[1]

```
%matplotlib inline
from os import listdir
from os.path import isfile, join
import matplotlib.pyplot as plt
from collections import Counter
import math
from collections import Counter
import numpy as np
from scipy.special import comb
import itertools as it
%load_ext line_profiler
from imp import reload
import itertools as it
import pandas as pd
import seaborn as sns
import sys
sys.path.insert(0, '../mallows kendall')
import mallows_kendall as mk
import cego_lop as cego

from IPython.core.display import display, HTML
display(HTML("<style>.container { width:90% !important; }
</style>"))
```

References

- http://www.spotseven.de/wp-content/papercite-data/pdf/zaef14c.pdf
- https://dl.acm.org/doi/pdf/10.1145/2576768.2598282
- https://pubsonline.informs.org/doi/10.1287/ijoc.1120.0506
- https://link.springer.com/article/10.1007/s11721-015-0106-x

  falta encontrar donde habia uno con el LOP

LOP instance generator

The instances M follow this distribution $M_\phi[i,j]$

$$M_\phi[i,j] = h(j-i+1,\phi) - h(j-i,\phi),$$

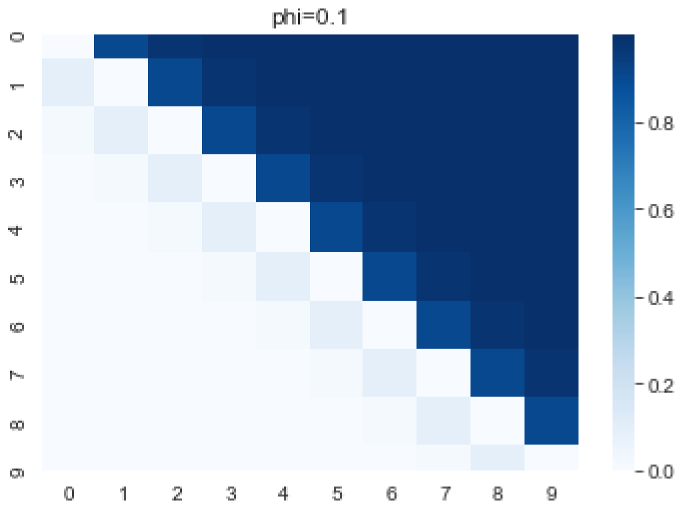
where

$$h(k,\phi) = k/(1-\phi^k).$$

Taking different values of ϕ we controll the uniformity of M:

[49]

```
def h(k,phi):
  if (1-phi**k) == 0 :
    return 0
  return k/(1-phi**k)
  #h(k,\phi)=k/(1-\phi^k)
def mij(i,j,phi):
  return h(j-i+1,phi) - h(j-i,phi)
      #h(j-i+1,\phi) - h(j-i,\phi)
n = 10
for phi in [0.1,0.5,0.7,0.9,0.999]:
  M = np.zeros((n,n))
  for i in range(n):
    for j in range(i+1,n):
      M[i,j] = mij(i,j,phi)
      M[j,i] = 1-M[i,j]
  g = sns.heatmap(M, cmap="Blues")
  g.set_title("phi="+str(phi))
  plt.show()
```



Do similar permutations have similar fitness?

In this experiment we analyse wether similar permutations in terms of Kendall distance have similar fitness funtion evaluation in the LOP instances. The process is as follows:

do 100 times:

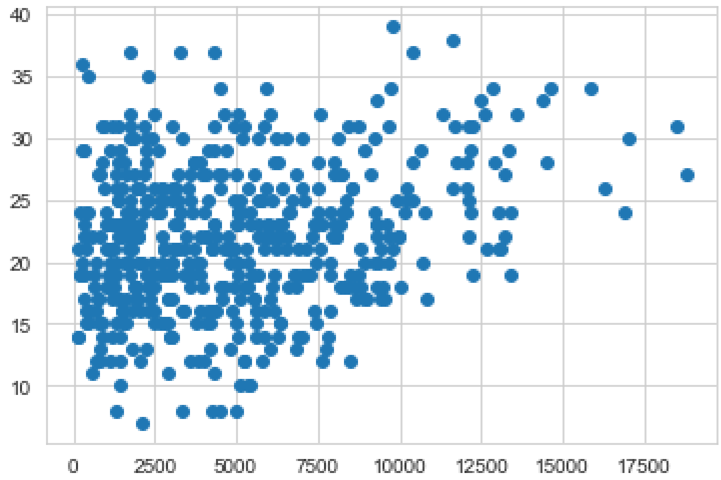
1. a,b = generate two u.a.r. permutations
2. let $x = |f(a) - f(b)|$
3. let $y = d(a, b)$

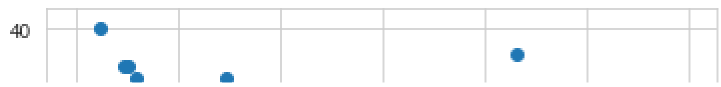
4. draw a point in (x, y)

We see that:

- permutations that are very different in fitness are distant
- permutations that are similar can have similar or differnt fitness values

[57]
```
n = 10
for phi_instance in [0.5,0.7,0.9]:
  instance = cego.synthetic_LOP(n,1000,phi_instance)
  xs, ys = [],[]
  for repes in range(500):
    a,b = np.random.permutation(range(n)),np.random.permutation(range(n))
    #cego.get_fitness(a, instance,"LOP"), cego.get_fitness(b, instance,"LOP"), mk.kendallTau(a,b)
    xs.append(abs(cego.get_fitness(a, instance,"LOP") - cego.get_fitness(b, instance,"LOP")))
    ys.append(mk.kendallTau(a,b))
  plt.scatter(xs,ys)
  plt.show()
```





running experimtns

How to run one experiment with a particular parameter configuration

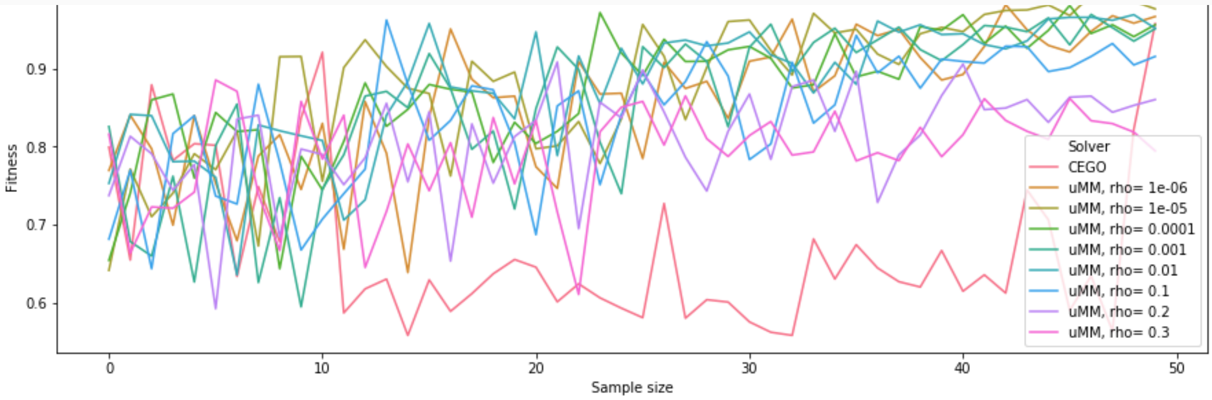
[28]
```
reload(cego)
n = 10
m_max = 50
```

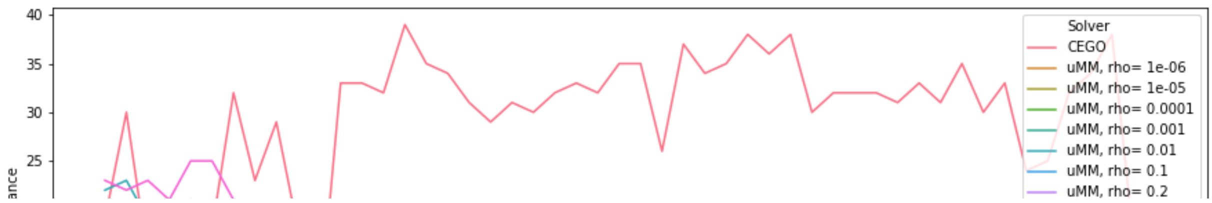
```
repe = 0
#m_ini = 10
phi_instance = 0.8
budgetGA = 25

cego.run_and_save(n,repe,phi_instance, budgetGA,m_max=m_max)
```

[3]

```
df = pd.read_pickle('pickles/pickLocal.pkl')#pick275670.pkl
color_variable = 'Solver'
y_variables = ['Fitness','Distance']
palette = sns.color_palette("husl",
len(df[color_variable].drop_duplicates()))
for y_variable in y_variables:
    plt.figure(figsize=(15,5))
    sns.lineplot(x='Sample
size',y=y_variable,hue='Solver',data=df, palette=palette)
    plt.show()
```





Plot the results

[13]

```
df = pd.concat([pd.read_pickle("pickles/"+f) for f in
listdir("pickles") if (f.endswith(".pkl")and "Local" not in f)]
)
df
```

0	0.673686	LOP	CEGO	0	0	100.0	19

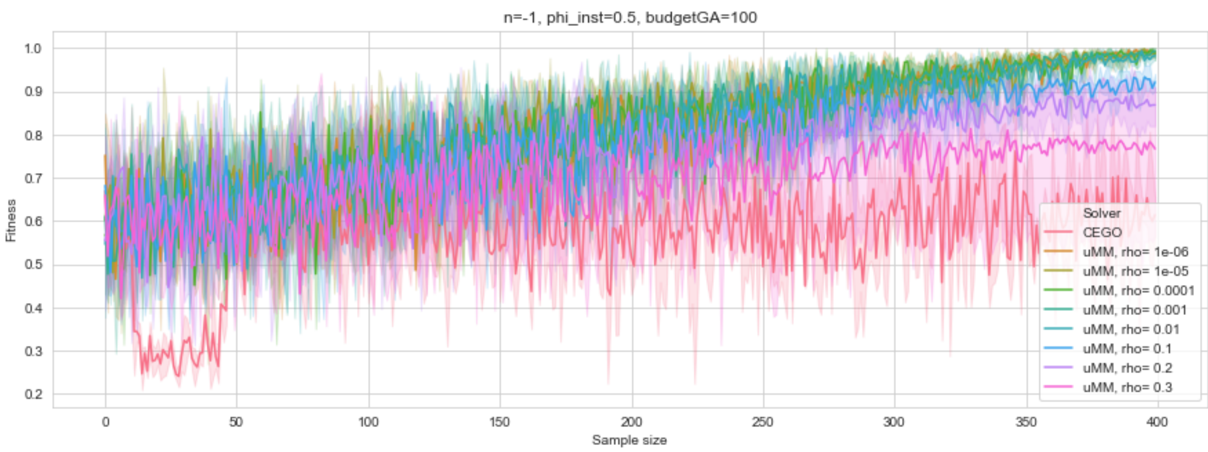
	Fitness	Problem	Solver	Sample size	repe	budgetGA	Dis
1	0.429744	LOP	CEGO	1	0	100.0	30
2	0.742816	LOP	CEGO	2	0	100.0	15
3	0.649583	LOP	CEGO	3	0	100.0	20
4	0.636075	LOP	CEGO	4	0	100.0	21
...	...	...	...	...	...	...	...
395	0.920394	LOP	uMM, rho=	395	3	NaN	8

[20]

```
df['n']=-1
df['budgetGA'] = 100
```

[24]

```
sns.set_style("whitegrid")
color_variable = 'Solver'
y_variables = ['Fitness','Distance']
palette = sns.color_palette("husl",
len(df[color_variable].drop_duplicates()))
for phi_i in df.phi_instance.drop_duplicates().values:
  for n in df.n.drop_duplicates().values:
    for budgetGA in df.budgetGA.drop_duplicates().values:
      for y_variable in y_variables:
          plt.figure(figsize=(15,5))
          aux = df[(df.phi_instance==phi_i) & (df.n==n) &
(df.budgetGA==budgetGA) ] #& (df.repe==0)
          g = sns.lineplot(x='Sample
size',y=y_variable,hue='Solver',data=aux, palette=palette)
          g.set_title('n='+str(n)+', phi_inst='+str(phi_i)+',
budgetGA='+str(budgetGA))
          plt.show()
```



[72]

```
true_sol = list(range(n))
instance = cego.synthetic_LOP(n,m_inst,phi_instance)
print("best* fitness",cego.get_fitness(true_sol,
instance,problem),"worst*
fitness",cego.get_fitness(true_sol[::-1], instance,problem),"
(*distributed according to)")

dfcego = cego.runCEGO(n,instance, m_ini = m_ini, m =
ms,repe=repe, best_known_sol=true_sol)
dfuMM = cego.solve_one_umm("LOP",instance,ms, rho, repe,
 m_ini, phi_ini,true_sol)
df = pd.concat([dfuMM,dfcego],sort=False)
df['best'] = cego.get_fitness(true_sol, instance,problem)
df['worst'] = cego.get_fitness(true_sol[::-1],
instance,problem)
```

	Problem	Solver	repe	Sample size	rho	Fitness	phi_estin
0	LOP	uMM, \rho= 0.001	0	0	0.001	0.719931	0.718161
1	LOP	uMM, \rho= 0.001	0	1	0.001	0.651240	0.776211
2	LOP	uMM, \rho= 0.001	0	2	0.001	0.626840	0.789803

TODO

- meter más problemas: **PFSP**, TSP, ...
- comparar con otras alternativas: LS?
- el símil con la optimización bayesiana no está claro, cómo se traslada aquí la función de utilidad?
- demostración de convergencia rápida
- escribir draft para tener el modelo claro

[ ]

Empty markdown cell, double click me to add content.