אור טרבלסי מטלה 4 שאלה 4

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import random
from random import randint
from statistics import mean
random.seed(10)
```

מימוש האלגוריתם לחלוקת משימות

1. נמיין את המשימות בסדר יורד
2. יצירת מערך מאותחל ב 0ים כמספר השחקנים עבור חישוב הערך לכל שחקן
3. יצירת מערך מאותחל במערכים ריקים כמספר השחקנים עבור חלוקת המשימות לכל שחקן
4. חלוקת המשימות לפי עומס נוחכי(לפי אלגוריתם הרשימה)
5. חישוב הקירוב בעזרת חלוקת סה"כ העבודות במספר המשתתפים כאשר ידוע שכל משתתף יקבל בדיוק את אותו ערך לעבודות

In [2]:
```
class LPT(object):
    """Implementation of LPT algorithm (Longest Processing Time)."""

    def __init__(self, jobs, processors):
        """Initializing with J Jobs, M Processors."""
        self.jobs = jobs
        self.processors = processors

    def run(self):
        """Run the LPT Algorithm."""
        scheduled_jobs, loads, ratio = self.lpt_algorithm()
        return scheduled_jobs, loads, ratio

    def lpt_algorithm(self):
        """Run the LPT Algorithm.
        Steps:
        1. Sort J jobs in descending order of processing time.
        2. Create a array representing loads on each processor
           a. Initially all loads will be 0
        3. Create an array of array representing the scheduled jobs
           on each processor
           a. Initially no jobs will be scheduled on the processors
        4. Assign each job to a processor based on lowest load
        """
        # Step 1
        sorted_jobs = sorted(self.jobs, reverse=True)

        # Step 2, Step 3
        loads = []
        scheduled_jobs = []
        for processor in range(self.processors):
            loads.append(0)
            scheduled_jobs.append([])

        # Step 4
        for job in sorted_jobs:
            minloadproc = self._minloadproc(loads)
            scheduled_jobs[minloadproc].append(job)
            loads[minloadproc] += job

        ratio = max(loads)/self.optimal()

        return scheduled_jobs, loads, ratio

    def _minloadproc(self, loads):
        """Find the processor with the minimum load.
        Break the tie of processors having same load on
        first come first serve basis.
        """
        minload = min(loads)
        for proc, load in enumerate(loads):
            if load == minload:
                return proc

    def optimal(self):
      return sum(self.jobs)/self.processors
```

יצירת מערך רנדומלי של משימות שהסכום שלהם מוגדר מראש

In [3]:
```
## n = sum of the random list (for each process)
## proc = number of process
## job_n = size of the job list (it can run forever if not -1)

def generate_values(n, proc, job_n = -1):
    values = []
    while len(values) != job_n:
      values = []
      for i in range(proc):
        n_ = n
        while n_ > 0:
            value = randint(1, n_)
            values.append(value)
            n_ -= value
      if job_n == -1:
        return values


    return values
```

יצירת מערך רנדומלי לגמרי

In [4]:
```
def get_random_list(jobs_langth = 0):
  rand_list=[]
  if jobs_langth == 0:
    n=random.randint(4,9)
  else:
    n = jobs_langth
  for i in range(n):
      rand_list.append(random.randint(1,11))
  return rand_list
```

סעיף 1

נניח שלשחקנים יש ערכים זהים לכל משימה

נחלק לכל השחקנים (בצורה רנדומלית) ערך של משימות בכדי לחשב את הערך האופטימלי

נריץ את האלגוריתם 1000 פעמים כאשר מספר השחקנים בין 2-4 בצורה רנדומלית

הערך הוא בין 32^6 לבין 32^7 אבל שווה בין השחקנית

In [5]:
```
indexes = []
vals = []

for i in range(1,1000):
  proc = random.randint(2,4)
  opt = random.randint(32**6,32**7)
  jobs = generate_values(opt,proc)
  lpt = LPT(jobs, proc)
  lpt_run = lpt.run()

  indexes.append(len(jobs))
  vals.append(lpt_run[2])

  print(f"RUN NUMBER {i}:\nNumbr of processes = {proc},\njobs = {jobs},\nLoads split = {lpt_run[1]},\nApproxima
```

```
RUN NUMBER 1:
Numbr of processes = 4,
jobs = [10662521197, 63704872, 885185153, 1986768136, 436648524, 32970345, 27635057, 1164007, 1370432, 168013,
18038, 272981, 128473, 21481, 2494, 4096, 5918, 366, 3444, 569, 1236, 364, 196, 54, 19, 17, 15, 3, 3, 2, 486610
4075, 4219432798, 4163490464, 658554661, 100728525, 5940562, 78239750, 34018, 1976809, 561922, 817694, 1270290,
1124798, 191966, 101195, 7871, 10302, 5462, 2249, 58, 28, 8, 8869082457, 2510606936, 2155717190, 167953536, 120
419592, 221658008, 15997009, 2477987, 2130485, 16663249, 5052698, 10178355, 75534, 559399, 2653, 4905, 6299, 61
31, 2463, 154, 58, 398, 2, 4, 1, 1, 1, 6073430913, 5349303911, 1179842094, 308192273, 1120679278, 23952551, 801
8336, 17919884, 15291925, 634536, 345900, 692503, 91388, 46161, 125865, 25325, 1424, 672, 446, 29, 1, 70, 2, 1
1, 3, 2, 1, 1],
Loads split = [14098595505, 14098595505, 14098595505, 14098595505],
Approximation ratio = 1.0

RUN NUMBER 2:
Numbr of processes = 3,
jobs = [13564503346, 6424241458, 1012540, 2203494, 71223, 136344, 7907, 4933, 256, 9, 22, 32, 22, 2, 1674173254
3, 583087222, 1805756107, 581208388, 187491707, 72298668, 13013089, 4109544, 2943376, 173851, 292931, 57849, 16
17, 6803, 6792, 73, 6, 450, 549, 3, 2, 12, 1, 2, 1, 2, 16318838304, 598287488, 1969844994, 937399837, 13451627
7, 30958782, 1875221, 179952, 135855, 120127, 13872, 5381, 5321, 134, 34, 3, 6],
Loads split = [19992184020, 19992180372, 19992180372],
Approximation ratio = 1.0000001216475545

RUN NUMBER 3:
Numbr of processes = 2,
jobs = [15583985031, 106956691, 3468567770, 62478831, 159064399, 116865169, 66180220, 38830708, 5975817, 155141
9, 20040245, 9237887, 221467, 598707, 170534, 351274, 62565, 2315, 5915, 3283, 942, 13, 12, 6, 2, 4, 4, 1, 4236
012811, 14209915924, 995916953, 186626206, 8765334, 3030278, 444059, 234684, 174083, 24362, 4551, 428, 545, 61
2, 187, 30, 161, 21, 1, 1],
Loads split = [19641151231, 19641151231],
Approximation ratio = 1.0

RUN NUMBER 4:
Numbr of processes = 3,
jobs = [8052757711, 8037333127, 350181200, 53617497, 9418417, 56843636, 2987093, 13822528, 57934, 364910, 6676
8, 94431, 29942, 13900, 3173, 5152, 579, 478, 151, 130, 2, 1, 6032536088, 7041594284, 1498854302, 1560246508, 1
48745868, 259092039, 29981931, 4001434, 2462844, 62903, 2751, 17689, 93, 12, 13, 1, 13608644189, 2589534250, 22
3200974, 151461691, 549359, 2568373, 34221, 1241037, 171792, 170776, 10555, 1211, 6497, 2500, 766, 290, 96, 62,
22, 40, 7, 36, 2, 4, 10],
Loads split = [16577507284, 16577507283, 16577781713],
Approximation ratio = 1.0000110361580496

RUN NUMBER 5:
Numbr of processes = 4,
jobs = [6105963395, 19625979600, 4225732504, 1274396168, 4307883, 105355292, 27021690, 45546031, 38536667, 8002
14, 2192864, 30561, 1659, 168, 7, 85, 2, 1, 10, 2, 1, 18573848793, 2741726693, 3318782869, 4799260614, 16950567
99, 185436711, 44097882, 4179370, 72034377, 17150283, 2930608, 184984, 833439, 230290, 75746, 22787, 978, 2385,
7287, 1447, 242, 157, 29, 29, 5, 9341952865, 16424679791, 1968791978, 834667889, 416750565, 1859280903, 1638088
41, 22196248, 100643635, 102074203, 153011567, 57548165, 6043000, 3155092, 104644, 108712, 173533, 326080, 2341
67, 25975, 92665, 26121, 166773, 1157, 93, 48, 39, 55, 13068594840, 1841548527, 14368116828, 837222202, 1055843
898, 87407628, 45198507, 108884767, 31422162, 5507142, 313089, 1452121, 1478059, 2438584, 370245, 55134, 2509,
8411, 101, 7, 26, 4, 4, 2, 4, 3],
Loads split = [31455832247, 31455962477, 31455832246, 31455832246],
Approximation ratio = 1.0000031050807412

RUN NUMBER 6:
Numbr of processes = 4,
jobs = [18425263238, 3828776139, 742132283, 560292385, 255117988, 32821016, 35737274, 1218961, 51256, 662925, 1
419913, 50648, 30004, 6351, 33443, 3886, 103, 591, 116, 2, 6797974420, 5528991888, 3790725804, 6968230680, 5993
01538, 142605024, 1751814, 5024198, 26857279, 1049147, 580063, 14446781, 2839525, 747370, 198703, 1503345, 3402
17, 327854, 62407, 51593, 2804, 155, 1685, 1043, 1185, 1965, 13, 8, 4, 7, 3, 19520765450, 2557852370, 118350950
5, 33388872, 134592262, 2932255, 66134814, 14854994, 29528492, 95488525, 33457388, 186067056, 12927559, 113093
5, 245423, 7280972, 3096605, 346906, 5992, 10226, 864, 98, 565, 87, 28, 193, 75, 8, 2, 1, 797774646, 1485987140
4, 3062183953, 2860294344, 597185530, 569756538, 543716084, 189058180, 48931236, 247358827, 1785563, 20616532,
37316448, 5273165, 35225378, 2587668, 1181012, 3216627, 54803, 140791, 15780, 37718, 21931, 13268, 8, 532, 530,
14, 9, 1, 1, 1],
Loads split = [23883618532, 23883618515, 23883618514, 23883618527],
Approximation ratio = 1.000000000418697

RUN NUMBER 7:
Numbr of processes = 4,
jobs = [18085928348, 62837833, 629775553, 3695371455, 224899980, 20254809, 3613353, 50213799, 34641525, 30296,
125257, 4908, 3803, 1900, 180, 145, 15, 207, 156, 20, 29, 22, 3, 3, 3, 15758424042, 664725876, 1672334780, 9093
28966, 2232743069, 759000231, 701819674, 75533378, 25476798, 1251980, 5382259, 663910, 774481, 83196, 34053, 72
38, 59986, 3494, 44154, 5481, 4325, 2138, 7, 49, 6, 30, 1, 548048026, 4258221329, 12587183528, 1869457494, 2831
661351, 512770924, 151972818, 44423623, 3032622, 94995, 107923, 71124, 384074, 239073, 28033, 2551, 1456, 519,
1638, 279, 179, 35, 4, 2, 1, 1, 19394420421, 2102117188, 828729134, 423171244, 12798371, 4558094, 35914790, 129
9506, 4514181, 34130, 18729, 86089, 25955, 7467, 1454, 1715, 97, 3721, 367, 681, 88, 94, 50, 14, 8, 6, 5, 3],
Loads split = [22809334709, 22807159900, 22807159900, 22807159899],
Approximation ratio = 1.0000715156171995

RUN NUMBER 8:
Numbr of processes = 3,
jobs = [1040638236, 447453324, 95446770, 91725007, 329493, 21848765, 19692214, 16827913, 4935844, 25186, 21313
3, 93600, 9300, 7525, 350, 119, 429, 110, 12, 16, 30, 6, 13, 1, 813315579, 570469034, 43005737, 3788958, 279303
319, 27739214, 1519256, 72187, 17033, 15828, 175, 864, 30, 105, 62, 14, 1, 845683361, 24901448, 476580262, 2463
01859, 132366543, 11670885, 879988, 313263, 162744, 236329, 43669, 29726, 61911, 8185, 4690, 1916, 414, 78, 12
5],
Loads split = [1739247395, 1739247394, 1739247399],
Approximation ratio = 1.000000001724884

RUN NUMBER 9:
Numbr of processes = 4,
jobs = [19104732257, 3262937349, 10740798872, 197346221, 169345848, 23652232, 6840854, 60815, 9943, 3011, 3841,
1828, 3062, 2471, 42, 136, 2, 18, 10, 1, 1, 1, 31868877231, 1635528519, 644981, 172671, 402591, 25654, 3725, 22
027, 30603, 10296, 1407, 16520, 1006, 73, 36, 291, 1003, 117, 46, 1, 14, 3, 11511457877, 1111875611, 925391480
1, 179850360, 11398046294, 44593292, 2006893, 479564, 2161073, 992743, 155425, 91889, 41974, 55893, 12759, 134
8, 612, 151, 220, 4, 7, 16, 6, 2, 1, 10612730905, 4595002881, 6790038142, 5676786756, 855141364, 2515281079, 18
1228773, 736016154, 346815367, 1037469451, 96733773, 26421753, 8058226, 13049796, 4162209, 3399485, 4819171, 32
4924, 2061133, 14046, 144113, 16821, 16008, 680, 5393, 314, 90, 2, 2, 2, 1, 1],
Loads split = [33531124595, 33497276889, 33497276888, 33497276888],
Approximation ratio = 1.000757654685371
```

ממוצע ערך הקירוב:

In [25]:
```
mean(vals)
```

Out[25]:
```
1.0014054813773718
```

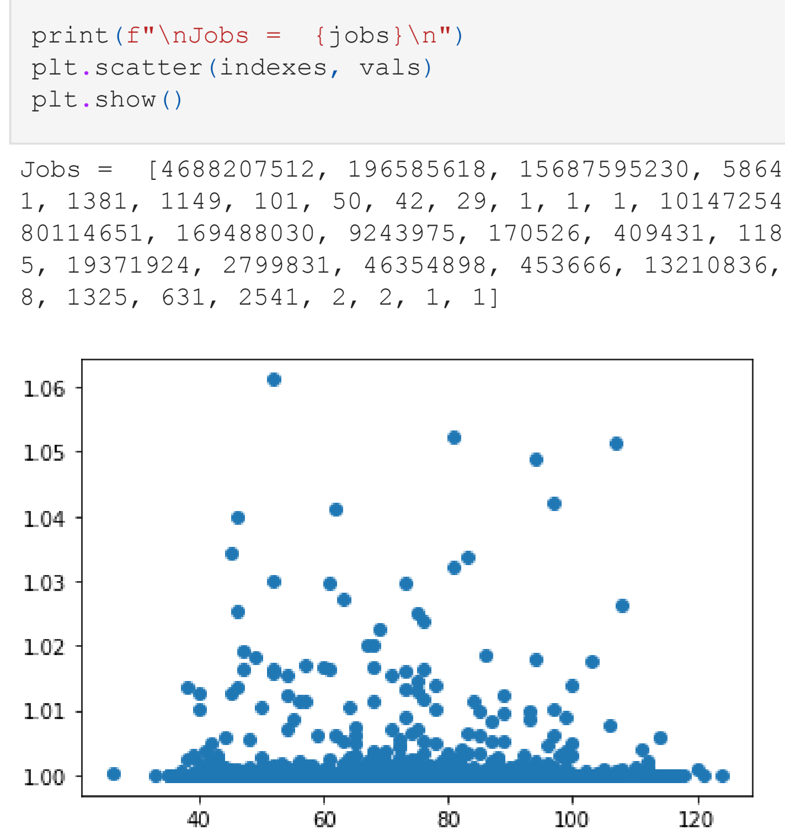
סעיף 2 ניסיון 1

הכנסתי לגרף את כל 1000 הריצות כאשר ציר האיקס הוא מספר המשימות ומיר האוי הוא יחס הקירוב

מכיוון שהדרך בה בחרנו את הערך האופטימלי מכוונת למקרה יחסית קל לא הצלחתי למצוא דפוס בדרך הזאת לכן ניסיתי בעוד דרך

In [15]:
```
print(f"\nJobs =  {jobs}\n")
plt.scatter(indexes, vals)
plt.show()
```

```
Jobs =  [4688207512, 196585618, 15687595230, 5864, 1181623, 1296869, 2891069, 1012979, 26582, 31398, 10664, 105
1, 1381, 1149, 101, 50, 42, 29, 1, 1, 1, 10147254881, 5333238255, 2673409438, 544996250, 888263992, 732245549,
80114651, 169488030, 9243975, 170526, 409431, 11834, 272, 1318, 795, 4, 8, 5, 10589952345, 9788946034, 11017044
5, 19371924, 2799831, 46354898, 453666, 13210836, 4635423, 1003980, 1255985, 202998, 331872, 81125, 13991, 5935
8, 1325, 631, 2541, 2, 2, 1, 1]
```



סעיף 2 ניסיון 2

הגרף דומה לפונקציה 1 חלקי איקס!

(עם ערכים חיובים כי מספר משימות לא יכול להיות שלילי)

הרצתי שוב אך הפעם בצורה רנדומלית לגמרי בלי לבחור ערך אופטימלי מראש

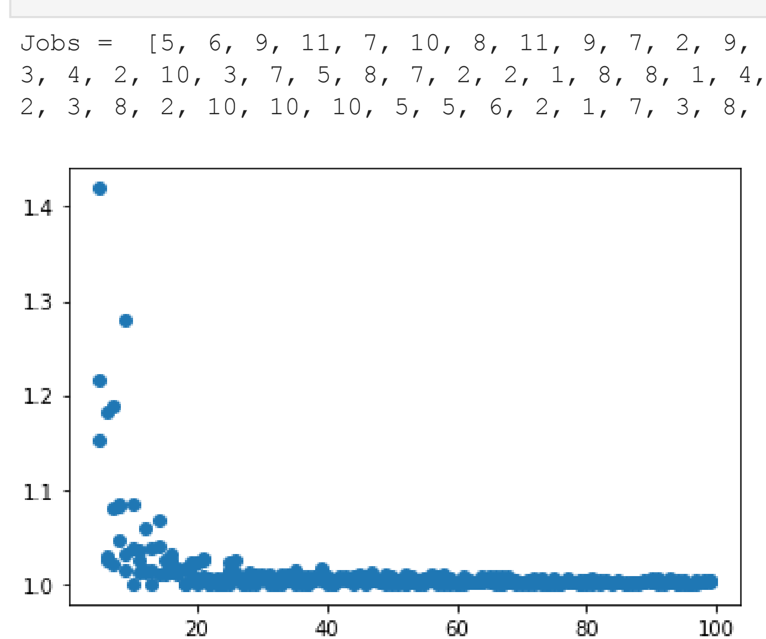
יחס הקירוב פה הוא לא מדויק מכיון שחילקתי אותו במספר המשתתפים בעוד שאיני יכול להבטיח שכולן יקבלו את אותו הערך.

אבל ניתן להעריך הם יקבלו ערך יחסית קרוב לכן במספרים גדולים יש חשיבות קטנה להבדל

In [18]:
```
from itertools import product

indexes = []
vals = []
for i ,j in product(range(5,100), [2,3,4]):
  jobs = get_random_list(i)
  lpt = LPT(jobs, j)
  indexes.append(i)
  vals.append(lpt.run()[2])
print(f"\nJobs =  {jobs}\n")
plt.scatter(indexes, vals)
plt.show()
```

```
Jobs =  [5, 6, 9, 11, 7, 10, 8, 11, 9, 7, 2, 9, 6, 9, 11, 2, 6, 5, 1, 5, 3, 1, 9, 2, 11, 2, 10, 4, 11, 11, 6,
3, 4, 2, 10, 3, 7, 5, 8, 7, 2, 2, 1, 8, 8, 1, 4, 1, 2, 4, 8, 2, 11, 6, 1, 8, 8, 10, 9, 10, 2, 4, 11, 8, 2, 11,
2, 3, 8, 2, 10, 10, 10, 5, 5, 6, 2, 1, 7, 3, 8, 11, 5, 3, 10, 8, 5, 4, 5, 4, 2, 1, 8, 2, 7, 11, 10, 9, 4]
```



In [ ]:
```
indexes = []
vals = []
for i ,j in product(range(5,70), [2,3,4]):
  jobs = get_random_list(i)
  lpt = LPT(jobs, j)
  indexes.append(i)
  vals.append(lpt.run()[2])
print(f"\nJobs =  {jobs}\n")
plt.scatter(indexes, vals)
plt.show()
```

```
Jobs =  [6, 7, 8, 3, 7, 2, 5, 11, 11, 3, 3, 2, 6, 11, 1, 5, 7, 10, 7, 7, 4, 9, 7, 8, 9, 1, 7, 11, 8, 9, 10, 2,
1, 6, 1, 1, 3, 2, 4, 6, 3, 5, 8, 4, 6, 3, 3, 10, 8, 7, 8, 5, 2, 7, 3, 7, 3, 9, 1, 7, 9, 3, 9, 9, 10, 8, 11, 7,
1]
```