure to be found for these optimized matrices and if it holds clues for patterns close to violation.

Potential next steps: H could be changed to test other graphs with 12 or less nodes that haven't been verified, or W could be changed to a 5*5 graphon (should converge in a few hours with the current framework).

This framework can be extended to optimize for EFX and the Second Neighborhood Conjecture by tweaking the score function and perturbation logic.

Optimization results from final version of AMCS:

```
H = np.array([
    [0,0,0,0,0,0,0,1,1,1],
    [0,0,0,0,0,1,0,0,1,1],
    [0,0,0,0,0,1,1,0,0,1],
    [0,0,0,0,0,1,1,1,0,0],
    [0,0,0,0,0,0,1,1,1,0],
```

```
    [0,1,1,1,0,0,0,0,0,0],
    [0,0,1,1,1,0,0,0,0,0],
    [1,0,0,1,1,0,0,0,0,0],
    [1,1,0,0,1,0,0,0,0,0],
    [1,1,1,0,0,0,0,0,0,0]
])
W = np.array([
  [1.006698144463033850e+00,9.828284542082723618e-01,1.006953717554858629e+00,1.003520601558
[9.828284542082723618e-01,1.007337069931104390e+00,1.011365901909568432e+00,9.98476260209566
[1.006953717554858629e+00,1.011365901909568432e+00,9.842679968971830284e-01,9.97404804407160
[1.003520601558819392e+00,9.984762602095663686e-01,9.974048044071606167e-01,1.00059730901218
])
np.round(W, 4)
print(sidorenko_ratio(H, W))
#W_initial = np.array([[0.1, 1.1, 1.3, 1.2],
#[1.1, 1.3, 0.4, 1.1],
#[1.3, 0.4, 0.9, 1.2],
#[1.2, 1.1, 1.2, 0.2]])
#Search complete. Best Sidorenko ratio: -1.7497e-08
#Total elapsed time: 3097.35 seconds
#AMCS_graphon(H, best_W, max_depth= 5, max_level= 8, max_steps = 8, epsilon = 0.3)
#min length = 0.0005

W = np.array([
  [9.870859047315494461e-01,1.012793678182828838e+00,9.894975254113421714e-01,1.010617752083
[1.012793678182828838e+00,1.010332578863655684e+00,9.933433720332041084e-01,9.83506069775493
[9.894975254113421714e-01,9.933433720332041084e-01,1.016731641432941036e+00,1.00042970203232
[1.010617752083826870e+00,9.835060697754934855e-01,1.000429702032327395e+00,1.00547367593380
])
np.round(W, 4)
print(sidorenko_ratio(H, W))
#W_initial = np.array([[0.1, 1.1, 1.3, 0.8],
#[1.1, 0.5, 0.4, 1.1],
#[1.3, 0.4, 0.9, 1.2],
#[0.8, 1.1, 1.2, 1.4]])
#Search complete. Best Sidorenko ratio: -2.9797e-08
#Total elapsed time: 3336.68 seconds
#AMCS_graphon(H, best_W, max_depth= 5, max_level= 8, max_steps = 8, epsilon = 0.3)
#min_length = 0.0005

W = np.array([
  [1.005918364232344153e+00,1.008561605423080598e+00,1.001622593800778116e+00,9.838745226963
```

```
[1.008561605423080598e+00,9.956187641638900576e-01,9.862188039403909645e-01,1.009638083189973
[1.001622593800778116e+00,9.862188039403909645e-01,1.011966560148175764e+00,1.000195910494968
[9.838745226963234725e-01,1.009638083189973834e+00,1.000195910494968476e+00,1.006273272364560
])
np.round(W, 4)
print(sidorenko_ratio(H, W))
#W_initial = np.array([[0.1, 1.1, 1.3, 1.2],
#    [1.1, 1.3, 0.4, 1.1],
#    [1.3, 0.4, 0.9, 1.2],
#    [1.2, 1.1, 1.2, 0.2]])
#Early stopping: max|W - 1| < 0.0167
#Search complete. Best Sidorenko ratio: -1.5431e-08
#Total elapsed time: 4420.93 seconds
#AMCS_graphon(H, best_W, max_depth= 10, max_level= 6, max_steps = 8, epsilon = 0.3)
#min_length = 0.0004

W = np.array([
  [9.937669153101358344e-01,1.016135336240906240e+00,9.884566213567120840e-01,1.001630615749
[1.016135336240906240e+00,9.910134625158074639e-01,9.912700210631636422e-01,1.001558727266020
[9.884566213567120840e-01,9.912700210631636422e-01,1.007400418292873745e+00,1.012914183393865
[1.001630615749603770e+00,1.001558727266020687e+00,1.012914183393865297e+00,9.838881937406414
])
np.round(W, 4)
print(sidorenko_ratio(H, W))
#W_initial = np.array([
    #[0.9, 1.1, 1.3, 0.8],
    #[1.1, 0.5, 0.4, 1.4],
    #[1.3, 0.4, 0.9, 0.3],
    #[0.8, 1.4, 0.3, 1.4]])
#Early stopping: max|W - 1| < 0.0167
#Search complete. Best Sidorenko ratio: -2.0361e-08
# AMCS_graphon(H, best_W, max_depth= 10, max_level= 6, max_steps = 10, epsilon = 0.3)
#min_length = 0.0004
#Total elapsed time: 3795.58 seconds

W = np.array([
  [1.003815032309734567e+00,1.002321130250688919e+00,9.802993524557340743e-01,1.013558287856
[1.002321130250688919e+00,9.858945971800775476e-01,1.009930703536328744e+00,1.001860064505293
[9.802993524557340743e-01,1.009930703536328744e+00,1.008668286555383764e+00,1.001089088428382
[1.013558287856340057e+00,1.001860064505293524e+00,1.001089088428382290e+00,9.835048298892703
])
np.round(W, 4)
```

```
print(sidorenko_ratio(H, W))
#W_initial = np.array([[0.1, 1.1, 1.3, 1.2],
   # [1.1, 1.3, 0.4, 1.1],
   # [1.3, 0.4, 0.9, 1.2],
   # [1.2, 1.1, 1.2, 0.2]])
#Search complete. Best Sidorenko ratio: -2.6723e-08
# AMCS_graphon(H, best_W, max_depth= 6, max_level= 6, max_steps = 15, epsilon = 0.3)
#min_length = 0.0004
#Total elapsed time: 3778.65 seconds

W = np.array([
  [1.0021001887899576 58e+00,1.013100042952112645e+00,9.83974979042985 4759e-01,1.0008416000658
[1.013100042952112645e+00,9.962253851697436824e-01,1.005479119659793641e+00,9.851842122305812
[9.83974979042985 4759e-01,1.005479119659793641e+00,1.013067090262603109e+00,9.97471416128637
[1.000841600065852210e+00,9.851842122305812799e-01,9.974714161286379888e-01,1.016504595617770
])
np.round(W, 4)
print(sidorenko_ratio(H, W))
#W_initial = np.array([
    #[0.9, 1.1, 1.3, 0.8],
    #[1.1, 0.5, 0.4, 1.4],
    #[1.3, 0.4, 0.9, 0.3],
    #[0.8, 1.4, 0.3, 1.4]])
#Early stopping: max|W - 1| < 0.0167

#Search complete. Best Sidorenko ratio: -2.2294e-08
#AMCS_graphon(H, best_W, max_depth= 8, max_level= 10, max_steps = 5, epsilon = 0.3)
# min_length = 0.0004
#Total elapsed time: 4691.44 seconds
```

```
-1.749725324393597e-08
-2.9797800227136645e-08
-1.5431189334381074e-08
-2.0361656124199164e-08
-2.672246859525017e-08
-2.2294513595088403e-08
```

Optimization results from previous versions of AMCS, stretched/ shrunk to the same scale to compare their ratios

```
W = np.array([
  [9.829822690147628217e-01,1.006239824946722372e+00,9.892987303600105919e-01,1.021524806997
[1.006239824946722372e+00,9.759788132675752959e-01,1.014080386099867059e+00,1.003746041432600
[9.892987303600105919e-01,1.014080386099867059e+00,1.018932138361181794e+00,9.777298907830966
[1.021524806997454071e+00,1.003746041432600267e+00,9.777298907830966890e-01,9.968674181169788
])
sidorenko_ratio(H, stretch(W, 1 / (60 * np.max(abs(W - 1)))))

W = np.array([
[1.008086217686960051e+00,1.006769525428973910e+00,9.915366153722398046e-01,9.936051460927676
[1.006769525428973910e+00,9.812253243159484439e-01,1.006088680845625305e+00,1.005917335263637
[9.915366153722398046e-01,1.006088680845625305e+00,1.001166744147237608e+00,1.001212472498347
[9.936051460927676215e-01,1.005917335263637247e+00,1.001212472498347328e+00,9.992621628466726
])
sidorenko_ratio(H, stretch(W, 1 / (60 * np.max(abs(W - 1)))))

W = np.array([
 [9.602514917187326970e-01,1.015286782632627061e+00,1.011833006234631238e+00,1.012574322152577
[1.015286782632627061e+00,1.003895567311315684e+00,9.741647409692193449e-01,1.006689663022229
[1.011833006234631238e+00,9.741647409692193449e-01,1.012872268312449586e+00,1.001127594711219
[1.012574322152577366e+00,1.006689663022229508e+00,1.001127594711219260e+00,9.796284532124968
])
sidorenko_ratio(H, stretch(W, 1 / (60 * np.max(abs(W - 1)))))
```

```
np.float64(-6.317677290174337e-09)
```

```
W = np.array([
  [9.746938100681739048e-01,1.011048495783134049e+00,1.007385985375905069e+00,1.006898802943
[1.011048495783134049e+00,9.778949562160562659e-01,1.004556144263334838e+00,1.006470554227467
[1.007385985375905069e+00,1.004556144263334838e+00,9.796630669374898348e-01,1.008385183820106
[1.006898802943898552e+00,1.006470554227467895e+00,1.008385183820106779e+00,9.782578339505850
])
sidorenko_ratio(H, stretch(W, 1 / (60 * np.max(abs(W - 1)))))

W = np.array([
  [9.785204923248900233e-01,1.006063664598876972e+00,1.008272758742907005e+00,1.007169217624
[1.006063664598876972e+00,9.809639196219169799e-01,1.006829849932037835e+00,1.006152334412331
[1.008272758742907005e+00,1.006829849932037835e+00,9.769347070419951429e-01,1.007911030016080
[1.007169217624657831e+00,1.006152334412331317e+00,1.007911030016080955e+00,9.787831703574124
])
sidorenko_ratio(H, stretch(W, 1 / (60 * np.max(abs(W - 1)))))
```

```
W = np.array([
  [1.029190107276600541e+00,9.91572484410669141 6e-01,9.892356910190851504e-01,9.900125185850
[9.91572484410669141 6e-01,1.017155698960063637e+00,9.930494485333212218e-01,9.98237755606516
[9.892356910190851504e-01,9.930494485333212218e-01,1.024217735972285714e+00,9.93471351866588
[9.900125185850182641e-01,9.98237755606516175 0e-01,9.934713518665881926e-01,1.018277957748653
])
sidorenko_ratio(H, stretch(W, 1 / (60 * np.max(abs(W - 1)))))
```

```
np.float64(-7.100126511953525e-09)
```