```
import pandas as pd
import numpy as np
import random
import matplotlib.pyplot as plt
import sys
import time
import warnings
warnings.filterwarnings("ignore")
```

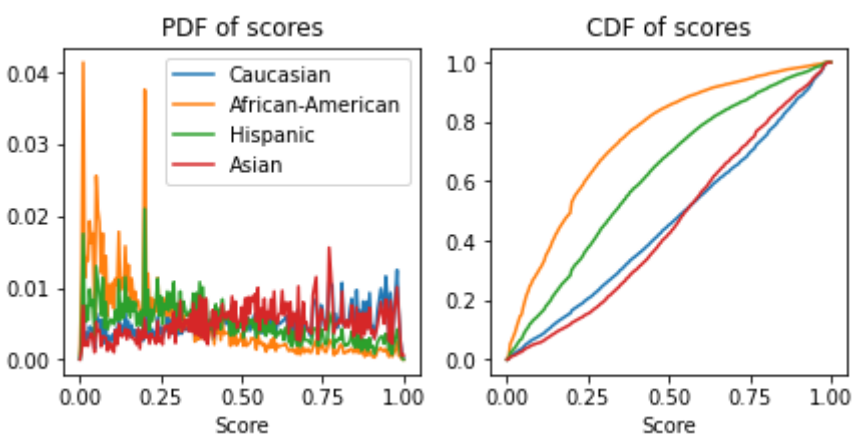
## load FICO dataset

In [4]:
```
DATA_DIR = 'data/'
file_name = ['transrisk_performance_by_race_ssa.csv','transrisk_cdf_by_race_ssa.csv','totals.csv']
df_repay = pd.read_csv(DATA_DIR + file_name[0])
df_cdf = pd.read_csv(DATA_DIR + file_name[1])
df_dm_ratio = pd.read_csv(DATA_DIR + file_name[2])

df_repay = df_repay.rename(columns={"Non- Hispanic white": "Caucasian", "Black": "African-American",
"Hispanic":"Hispanic", "Asian":"Asian"})
df_cdf = df_cdf.rename(columns={"Non- Hispanic white": "Caucasian", "Black": "African-American", "His
panic":"Hispanic", "Asian":"Asian"})
df_dm_ratio = df_dm_ratio.rename(columns={"Non- Hispanic white": "Caucasian", "Black": "African-Ameri
can", "Hispanic":"Hispanic", "Asian":"Asian"})

df_cdf.iloc[:,1:] = df_cdf.iloc[:,1:]/100.0
df_repay.iloc[:,1:] = (100-df_repay.iloc[:,1:])/100.0
```

## obtain PDF from CDF ¶

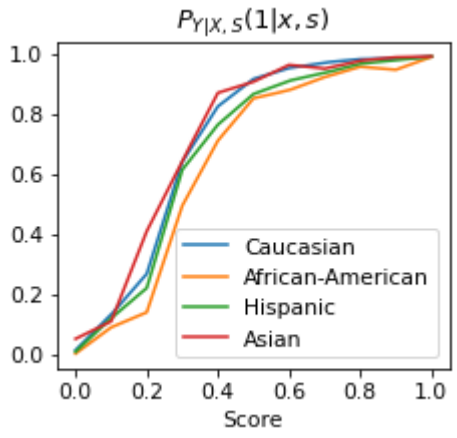
In [5]:
```
nrow, ncol = df_cdf.shape
df_pdf = df_cdf.copy()
for i in range(nrow-1):
    indx = i + 1
    df_pdf.iloc[i+1,1:] = df_cdf.iloc[i+1,1:]- df_cdf.iloc[i,1:]
```

In [20]:
```
fig = plt.figure(figsize=(7,3))
plt.subplot(1,2,1)
plt.plot(df_pdf["Score"]/100,df_pdf['Caucasian'],label='Caucasian')
plt.plot(df_pdf["Score"]/100,df_pdf['African-American'],label='African-American')
plt.plot(df_pdf["Score"]/100,df_pdf['Hispanic'],label='Hispanic')
plt.plot(df_pdf["Score"]/100,df_pdf['Asian'],label='Asian')
plt.xlabel('Score')
plt.title("PDF of scores")
plt.legend()
plt.subplot(1,2,2)
plt.plot(df_cdf["Score"]/100,df_cdf['Caucasian'],label='Caucasian')
plt.plot(df_cdf["Score"]/100,df_cdf['African-American'],label='African-American')
plt.plot(df_cdf["Score"]/100,df_cdf['Hispanic'],label='Hispanic')
plt.plot(df_cdf["Score"]/100,df_cdf['Asian'],label='Asian')
plt.xlabel('Score')
plt.title("CDF of scores")
#fig.savefig('pdf_cdf.eps', bbox_inches='tight')
plt.show()
```




**repayment probability** $P_{Y|X,S}(1|x, s)$PY|X,S(1|x,s)

In [21]:
```
import matplotlib
matplotlib.rcParams.update({'font.size': 10.7})

fig = plt.figure(figsize=(3.5,3))
plt.plot(df_repay["Score"]/100,df_repay.iloc[:,1],label='Caucasian')
plt.plot(df_repay["Score"]/100,df_repay.iloc[:,2],label='African-American')
plt.plot(df_repay["Score"]/100,df_repay.iloc[:,3],label='Hispanic')
plt.plot(df_repay["Score"]/100,df_repay.iloc[:,4],label='Asian')
plt.title(r"$P_{Y|X,S}(1|x,s)$")
plt.xlabel('Score')
plt.legend()
#fig.savefig('repay_prob.eps', bbox_inches='tight')
plt.show()
```



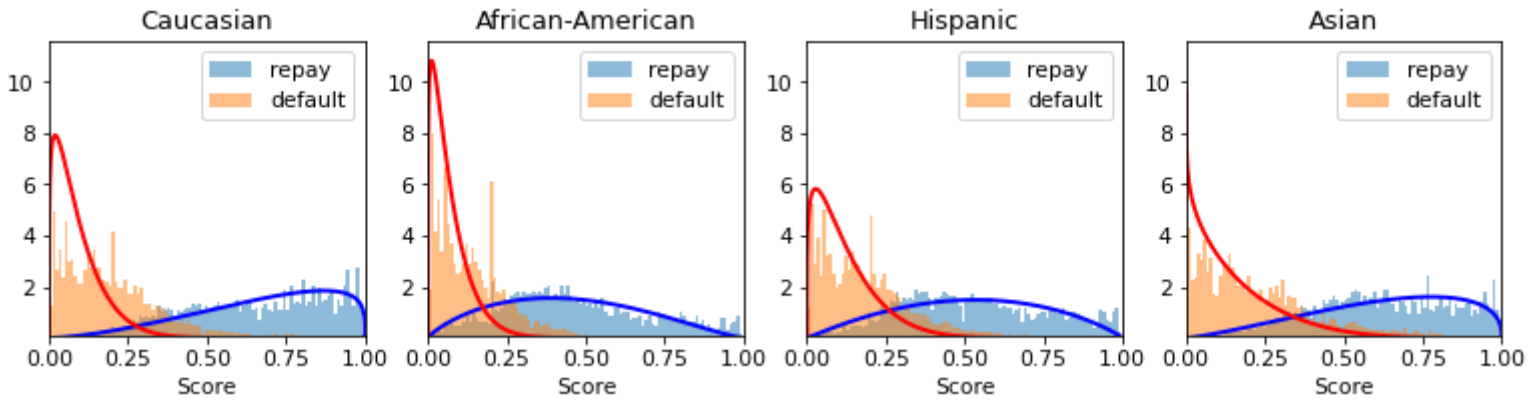

## Create a simulator

**Step 1.** Generate simulated data from the joint distributions given above,
$P(X = x, Y = y \mid S = s) = P(Y = y \mid X = x, S = s)P(X = x \mid S = s)$ P(X=x,Y=y|S=s)=P(Y=y|X=x,S=s)P(X=x|S=s).

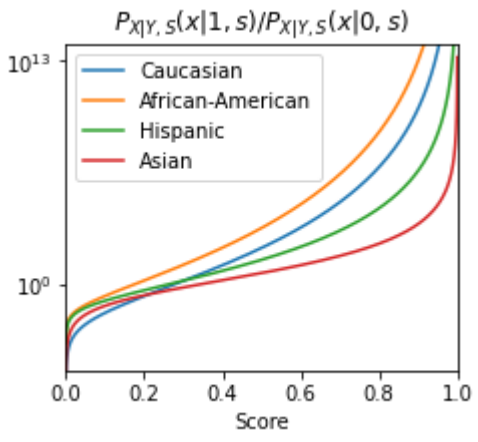
In [7]:
```
np.random.seed(777)
# Step 1
NUM_SAMPLES = 100000
elements = df_pdf["Score"]
probabilities_c = df_pdf["Caucasian"]
probabilities_aa = df_pdf["African-American"]
probabilities_h = df_pdf["Hispanic"]
probabilities_a = df_pdf["Asian"]

scores_c = np.random.choice(elements, NUM_SAMPLES, p=probabilities_c)
scores_aa = np.random.choice(elements, NUM_SAMPLES, p=probabilities_aa)
scores_h = np.random.choice(elements, NUM_SAMPLES, p=probabilities_h)
scores_a = np.random.choice(elements, NUM_SAMPLES, p=probabilities_a)
prob_repay_c = [df_repay.loc[df_repay["Score"]==x]["Caucasian"].values[0] for x in scores_c]
prob_repay_aa = [df_repay.loc[df_repay["Score"]==x]["African-American"].values[0] for x in scores_aa]
prob_repay_h = [df_repay.loc[df_repay["Score"]==x]["Hispanic"].values[0] for x in scores_h]
prob_repay_a = [df_repay.loc[df_repay["Score"]==x]["Asian"].values[0] for x in scores_a]

repay_c = np.random.binomial(1, prob_repay_c, NUM_SAMPLES)
repay_aa = np.random.binomial(1, prob_repay_aa, NUM_SAMPLES)
repay_h = np.random.binomial(1, prob_repay_h, NUM_SAMPLES)
repay_a = np.random.binomial(1, prob_repay_a, NUM_SAMPLES)

scores_c = scores_c/100.0
scores_aa = scores_aa/100.0
scores_h = scores_h/100.0
scores_a = scores_a/100.0
```

**Step 2.** Fit beta distributions to the generated data

In [22]:
```
from scipy.stats import beta
import scipy.stats as ss
# Step 2
a_c1,b_c1,loc_c1,scale_c1 = beta.fit(scores_c[repay_c==1])
a_c0,b_c0,loc_c0,scale_c0 = beta.fit(scores_c[repay_c==0])
a_aa0,b_aa0,loc_aa0,scale_aa0 = beta.fit(scores_aa[repay_aa==0])
a_aa1,b_aa1,loc_aa1,scale_aa1 = beta.fit(scores_aa[repay_aa==1])

a_h0,b_h0,loc_h0,scale_h0 = beta.fit(scores_h[repay_h==0])
a_h1,b_h1,loc_h1,scale_h1 = beta.fit(scores_h[repay_h==1])
a_a0,b_a0,loc_a0,scale_a0 = beta.fit(scores_a[repay_a==0])
a_a1,b_a1,loc_a1,scale_a1 = beta.fit(scores_a[repay_a==1])


def plot_beta(x_range, a, b, mu=0, sigma=1, cdf=False, **kwargs):
    '''
    Plots the f distribution function for a given x range, a and b
    If mu and sigma are not provided, standard beta is plotted
    If cdf=True cumulative distribution is plotted
    Passes any keyword arguments to matplotlib plot function
    '''
    x = x_range
    if cdf:
        y = ss.beta.cdf(x, a, b, mu, sigma)
    else:
        y = ss.beta.pdf(x, a, b, mu, sigma)
    plt.plot(x, y, **kwargs)

x = np.linspace(0, 1, 5000)
fig = plt.figure(figsize=(13, 2.7))
plt.subplot(1,4,1)
plt.ylim(0.1, 11.6)
plt.xlim(0, 1)
plot_beta(x, a_c1,b_c1, 0, 1, color='blue', lw=2, ls='-')
plot_beta(x, a_c0,b_c0, 0, 1, color='red', lw=2, ls='-')
plt.hist(scores_c[repay_c==1], density=True, bins=100, label='repay',alpha=0.5)
plt.hist(scores_c[repay_c==0], density=True, bins=100, label='default',alpha=0.5)
plt.title("Caucasian")
plt.xlabel('Score')
plt.legend()
plt.subplot(1,4,2)
plt.ylim(0.1, 11.6)
plt.xlim(0, 1)
plot_beta(x, a_aa1,b_aa1, 0, 1, color='blue', lw=2, ls='-')
plot_beta(x,a_aa0,b_aa0, 0, 1, color='red', lw=2, ls='-')
plt.hist(scores_aa[repay_aa==1], density=True, bins=100, label='repay',alpha=0.5)
plt.hist(scores_aa[repay_aa==0], density=True, bins=100, label='default',alpha=0.5)
plt.title("African-American")
plt.xlabel('Score')
plt.legend()
plt.subplot(1,4,3)
plt.ylim(0.1, 11.6)
plt.xlim(0, 1)
plot_beta(x, a_h1,b_h1, 0, 1, color='blue', lw=2, ls='-')
plot_beta(x,a_h0,b_h0, 0, 1, color='red', lw=2, ls='-')
plt.hist(scores_h[repay_h==1], density=True, bins=100, label='repay',alpha=0.5)
plt.hist(scores_h[repay_h==0], density=True, bins=100, label='default',alpha=0.5)
plt.title("Hispanic")
plt.xlabel('Score')
plt.legend()
plt.subplot(1,4,4)
plt.ylim(0.1, 11.6)
plt.xlim(0, 1)
plot_beta(x, a_a1,b_a1, 0, 1, color='blue', lw=2, ls='-')
plot_beta(x,a_a0,b_a0, 0, 1, color='red', lw=2, ls='-')
plt.hist(scores_a[repay_a==1], density=True, bins=100, label='repay',alpha=0.5)
plt.hist(scores_a[repay_a==0], density=True, bins=100, label='default',alpha=0.5)
plt.title("Asian")
plt.xlabel('Score')
plt.legend()
fig.savefig('score_pdf.pdf', bbox_inches='tight')
plt.show()
```



In [17]:
```
from matplotlib.ticker import LogLocator
import warnings
warnings.filterwarnings("ignore")
def plot_monotone(x_range, a0, b0, a1, b1, mu=0, sigma=1, **kwargs):
    '''
    Plots the f distribution function for a given x range, a and b
    If mu and sigma are not provided, standard beta is plotted
    If cdf=True cumulative distribution is plotted
    Passes any keyword arguments to matplotlib plot function
    '''
    x = x_range
    y0 = ss.beta.pdf(x, a0, b0, mu, sigma)
    y1 = ss.beta.pdf(x, a1, b1, mu, sigma)
    ratio = y1/y0
    plt.plot(x, ratio, **kwargs)
    plt.yscale('log')

fig = plt.figure(figsize=(3.5, 3))
ax = fig.add_subplot(111)
plt.xlim(0, 1)
plt.ylim(0.00001, 10**14)
plot_monotone(x,a_c0,b_c0,a_c1,b_c1, 0, 1, lw=1.5, ls='-',label='Caucasian')
plot_monotone(x,a_aa0,b_aa0,a_aa1,b_aa1, 0, 1, lw=1.5, ls='-',label='African-American')
plot_monotone(x,a_h0,b_h0,a_h1,b_h1, 0, 1,  lw=1.5, ls='-',label='Hispanic')
plot_monotone(x,a_a0,b_a0,a_a1,b_a1, 0, 1, lw=1.5, ls='-',label='Asian')

ax.yaxis.set_major_locator(LogLocator(base=10**13))

plt.xlabel('Score')
plt.title(r'$P_{X|Y,S}(x|1,s)/P_{X|Y,S}(x|0,s)$')
plt.legend()
#fig.savefig('monotone.pdf', bbox_inches='tight')
plt.show()
```




**demographic parameters**

In [9]:
```
# relative sizes
total = df_dm_ratio.iloc[0]['Caucasian']+df_dm_ratio.iloc[0]['African-American']+df_dm_ratio.iloc[0][
'Hispanic']+df_dm_ratio.iloc[0]['Asian']
P_c = df_dm_ratio.iloc[0]['Caucasian']/total
P_aa = df_dm_ratio.iloc[0]['African-American']/total
P_h = df_dm_ratio.iloc[0]['Hispanic']/total
P_a = df_dm_ratio.iloc[0]['Asian']/total
print(P_c,P_aa,P_h,P_a)
```

```
0.7651094244658052 0.10499462788786937 0.08447143587651611 0.04542451176980931
```

In [10]:
```
# repayment rates
alpha_aa, alpha_c,alpha_h, alpha_asian = np.sum(repay_aa)/NUM_SAMPLES, np.sum(repay_c)/NUM_SAMPLES,np
.sum(repay_h)/NUM_SAMPLES, np.sum(repay_a)/NUM_SAMPLES
print(alpha_aa, alpha_c,alpha_h, alpha_asian)
```

```
0.33849 0.75972 0.56977 0.80467
```

# Verification of results

## 1. Impact of being strategic

In [11]:
```
# scaled utility function & strategic optimal threshold
def opt_strategic(u_cost,u_benefit,alpha,a0,b0,a1,b1,cost_a,cost_b,cost_type):
    theta = np.linspace(0,1,10000)
    Delta = beta.cdf(theta, a0, b0, 0, 1) - beta.cdf(theta, a1, b1, 0, 1)
    rho = u_cost*(1-alpha)/(u_benefit*alpha)
    if cost_type == "beta":
        U = beta.cdf(theta,a0,b0)*(1-beta.cdf(Delta,cost_a,cost_b,0,1))*rho - beta.cdf(theta,a1,b1)*(
1-rho*beta.cdf(Delta,cost_a,cost_b,0,1))
    if cost_type == "uniform":
        U = beta.cdf(theta,a0,b0)*(1-uniform.cdf(Delta,cost_a,cost_b-cost_a))*rho - beta.cdf(theta,a1
,b1)*(1-rho*uniform.cdf(Delta,cost_a,cost_b-cost_a))
    return  U, theta[int(np.argmax(U))]

# scaled utility function & non-strategic optimal threshold
def opt_non_strategic(u_cost,u_benefit,alpha,a0,b0,a1,b1):
    theta = np.linspace(0,1,10000)
    U = beta.cdf(theta,a0,b0)*(1-alpha)*u_cost -  beta.cdf(theta,a1,b1)*alpha*u_benefit
    return  U, theta[int(np.argmax(U))]

# one group: measure for calculating unfairness
def fair_measure(threshold,a0,b0,a1,b1,alpha,fair_type):
    if fair_type == "eqopt":
        return beta.cdf(threshold, a1, b1, 0, 1)
    if fair_type == "dp":
        return beta.cdf(threshold, a1, b1, 0, 1)*alpha + beta.cdf(threshold, a0, b0, 0, 1)*(1-alpha)


def check_exacerbate(x_star_a,a0_a,b0_a,a1_a,b1_a,alpha_a,x_star_b,a0_b,b0_b,a1_b,b1_b,alpha_b):
    I_eqopt,I_dp = 0,0
    if beta.cdf(x_star_a,a1_a,b1_a) <= beta.cdf(x_star_b,a1_b,b1_b):
        I_eqopt = 1
    if beta.cdf(x_star_a,a1_a,b1_a)*alpha_a + beta.cdf(x_star_a,a0_a,b0_a)*(1-alpha_a) <= beta.cdf(x_
star_b,a1_b,b1_b)*alpha_b + beta.cdf(x_star_b,a0_b,b0_b)*(1-alpha_b):
        I_dp = 1
    return I_eqopt,I_dp



def find_x_star(a0,b0,a1,b1):
    theta = np.linspace(0,1,10000)
    diff = np.abs(beta.pdf(theta, a0, b0, 0, 1) - beta.pdf(theta, a1, b1, 0, 1))
    return theta[int(np.argmin(diff[10:-10]))+10]
```

In [12]:
```
print(a_c0,b_c0,a_c1,b_c1)
print(a_aa0,b_aa0,a_aa1,b_aa1)
print(a_h0,b_h0,a_h1,b_h1)
print(a_a0,b_a0,a_a1,b_a1)
```

```
1.2270282886666484 12.340422955024335 2.573294489243736 1.2350690397024984
1.1773023697277172 15.987695806847078 1.8404787762920347 2.317591652871121
1.2305911110521812 9.019826285918608 2.0344703251184457 1.9023299221561116
0.8910805045005428 4.944756596979116 2.3108301137946197 1.3779368618213224
```

**(1)**. $\alpha_b < \frac{u_-}{u_+ + u_-} < \alpha_a$ɑb<u−u++u−<ɑa, strategic policies exacerbate unfairness

In [24]:
```
def opt_threshold(u_cost,u_benefit,alpha_c,alpha_aa,alpha_h,alpha_asian,cost_c_a,cost_c_b,cost_aa_a,c
ost_aa_b,cost_h_a,cost_h_b,cost_asian_a,cost_asian_b,cost_type):
    # non-strategic & strategic policies of each group
    _,opt_strategic_c = opt_strategic(u_cost,u_benefit,alpha_c,a_c0,b_c0,a_c1,b_c1,cost_c_a,cost_c_b,
cost_type)
    _,opt_non_strategic_c = opt_non_strategic(u_cost,u_benefit,alpha_c,a_c0,b_c0,a_c1,b_c1)

    _,opt_strategic_aa = opt_strategic(u_cost,u_benefit,alpha_aa,a_aa0,b_aa0,a_aa1,b_aa1,cost_aa_a,co
st_aa_b,cost_type)
    _,opt_non_strategic_aa = opt_non_strategic(u_cost,u_benefit,alpha_aa,a_aa0,b_aa0,a_aa1,b_aa1)

    _,opt_strategic_h = opt_strategic(u_cost,u_benefit,alpha_h,a_h0,b_h0,a_h1,b_h1,cost_h_a,cost_h_b,
cost_type)
    _,opt_non_strategic_h = opt_non_strategic(u_cost,u_benefit,alpha_h,a_h0,b_h0,a_h1,b_h1)

    _,opt_strategic_a = opt_strategic(u_cost,u_benefit,alpha_asian,a_a0,b_a0,a_a1,b_a1,cost_asian_a,c
ost_asian_b,cost_type)
    _,opt_non_strategic_a = opt_non_strategic(u_cost,u_benefit,alpha_asian,a_a0,b_a0,a_a1,b_a1)

    return opt_strategic_c,opt_non_strategic_c,opt_strategic_aa,opt_non_strategic_aa,opt_strategic_h,
opt_non_strategic_h,opt_strategic_a,opt_non_strategic_a
```

In [26]:
```
u_cost,u_benefit = 1,1

'''
cost_type = "beta"
cost_c_a,cost_c_b = 10,2
cost_aa_a,cost_aa_b = 10,6 # 10,6
cost_h_a,cost_h_b = 10,2
cost_asian_a,cost_asian_b = 10,2
'''
cost_type = "uniform"
cost_c_a,cost_c_b = 0,1
cost_aa_a,cost_aa_b = 0,1# 0,0.5
cost_h_a,cost_h_b = 0,1
cost_a_a,cost_a_b = 0,1


opt_strategic_c,opt_non_strategic_c,opt_strategic_aa,opt_non_strategic_aa,opt_strategic_h,opt_non_str
ategic_h,opt_strategic_a,opt_non_strategic_a = opt_threshold(u_cost,u_benefit,alpha_c,alpha_aa,alpha_
h,alpha_asian,cost_c_a,cost_c_b,cost_aa_a,cost_aa_b,cost_h_a,cost_h_b,cost_a_a,cost_a_b,cost_type)

# unfairness induced by each policy
fair_type = "dp"
fair_strategic_c = fair_measure(opt_strategic_c,a_c0,b_c0,a_c1,b_c1,alpha_c,fair_type)
fair_non_strategic_c = fair_measure(opt_non_strategic_c,a_c0,b_c0,a_c1,b_c1,alpha_c,fair_type)

fair_strategic_aa = fair_measure(opt_strategic_aa,a_aa0,b_aa0,a_aa1,b_aa1,alpha_aa,fair_type)
fair_non_strategic_aa = fair_measure(opt_non_strategic_aa,a_aa0,b_aa0,a_aa1,b_aa1,alpha_aa,fair_type)

fair_strategic_h = fair_measure(opt_strategic_h,a_h0,b_h0,a_h1,b_h1,alpha_h,fair_type)
fair_non_strategic_h = fair_measure(opt_non_strategic_h,a_h0,b_h0,a_h1,b_h1,alpha_h,fair_type)

fair_strategic_a = fair_measure(opt_strategic_a,a_a0,b_a0,a_a1,b_a1,alpha_asian,fair_type)
fair_non_strategic_a = fair_measure(opt_non_strategic_a,a_a0,b_a0,a_a1,b_a1,alpha_asian,fair_type)

unfair_strategic_c_aa = fair_strategic_c - fair_strategic_aa
unfair_non_strategic_c_aa = fair_non_strategic_c - fair_non_strategic_aa

unfair_strategic_a_aa = fair_strategic_a - fair_strategic_aa
unfair_non_strategic_a_aa = fair_non_strategic_a - fair_non_strategic_aa

unfair_strategic_h_aa = fair_strategic_h - fair_strategic_aa
unfair_non_strategic_h_aa = fair_non_strategic_h - fair_non_strategic_aa

print(-unfair_strategic_c_aa,-unfair_non_strategic_c_aa)
print(-unfair_strategic_h_aa,-unfair_non_strategic_h_aa)
print(-unfair_strategic_a_aa,-unfair_non_strategic_a_aa)
```

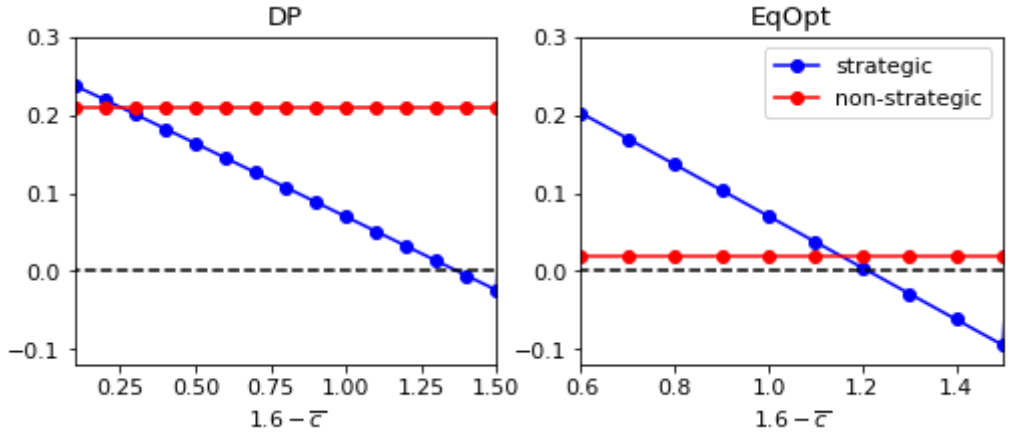
```
0.7942404109148179 0.44914087827756666
0.684228704065418 0.2417501236398395
0.8245954692312487 0.5218406394166468
```

**(2)**. $\alpha_a, \alpha_b < \frac{u_-}{u_+ + u_-}$ αa,αb<u−u++u−, there exists $C_b$Cb s.t. fairness is improved (disadvantaged group is flipped).

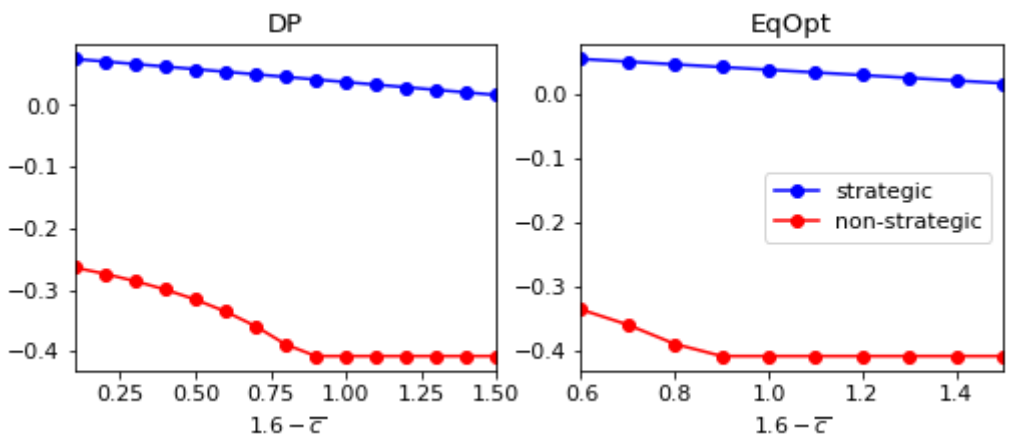
In [27]:
```
def Utility(threshold, u_cost,u_benefit,alpha,a0,b0,a1,b1,cost_a,cost_b,cost_type):
    Delta = beta.cdf(threshold, a0, b0) - beta.cdf(threshold, a1, b1)
    rho = u_cost*(1-alpha)/(u_benefit*alpha)
    if cost_type == "beta":
        U = beta.cdf(threshold,a0,b0)*(1-beta.cdf(Delta,cost_a,cost_b,0,1))*rho - beta.cdf(threshold,
a1,b1)*(1-rho*beta.cdf(Delta,cost_a,cost_b,0,1))
    if cost_type == "uniform":
        U = beta.cdf(threshold,a0,b0)*(1-uniform.cdf(Delta,cost_a,cost_b-cost_a))*rho - beta.cdf(thre
shold,a1,b1)*(1-rho*uniform.cdf(Delta,cost_a,cost_b-cost_a))
    return  U*u_benefit*alpha + u_benefit*alpha - u_cost*(1-alpha)
```

In [28]:
```
import matplotlib
matplotlib.rcParams.update({'font.size': 11})
from scipy.stats import uniform
u_cost,u_benefit = 2,1

'''
cost_type = "beta"
cost_aa_a,cost_aa_b = 10,5
cost_h_a,cost_h_b = 10,4
'''


cost_type = "uniform"
cost_aa_a,cost_aa_b = 0,1# 10,6
cost_h_a,cost_h_b = 0,2

Pa = P_h/(P_h+P_aa)

fair_type = "eqopt"
unfair_strategicList = []
utility_strategicList = []
utility_nonstrategicList = []
#b_list = [4,5,6,7,8,9,10,11]
b_list = np.arange(0,1.1,0.1)
for cost_h_b in b_list:
    _,_,opt_strategic_aa,opt_non_strategic_aa,opt_strategic_h,opt_non_strategic_h,_,_ = opt_threshold
(u_cost,u_benefit,alpha_c,alpha_aa,alpha_h,alpha_a,cost_c_a,cost_c_b,cost_aa_a,cost_aa_b,cost_h_a,cos
t_h_b,cost_a_a,cost_a_b,cost_type)

    fair_strategic_aa = fair_measure(opt_strategic_aa,a_aa0,b_aa0,a_aa1,b_aa1,alpha_aa,fair_type)
    fair_non_strategic_aa = fair_measure(opt_non_strategic_aa,a_aa0,b_aa0,a_aa1,b_aa1,alpha_aa,fair_t
ype)

    fair_strategic_h = fair_measure(opt_strategic_h,a_h0,b_h0,a_h1,b_h1,alpha_h,fair_type)
    fair_non_strategic_h = fair_measure(opt_non_strategic_h,a_h0,b_h0,a_h1,b_h1,alpha_h,fair_type)

    unfair_strategic_h_aa = fair_strategic_h - fair_strategic_aa
    unfair_non_strategic_h_aa = fair_non_strategic_h - fair_non_strategic_aa

    unfair_strategicList.append(-unfair_strategic_h_aa)


    Ua = Utility(opt_strategic_h,u_cost,u_benefit,alpha_h,a_h0,b_h0,a_h1,b_h1,cost_h_a,cost_h_b,cost_
type)
    Ub = Utility(opt_strategic_aa,u_cost,u_benefit,alpha_aa,a_aa0,b_aa0,a_aa1,b_aa1,cost_aa_a,cost_aa
_b,cost_type)
    utility_strategicList.append(Pa*Ua + (1-Pa)*Ub)
    Ua = Utility(opt_non_strategic_h,u_cost,u_benefit,alpha_h,a_h0,b_h0,a_h1,b_h1,cost_h_a,cost_h_b,c
ost_type)
    Ub = Utility(opt_non_strategic_aa,u_cost,u_benefit,alpha_aa,a_aa0,b_aa0,a_aa1,b_aa1,cost_aa_a,cos
t_aa_b,cost_type)
    utility_nonstrategicList.append(Pa*Ua + (1-Pa)*Ub)

fair_type = "dp"
unfair_strategicList1 = []
utility_strategicList1 = []
utility_nonstrategicList1 = []
#b_list1 = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]
b_list1 = np.arange(0,1.6,0.1)

for cost_h_b in b_list1:
    _,_,opt_strategic_aa,opt_non_strategic_aa,opt_strategic_h,opt_non_strategic_h,_,_ = opt_threshold
(u_cost,u_benefit,alpha_c,alpha_aa,alpha_h,alpha_a,cost_c_a,cost_c_b,cost_aa_a,cost_aa_b,cost_h_a,cos
t_h_b,cost_a_a,cost_a_b,cost_type)

    fair_strategic_aa = fair_measure(opt_strategic_aa,a_aa0,b_aa0,a_aa1,b_aa1,alpha_aa,fair_type)
    fair_non_strategic_aa = fair_measure(opt_non_strategic_aa,a_aa0,b_aa0,a_aa1,b_aa1,alpha_aa,fair_t
ype)

    fair_strategic_h = fair_measure(opt_strategic_h,a_h0,b_h0,a_h1,b_h1,alpha_h,fair_type)
    fair_non_strategic_h = fair_measure(opt_non_strategic_h,a_h0,b_h0,a_h1,b_h1,alpha_h,fair_type)

    unfair_strategic_h_aa = fair_strategic_h - fair_strategic_aa
    unfair_non_strategic_h_aa1 = fair_non_strategic_h - fair_non_strategic_aa
    unfair_strategicList1.append(-unfair_strategic_h_aa)

    Ua = Utility(opt_strategic_h,u_cost,u_benefit,alpha_h,a_h0,b_h0,a_h1,b_h1,cost_h_a,cost_h_b,cost_
type)
    Ub = Utility(opt_strategic_aa,u_cost,u_benefit,alpha_aa,a_aa0,b_aa0,a_aa1,b_aa1,cost_aa_a,cost_aa
_b,cost_type)
    utility_strategicList1.append(Pa*Ua + (1-Pa)*Ub)
    Ua = Utility(opt_non_strategic_h,u_cost,u_benefit,alpha_h,a_h0,b_h0,a_h1,b_h1,cost_h_a,cost_h_b,c
ost_type)
    Ub = Utility(opt_non_strategic_aa,u_cost,u_benefit,alpha_aa,a_aa0,b_aa0,a_aa1,b_aa1,cost_aa_a,cos
t_aa_b,cost_type)
    utility_nonstrategicList1.append(Pa*Ua + (1-Pa)*Ub)
```

In [29]:
```
b_list = [1.6-i for i in b_list]
b_list1 = [1.6-i for i in b_list1]
fig = plt.figure(figsize=(8.3, 3))
plt.subplot(1,2,2)
plt.plot(b_list,unfair_strategicList,'o-',color='b',label='strategic')
plt.plot(b_list,[-unfair_non_strategic_h_aa]*11,'o-',color='r',label='non-strategic')
plt.plot(b_list,[0]*11,'k--')
plt.title("EqOpt")
plt.xlabel(r'$1.6-\overline{c}$')
#plt.xlabel(r'$b$')
#plt.xlim([4,11])
plt.xlim([0.6,1.5])
plt.ylim([-0.12,0.3])
plt.legend()
plt.subplot(1,2,1)
plt.plot(b_list1,unfair_strategicList1,'o-',color='b',label='strategic')
plt.plot(b_list1,[-unfair_non_strategic_h_aa1]*16,'o-',color='r',label='non-strategic')
plt.plot(b_list1,[0]*16,'k--')
plt.title("DP")
#plt.xlim([1,20])
plt.xlim([0.1,1.5])
plt.ylim([-0.12,0.3])
plt.xlabel(r'$1.6-\overline{c}$')
#plt.xlabel(r'$b$')
plt.show()
#fig.savefig('improve_uniform.eps', bbox_inches='tight')
```




In [47]:
```
fig = plt.figure(figsize=(8.3, 3))
plt.subplot(1,2,2)
plt.plot(b_list,utility_strategicList,'o-',color='b',label='strategic')
plt.plot(b_list,utility_nonstrategicList,'o-',color='r',label='non-strategic')
plt.title("EqOpt")
plt.xlabel(r'$1.6-\overline{c}$')
#plt.xlabel(r'$b$')
#plt.xlim([4,11])
plt.xlim([0.6,1.5])
#plt.ylim([-0.12,0.3])
plt.legend()
plt.subplot(1,2,1)
plt.plot(b_list1,utility_strategicList1,'o-',color='b',label='strategic')
plt.plot(b_list1,utility_nonstrategicList1,'o-',color='r',label='non-strategic')
plt.title("DP")
#plt.xlim([1,20])
plt.xlim([0.1,1.5])
#plt.ylim([-0.12,0.3])
plt.xlabel(r'$1.6-\overline{c}$')
#plt.xlabel(r'$b$')
plt.show()
#fig.savefig('improve_uniform.eps', bbox_inches='tight')
```




## 2. Impact of fairness intervention

**(1)**. Fairness constraints EqOpt and DP

In [52]:
```
from scipy.optimize import newton
from pynverse import inversefunc
import warnings
warnings.filterwarnings("ignore")

# feasible pairs (x,f_eqopt(x)) that satisfy EqOpt
def f_eqopt(x,alpha_a,alpha_b,a0_a,b0_a,a1_a,b1_a,a0_b,b0_b,a1_b,b1_b):
    return beta.ppf(beta.cdf(x, a1_a,b1_a),a1_b,b1_b)

def Peqopt_s(x,alpha_s,a0_s,b0_s,a1_s,b1_s):
    return beta.pdf(x, a1_s,b1_s)

# feasible pairs (x,f_dp(x)) that satisfy DP
def f_dp(x,alpha_a,alpha_b,a0_a,b0_a,a1_a,b1_a,a0_b,b0_b,a1_b,b1_b):
    f_b = (lambda x_b: alpha_b*beta.cdf(x_b,a1_b,b1_b)+(1-alpha_b)*beta.cdf(x_b,a0_b,b0_b))
    inv_f_b = inversefunc(f_b,domain=[0, 1],open_domain=[False, False])
    return float(inv_f_b(alpha_a*beta.cdf(x,a1_a,b1_a)+(1-alpha_a)*beta.cdf(x,a0_a,b0_a)))

def Pdp_s(x,alpha_s,a0_s,b0_s,a1_s,b1_s):
    return alpha_s*beta.pdf(x,a1_s,b1_s)+(1-alpha_s)*beta.pdf(x,a0_s,b0_s)
```

In [289]:
```
def Phi(x,a0,b0,a1,b1,cost_a,cost_b,cost_type):
    Delta = beta.cdf(x, a0, b0, 0, 1) - beta.cdf(x, a1, b1, 0, 1)
    if cost_type == "uniform": # Uniformly distributed between cost_a & cost_b
        return uniform.cdf(Delta,cost_a,cost_b-cost_a) + Delta*uniform.pdf(Delta,cost_a,cost_b-cost_a
)
    if cost_type == "beta": # Beta
        return beta.cdf(Delta,cost_a,cost_b,0,1) + Delta*beta.pdf(Delta,cost_a,cost_b,0,1)
    if cost_type == "non-strategic":
        return 0

# optimal equations
def dir_Us(x,alpha,u_cost,u_benefit,cost_a,cost_b,a0,b0,a1,b1,Pc_s,cost_type):
    term1 = (beta.pdf(x,a1,b1) - beta.pdf(x,a0,b0))/Pc_s(x,alpha,a0,b0,a1,b1)
    term2 = (beta.pdf(x,a0,b0)*u_cost*(1-alpha) - beta.pdf(x,a1,b1)*u_benefit*alpha)/Pc_s(x,alpha,a0,
b0,a1,b1)
    return term1*Phi(x,a0,b0,a1,b1,cost_a,cost_b,cost_type)*u_cost*(1-alpha) + term2

def opt_eqn(x_a,alpha_a,alpha_b,Pa,f_c,u_cost,u_benefit,cost_a_a,cost_b_a,a0_a,b0_a,a1_a,b1_a,cost_a_
b,cost_b_b,a0_b,b0_b,a1_b,b1_b,Pc_s,cost_type):
    x_b = f_c(x_a,alpha_a,alpha_b,a0_a,b0_a,a1_a,b1_a,a0_b,b0_b,a1_b,b1_b)
    return Pa*dir_Us(x_a,alpha_a,u_cost,u_benefit,cost_a_a,cost_b_a,a0_a,b0_a,a1_a,b1_a,Pc_s,cost_typ
e) + (1-Pa)*dir_Us(x_b,alpha_b,u_cost,u_benefit,cost_a_b,cost_b_b,a0_b,b0_b,a1_b,b1_b,Pc_s,cost_type)

def get_policy(f_c,alpha_a,alpha_b,Pa,u_cost,u_benefit,cost_a_a,cost_b_a,a0_a,b0_a,a1_a,b1_a,cost_a_b
,cost_b_b,a0_b,b0_b,a1_b,b1_b,Pc_s,cost_type):
    root=[]
    for i in np.arange(0.01,0.99,0.01):
        try:
            root.append(newton(opt_eqn,x0 = i,tol=1.48e-8,maxiter=50,args = (alpha_a,alpha_b,Pa,f_c,u
_cost,u_benefit,cost_a_a,cost_b_a,a0_a,b0_a,a1_a,b1_a,cost_a_b,cost_b_b,a0_b,b0_b,a1_b,b1_b,Pc_s,cost
_type)))
        except(RuntimeError):
            pass
    root = [float(format(round(r, 4))) for r in root]
    root_a = np.unique(root)
    root_b = [f_c(i,alpha_a,alpha_b,a0_a,b0_a,a1_a,b1_a,a0_b,b0_b,a1_b,b1_b) for i in root_a]


    return root_a,root_b
```

In [468]:
```
def manipulation_prob(theta_un,theta_eqopt,theta_dp,a0,b0,a1,b1,cost_a,cost_b,cost_type):
    Delta_un = beta.cdf(theta_un, a0, b0) - beta.cdf(theta_un, a1, b1)
    Delta_eqopt = beta.cdf(theta_eqopt, a0, b0) - beta.cdf(theta_eqopt, a1, b1)
    Delta_dp = beta.cdf(theta_dp, a0, b0) - beta.cdf(theta_dp, a1, b1)
    if cost_type == "uniform":
        p_un = uniform.cdf(Delta_un,cost_a,cost_b-cost_a)
        p_eqopt = uniform.cdf(Delta_eqopt,cost_a,cost_b-cost_a)
        p_dp = uniform.cdf(Delta_dp,cost_a,cost_b-cost_a)
    if cost_type == "beta":
        p_un = beta.cdf(Delta_un,cost_a,cost_b)
        p_eqopt = beta.cdf(Delta_eqopt,cost_a,cost_b)
        p_dp = beta.cdf(Delta_dp,cost_a,cost_b)
    return p_un,p_eqopt,p_dp


def U_strategic(theta,u_cost,u_benefit,alpha,a0,b0,a1,b1,cost_a,cost_b,cost_type):
    Delta = beta.cdf(theta, a0, b0, 0, 1) - beta.cdf(theta, a1, b1, 0, 1)
    rho = u_cost*(1-alpha)/(u_benefit*alpha)
    if cost_type == "beta":
        U = beta.cdf(theta,a0,b0)*(1-beta.cdf(Delta,cost_a,cost_b,0,1))*rho - beta.cdf(theta,a1,b1)*(
1-rho*beta.cdf(Delta,cost_a,cost_b,0,1))
    if cost_type == "uniform":
        U = beta.cdf(theta,a0,b0)*(1-uniform.cdf(Delta,cost_a,cost_b-cost_a))*rho - beta.cdf(theta,a1
,b1)*(1-rho*uniform.cdf(Delta,cost_a,cost_b-cost_a))

    return U*u_benefit*alpha + u_benefit*alpha - u_cost*(1-alpha)

def find_max(root_a,root_b,pa,u_cost,u_benefit,alpha_a,a0_a,b0_a,a1_a,b1_a,cost_a_a,cost_b_a,alpha_b,
a0_b,b0_b,a1_b,b1_b,cost_a_b,cost_b_b,cost_type):
    U_tot = []
    for i in range(len(root_a)):
        Ua = U_strategic(root_a[i],u_cost,u_benefit,alpha_a,a0_a,b0_a,a1_a,b1_a,cost_a_a,cost_b_a,cos
t_type)
        Ub = U_strategic(root_b[i],u_cost,u_benefit,alpha_b,a0_b,b0_b,a1_b,b1_b,cost_a_b,cost_b_b,cos
t_type)
        U_tot.append(pa*Ua + (1-pa)*Ub)
    Idx = np.argmax(U_tot)
    return root_a[Idx],root_b[Idx]
```

**Non-strategic fair policies may attain the higher utility for both groups than non-strategic policies**

In [270]:
```
cost_type = "beta"
u_cost,u_benefit = 4,1

alpha_a,alpha_b = alpha_c,alpha_asian
a0_a,b0_a,a1_a,b1_a = a_c0,b_c0,a_c1,b_c1
a0_b,b0_b,a1_b,b1_b = a_a0,b_a0,a_a1,b_a1


_,opt_non_strategic_aa = opt_non_strategic(u_cost,u_benefit,alpha_aa,a_aa0,b_aa0,a_aa1,b_aa1)
_,opt_non_strategic_h = opt_non_strategic(u_cost,u_benefit,alpha_h,a_h0,b_h0,a_h1,b_h1)
_,opt_non_strategic_c = opt_non_strategic(u_cost,u_benefit,alpha_c,a_c0,b_c0,a_c1,b_c1)
_,opt_non_strategic_asian = opt_non_strategic(u_cost,u_benefit,alpha_asian,a_a0,b_a0,a_a1,b_a1)


U1_a,opt_strategic_a = opt_strategic(u_cost,u_benefit,alpha_a,a0_a,b0_a,a1_a,b1_a,cost_a_a,cost_b_a,c
ost_type)
U1,opt_strategic_b = opt_strategic(u_cost,u_benefit,alpha_b,a0_b,b0_b,a1_b,b1_b,cost_a_b,cost_b_b,cos
t_type)



if alpha_a < u_cost/(u_cost+u_benefit) and alpha_b > u_cost/(u_cost+u_benefit):
    print('yes')
else:
    print('no')
if beta.cdf(opt_non_strategic_b,a1_b,b1_b) > beta.cdf(opt_non_strategic_a,a1_a,b1_a):
    print('yes')
else:
    print('no')


pa = P_c/(P_a + P_c)

cost_a_a,cost_b_a = 10,10
cost_a_b,cost_b_b = 10,10
```

```
0.2563011954095574 0.25507804277256774 0.05588194036328951 0.12635526994589577
yes
yes
```

In [271]:
```
cost_type = "beta"
u_cost,u_benefit = 3.1,1

alpha_a,alpha_b = alpha_c,alpha_asian
a0_a,b0_a,a1_a,b1_a = a_c0,b_c0,a_c1,b_c1
#a0_a,b0_a,a1_a,b1_a = a_h0,b_h0,a_h1,b_h1
#a0_a,b0_a,a1_a,b1_a = a_a0,b_a0,a_a1,b_a1
#a0_a,b0_a,a1_a,b1_a = a_aa0,b_aa0,a_aa1,b_aa1


a0_b,b0_b,a1_b,b1_b = a_a0,b_a0,a_a1,b_a1
#a0_b,b0_b,a1_b,b1_b = a_h0,b_h0,a_h1,b_h1
#a0_b,b0_b,a1_b,b1_b = a_aa0,b_aa0,a_aa1,b_aa1


U1_a,opt_strategic_a = opt_strategic(u_cost,u_benefit,alpha_a,a0_a,b0_a,a1_a,b1_a,cost_a_a,cost_b_a,c
ost_type)
U1,opt_strategic_b = opt_strategic(u_cost,u_benefit,alpha_b,a0_b,b0_b,a1_b,b1_b,cost_a_b,cost_b_b,cos
t_type)
U,opt_non_strategic_b = opt_non_strategic(u_cost,u_benefit,alpha_b,a0_b,b0_b,a1_b,b1_b)
U_a,opt_non_strategic_a = opt_non_strategic(u_cost,u_benefit,alpha_a,a0_a,b0_a,a1_a,b1_a)


if alpha_a > u_cost/(u_cost+u_benefit) and alpha_b > u_cost/(u_cost+u_benefit):
    print('yes')
else:
    print('no')
if beta.cdf(opt_non_strategic_a,a1_a,b1_a) < beta.cdf(opt_non_strategic_b,a1_b,b1_b):
    print('yes')
else:
    print('no')


pa = P_c/(P_a + P_c)

cost_a_a,cost_b_a = 10,1
cost_a_b,cost_b_b = 10,10
```

```
yes
yes
```

In [273]:
```
# non-strategic optimal policies
U,opt_non_strategic_b = opt_non_strategic(u_cost,u_benefit,alpha_b,a0_b,b0_b,a1_b,b1_b)
U_a,opt_non_strategic_a = opt_non_strategic(u_cost,u_benefit,alpha_a,a0_a,b0_a,a1_a,b1_a)


# non-strategic fair optimal policies
root_eqopt_a,root_eqopt_b = get_policy(f_eqopt,alpha_a,alpha_b,pa,u_cost,u_benefit,cost_a_a,cost_b_a,
a0_a,b0_a,a1_a,b1_a,cost_a_b,cost_b_b,a0_b,b0_b,a1_b,b1_b,Peqopt_s,"non-strategic")
theta_nonstrategic_eqopt_a,theta_nonstrategic_eqopt_b = find_max(root_eqopt_a,root_eqopt_b,pa,u_cost,
u_benefit,alpha_a,a0_a,b0_a,a1_a,b1_a,cost_a_a,cost_b_a,alpha_b,a0_b,b0_b,a1_b,b1_b,cost_a_b,cost_b_b
,cost_type)
root_dp_a,root_dp_b = get_policy(f_dp,alpha_a,alpha_b,pa,u_cost,u_benefit,cost_a_a,cost_b_a,a0_a,b0_a
,a1_a,b1_a,cost_a_b,cost_b_b,a0_b,b0_b,a1_b,b1_b,Pdp_s,"non-strategic")
theta_nonstrategic_dp_a,theta_nonstrategic_dp_b = find_max(root_dp_a,root_dp_b,pa,u_cost,u_benefit,al
pha_a,a0_a,b0_a,a1_a,b1_a,cost_a_a,cost_b_a,alpha_b,a0_b,b0_b,a1_b,b1_b,cost_a_b,cost_b_b,cost_type)


Ua_un = U_strategic(opt_non_strategic_a,u_cost,u_benefit,alpha_a,a0_a,b0_a,a1_a,b1_a,cost_a_a,cost_b_
a,cost_type)
Ub_un = U_strategic(opt_non_strategic_b,u_cost,u_benefit,alpha_b,a0_b,b0_b,a1_b,b1_b,cost_a_b,cost_b_
b,cost_type)

Ua_dp = U_strategic(theta_nonstrategic_dp_a,u_cost,u_benefit,alpha_a,a0_a,b0_a,a1_a,b1_a,cost_a_a,cos
t_b_a,cost_type)
Ub_dp = U_strategic(theta_nonstrategic_dp_b,u_cost,u_benefit,alpha_b,a0_b,b0_b,a1_b,b1_b,cost_a_b,cos
t_b_b,cost_type)

Ua_eqopt = U_strategic(theta_nonstrategic_eqopt_a,u_cost,u_benefit,alpha_a,a0_a,b0_a,a1_a,b1_a,cost_a
_a,cost_b_a,cost_type)
Ub_eqopt = U_strategic(theta_nonstrategic_eqopt_b,u_cost,u_benefit,alpha_b,a0_b,b0_b,a1_b,b1_b,cost_a
_b,cost_b_b,cost_type)


print(Ua_un,Ua_eqopt,Ua_dp)
print(Ub_un,Ub_eqopt,Ub_dp)
```

```
-0.02379416865461126 -0.023089378705530406 0.04599263345208504
-0.47684220282599094 -0.4847305592889497 -0.521029388337684
```

In [254]:
```
# strategic optimal policies
U1_a,opt_strategic_a = opt_strategic(u_cost,u_benefit,alpha_a,a0_a,b0_a,a1_a,b1_a,cost_a_a,cost_b_a,c
ost_type)
U1,opt_strategic_b = opt_strategic(u_cost,u_benefit,alpha_b,a0_b,b0_b,a1_b,b1_b,cost_a_b,cost_b_b,cos
t_type)

# strategic fair optimal policies
root_eqopt_a,root_eqopt_b = get_policy(f_eqopt,alpha_a,alpha_b,pa,u_cost,u_benefit,cost_a_a,cost_b_a,
a0_a,b0_a,a1_a,b1_a,cost_a_b,cost_b_b,a0_b,b0_b,a1_b,b1_b,Peqopt_s,cost_type)
theta_eqopt_a,theta_eqopt_b = find_max(root_eqopt_a,root_eqopt_b,pa,u_cost,u_benefit,alpha_a,a0_a,b0_
a,a1_a,b1_a,cost_a_a,cost_b_a,alpha_b,a0_b,b0_b,a1_b,b1_b,cost_a_b,cost_b_b,cost_type)

root_dp_a,root_dp_b = get_policy(f_dp,alpha_a,alpha_b,pa,u_cost,u_benefit,cost_a_a,cost_b_a,a0_a,b0_a
,a1_a,b1_a,cost_a_b,cost_b_b,a0_b,b0_b,a1_b,b1_b,Pdp_s,cost_type)
theta_dp_a,theta_dp_b = find_max(root_dp_a,root_dp_b,pa,u_cost,u_benefit,alpha_a,a0_a,b0_a,a1_a,b1_a,
cost_a_a,cost_b_a,alpha_b,a0_b,b0_b,a1_b,b1_b,cost_a_b,cost_b_b,cost_type)

# manipulation probabilities
p_un_a,p_eqopt_a,p_dp_a = manipulation_prob(opt_strategic_a,theta_eqopt_a,theta_dp_a,a0_a,b0_a,a1_a,b
1_a,cost_a_a,cost_b_a,cost_type)
p_un_b,p_eqopt_b,p_dp_b = manipulation_prob(opt_strategic_b,theta_eqopt_b,theta_dp_b,a0_b,b0_b,a1_b,b
1_b,cost_a_b,cost_b_b,cost_type)

print(opt_strategic_b,theta_eqopt_b,theta_dp_b)
print(opt_strategic_a,theta_eqopt_a,theta_dp_a)

print('-----------------------')
print(p_un_a,p_eqopt_a,p_dp_a)
print(p_un_b,p_eqopt_b,p_dp_b)
```

```
0.08010801080108011 0.10545073155141539 0.23871799478756256
0.1427142714271427 0.1413 0.1424
-----------------------
0.09071932287835724 0.08661147840702481 0.08979827631750886
0.15238276596368167 0.3842590182693554 0.97813123183322
```

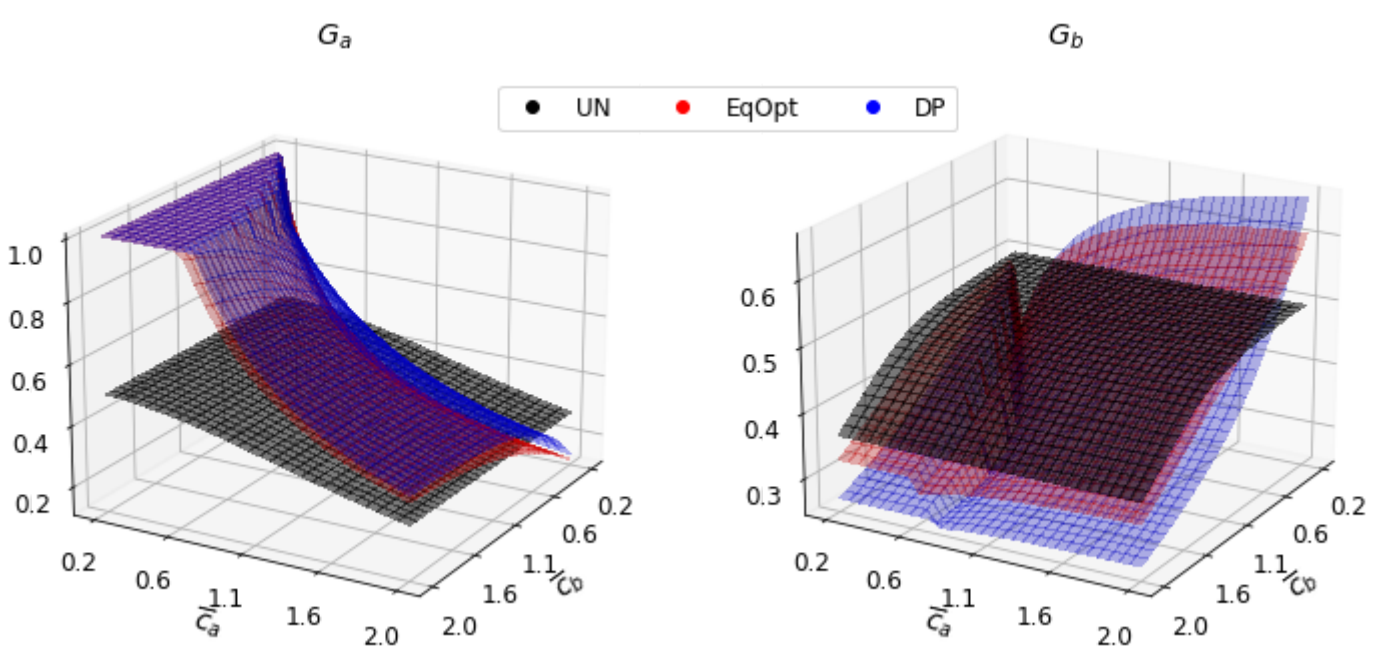
In [13]:
```
cost_type = "uniform"
u_cost,u_benefit = 1,1

alpha_a,alpha_b = alpha_asian,alpha_h
#a0_a,b0_a,a1_a,b1_a = a_c0,b_c0,a_c1,b_c1
#a0_a,b0_a,a1_a,b1_a = a_h0,b_h0,a_h1,b_h1
a0_a,b0_a,a1_a,b1_a = a_a0,b_a0,a_a1,b_a1

#a0_b,b0_b,a1_b,b1_b = a_aa0,b_aa0,a_aa1,b_aa1
a0_b,b0_b,a1_b,b1_b = a_h0,b_h0,a_h1,b_h1
#a0_b,b0_b,a1_b,b1_b = a_a0,b_a0,a_a1,b_a1


pa = P_a/(P_a + P_h)

U,opt_non_strategic_b = opt_non_strategic(u_cost,u_benefit,alpha_b,a0_b,b0_b,a1_b,b1_b)
U_a,opt_non_strategic_a = opt_non_strategic(u_cost,u_benefit,alpha_a,a0_a,b0_a,a1_a,b1_a)


N = 30
p_un_aList,p_eqopt_aList,p_dp_aList = np.zeros([N,N]),np.zeros([N,N]),np.zeros([N,N])
p_un_bList,p_eqopt_bList,p_dp_bList = np.zeros([N,N]),np.zeros([N,N]),np.zeros([N,N])

i = -1
for c_a in np.linspace(0.2,2,N):
    i+=1
    j = -1
    for c_b in np.linspace(0.2,2,N):
        j += 1
        cost_b_a,cost_b_b = c_a,c_b
        cost_a_a,cost_a_b = 0,0

        U1_a,opt_strategic_a = opt_strategic(u_cost,u_benefit,alpha_a,a0_a,b0_a,a1_a,b1_a,cost_a_a,co
st_b_a,cost_type)
        U1,opt_strategic_b = opt_strategic(u_cost,u_benefit,alpha_b,a0_b,b0_b,a1_b,b1_b,cost_a_b,cost
_b_b,cost_type)

        root_eqopt_a,root_eqopt_b = get_policy(f_eqopt,alpha_a,alpha_b,pa,u_cost,u_benefit,cost_a_a,c
ost_b_a,a0_a,b0_a,a1_a,b1_a,cost_a_b,cost_b_b,a0_b,b0_b,a1_b,b1_b,Peqopt_s,cost_type)
        theta_eqopt_a,theta_eqopt_b = find_max(root_eqopt_a,root_eqopt_b,pa,u_cost,u_benefit,alpha_a,
a0_a,b0_a,a1_a,b1_a,cost_a_a,cost_b_a,alpha_b,a0_b,b0_b,a1_b,b1_b,cost_a_b,cost_b_b,cost_type)

        root_dp_a,root_dp_b = get_policy(f_dp,alpha_a,alpha_b,pa,u_cost,u_benefit,cost_a_a,cost_b_a,a
0_a,b0_a,a1_a,b1_a,cost_a_b,cost_b_b,a0_b,b0_b,a1_b,b1_b,Pdp_s,cost_type)
        theta_dp_a,theta_dp_b = find_max(root_dp_a,root_dp_b,pa,u_cost,u_benefit,alpha_a,a0_a,b0_a,a1
_a,b1_a,cost_a_a,cost_b_a,alpha_b,a0_b,b0_b,a1_b,b1_b,cost_a_b,cost_b_b,cost_type)

        p_un_aList[i,j],p_eqopt_aList[i,j],p_dp_aList[i,j] = manipulation_prob(opt_strategic_a,theta_
eqopt_a,theta_dp_a,a0_a,b0_a,a1_a,b1_a,cost_a_a,cost_b_a,cost_type)
        p_un_bList[i,j],p_eqopt_bList[i,j],p_dp_bList[i,j] = manipulation_prob(opt_strategic_b,theta_
eqopt_b,theta_dp_b,a0_b,b0_b,a1_b,b1_b,cost_a_b,cost_b_b,cost_type)

        if p_un_aList[i,j] > p_eqopt_aList[i,j] and p_un_bList[i,j] > p_eqopt_bList[i,j]:
            print('eqopt')
        if p_un_aList[i,j] > p_dp_aList[i,j] and p_un_bList[i,j] > p_dp_bList[i,j]:
            print('dp')
```

In [506]:
```
cost_type = "uniform"
u_cost,u_benefit = 1,1

alpha_a,alpha_b = alpha_c,alpha_asian
a0_a,b0_a,a1_a,b1_a = a_c0,b_c0,a_c1,b_c1
#a0_a,b0_a,a1_a,b1_a = a_h0,b_h0,a_h1,b_h1
#a0_a,b0_a,a1_a,b1_a = a_a0,b_a0,a_a1,b_a1

#a0_b,b0_b,a1_b,b1_b = a_aa0,b_aa0,a_aa1,b_aa1
#a0_b,b0_b,a1_b,b1_b = a_h0,b_h0,a_h1,b_h1
a0_b,b0_b,a1_b,b1_b = a_a0,b_a0,a_a1,b_a1


pa = P_c/(P_c + P_a)

U,opt_non_strategic_b = opt_non_strategic(u_cost,u_benefit,alpha_b,a0_b,b0_b,a1_b,b1_b)
U_a,opt_non_strategic_a = opt_non_strategic(u_cost,u_benefit,alpha_a,a0_a,b0_a,a1_a,b1_a)


N = 20
p_un_aList,p_eqopt_aList,p_dp_aList = np.zeros([N,N]),np.zeros([N,N]),np.zeros([N,N])
p_un_bList,p_eqopt_bList,p_dp_bList = np.zeros([N,N]),np.zeros([N,N]),np.zeros([N,N])


cost_b_a,cost_b_b = 0.2,2
cost_a_a,cost_a_b = 0,0

U1_a,opt_strategic_a = opt_strategic(u_cost,u_benefit,alpha_a,a0_a,b0_a,a1_a,b1_a,cost_a_a,cost_b_a,c
ost_type)
U1,opt_strategic_b = opt_strategic(u_cost,u_benefit,alpha_b,a0_b,b0_b,a1_b,b1_b,cost_a_b,cost_b_b,cos
t_type)

root_eqopt_a,root_eqopt_b = get_policy(f_eqopt,alpha_a,alpha_b,pa,u_cost,u_benefit,cost_a_a,cost_b_a,
a0_a,b0_a,a1_a,b1_a,cost_a_b,cost_b_b,a0_b,b0_b,a1_b,b1_b,Peqopt_s,cost_type)
theta_eqopt_a,theta_eqopt_b = find_max(root_eqopt_a,root_eqopt_b,pa,u_cost,u_benefit,alpha_a,a0_a,b0_
a,a1_a,b1_a,cost_a_a,cost_b_a,alpha_b,a0_b,b0_b,a1_b,b1_b,cost_a_b,cost_b_b,cost_type)

root_dp_a,root_dp_b = get_policy(f_dp,alpha_a,alpha_b,pa,u_cost,u_benefit,cost_a_a,cost_b_a,a0_a,b0_a
,a1_a,b1_a,cost_a_b,cost_b_b,a0_b,b0_b,a1_b,b1_b,Pdp_s,cost_type)
theta_dp_a,theta_dp_b = find_max(root_dp_a,root_dp_b,pa,u_cost,u_benefit,alpha_a,a0_a,b0_a,a1_a,b1_a,
cost_a_a,cost_b_a,alpha_b,a0_b,b0_b,a1_b,b1_b,cost_a_b,cost_b_b,cost_type)

p_un_aList,p_eqopt_aList,p_dp_aList = manipulation_prob(opt_strategic_a,theta_eqopt_a,theta_dp_a,a0_a
,b0_a,a1_a,b1_a,cost_a_a,cost_b_a,cost_type)
p_un_bList,p_eqopt_bList,p_dp_bList = manipulation_prob(opt_strategic_b,theta_eqopt_b,theta_dp_b,a0_b
,b0_b,a1_b,b1_b,cost_a_b,cost_b_b,cost_type)

print(p_un_aList,p_eqopt_aList,p_dp_aList)
print(p_un_bList,p_eqopt_bList,p_dp_bList)
```

```
0.5001259430635678 0.5118424444870163 0.52362913223781
0.2871276436682684 0.030272625598225435 0.06395735079354069
```

In [14]:
```
cost_type = "beta"
u_cost,u_benefit = 1,1

alpha_a,alpha_b = alpha_aa,alpha_h
#a0_a,b0_a,a1_a,b1_a = a_c0,b_c0,a_c1,b_c1
#a0_a,b0_a,a1_a,b1_a = a_h0,b_h0,a_h1,b_h1
#a0_a,b0_a,a1_a,b1_a = a_a0,b_a0,a_a1,b_a1
a0_a,b0_a,a1_a,b1_a = a_aa0,b_aa0,a_aa1,b_aa1

#a0_b,b0_b,a1_b,b1_b = a_aa0,b_aa0,a_aa1,b_aa1
a0_b,b0_b,a1_b,b1_b = a_h0,b_h0,a_h1,b_h1
#a0_b,b0_b,a1_b,b1_b = a_a0,b_a0,a_a1,b_a1


pa = P_aa/(P_aa + P_h)

U,opt_non_strategic_b = opt_non_strategic(u_cost,u_benefit,alpha_b,a0_b,b0_b,a1_b,b1_b)
U_a,opt_non_strategic_a = opt_non_strategic(u_cost,u_benefit,alpha_a,a0_a,b0_a,a1_a,b1_a)


cost_a_a,cost_b_a = 0,1
cost_a_b,cost_b_b = 0,1

N = 30
p_un_aList,p_eqopt_aList,p_dp_aList = np.zeros([N,N]),np.zeros([N,N]),np.zeros([N,N])
p_un_bList,p_eqopt_bList,p_dp_bList = np.zeros([N,N]),np.zeros([N,N]),np.zeros([N,N])

i = -1
for c_a in np.linspace(1,15,N):
    i +=1
    j = -1
    for c_b in np.linspace(1,15,N):
        print(c_a,c_b)
        j += 1
        cost_b_a,cost_b_b = c_b,c_b
        cost_a_a,cost_a_b = c_a,c_a

        U1_a,opt_strategic_a = opt_strategic(u_cost,u_benefit,alpha_a,a0_a,b0_a,a1_a,b1_a,cost_a_a,co
st_b_a,cost_type)
        U1,opt_strategic_b = opt_strategic(u_cost,u_benefit,alpha_b,a0_b,b0_b,a1_b,b1_b,cost_a_b,cost
_b_b,cost_type)

        root_eqopt_a,root_eqopt_b = get_policy(f_eqopt,alpha_a,alpha_b,pa,u_cost,u_benefit,cost_a_a,c
ost_b_a,a0_a,b0_a,a1_a,b1_a,cost_a_b,cost_b_b,a0_b,b0_b,a1_b,b1_b,Peqopt_s,cost_type)
        theta_eqopt_a,theta_eqopt_b = find_max(root_eqopt_a,root_eqopt_b,pa,u_cost,u_benefit,alpha_a,
a0_a,b0_a,a1_a,b1_a,cost_a_a,cost_b_a,alpha_b,a0_b,b0_b,a1_b,b1_b,cost_a_b,cost_b_b,cost_type)

        root_dp_a,root_dp_b = get_policy(f_dp,alpha_a,alpha_b,pa,u_cost,u_benefit,cost_a_a,cost_b_a,a
0_a,b0_a,a1_a,b1_a,cost_a_b,cost_b_b,a0_b,b0_b,a1_b,b1_b,Pdp_s,cost_type)
        theta_dp_a,theta_dp_b = find_max(root_dp_a,root_dp_b,pa,u_cost,u_benefit,alpha_a,a0_a,b0_a,a1
_a,b1_a,cost_a_a,cost_b_a,alpha_b,a0_b,b0_b,a1_b,b1_b,cost_a_b,cost_b_b,cost_type)

        p_un_aList[i,j],p_eqopt_aList[i,j],p_dp_aList[i,j] = manipulation_prob(opt_strategic_a,theta_
eqopt_a,theta_dp_a,a0_a,b0_a,a1_a,b1_a,cost_a_a,cost_b_a,cost_type)
        p_un_bList[i,j],p_eqopt_bList[i,j],p_dp_bList[i,j] = manipulation_prob(opt_strategic_b,theta_
eqopt_b,theta_dp_b,a0_b,b0_b,a1_b,b1_b,cost_a_b,cost_b_b,cost_type)

        if p_un_aList[i,j] > p_eqopt_aList[i,j] and p_un_bList[i,j] > p_eqopt_bList[i,j]:
            print('eqopt')
        if p_un_aList[i,j] > p_dp_aList[i,j] and p_un_bList[i,j] > p_dp_bList[i,j]:
            print('dp')
```

In [535]:
```
matplotlib.rcParams.update({'font.size': 12})

fig = plt.figure(figsize=(12,6))
ax1 = fig.add_subplot(121, projection='3d')
ax2 = fig.add_subplot(122, projection='3d')

X = np.linspace(0.2,2,30)
Y = np.linspace(0.2,2,30)

X, Y = np.meshgrid(X, Y)

surf = ax1.plot_surface(X, Y, p_un_aList1, color='black',linewidth=0, antialiased=False,alpha=0.5,lab
el='UN')
surf1 = ax1.plot_surface(X, Y, p_eqopt_aList1, color='red',linewidth=0, antialiased=False,alpha=0.3,l
abel='EqOpt')
surf1 = ax1.plot_surface(X, Y, p_dp_aList1, color='blue',linewidth=0, antialiased=False,alpha=0.3,lab
el='DP')


surf2 = ax2.plot_surface(X, Y, p_un_bList1,color='black',linewidth=0, antialiased=False,alpha=0.5)
surf3 = ax2.plot_surface(X, Y, p_eqopt_bList1,color='red',linewidth=0, antialiased=False,alpha=0.3)
surf3 = ax2.plot_surface(X, Y, p_dp_bList1, color='blue',linewidth=0, antialiased=False,alpha=0.3)

fake2Dline1 = matplotlib.lines.Line2D([0],[0], linestyle="none", c='black', marker = 'o')
fake2Dline2 = matplotlib.lines.Line2D([0],[0], linestyle="none", c='red', marker = 'o')
fake2Dline3 = matplotlib.lines.Line2D([0],[0], linestyle="none", c='blue', marker = 'o')
ax2.legend([fake2Dline1,fake2Dline2,fake2Dline3], ['UN','EqOpt','DP'], numpoints = 1,loc='upper righ
t', bbox_to_anchor=(0.34, 0.97),ncol=3)

ax1.title.set_text(r'$G_a$')
ax2.title.set_text(r'$G_b$')
ax1.set_ylabel(r'$\overline{c}_a$',fontsize=14)
ax1.set_xlabel(r'$\overline{c}_b$',fontsize=14)
ax2.set_ylabel(r'$\overline{c}_a$',fontsize=14)
ax2.set_xlabel(r'$\overline{c}_b$',fontsize=14)


ax1.set_xticklabels(list(np.round(np.linspace(0.2,2,5),1)))
ax2.set_xticklabels(list(np.round(np.linspace(0.2,2,5),1)))
ax1.set_yticklabels(list(np.round(np.linspace(0.2,2,5),1)))
ax2.set_yticklabels(list(np.round(np.linspace(0.2,2,5),1)))
ax1.set_xticks(np.linspace(0.2,2,5))
ax2.set_xticks(np.linspace(0.2,2,5))
ax1.set_yticks(np.linspace(0.2,2,5))
ax2.set_yticks(np.linspace(0.2,2,5))


#ax1.view_init(-140, 0)
ax1.view_init(20, 30)
ax2.view_init(20, 30)

plt.savefig('uniform_manipulation_a_h.pdf',bbox_inches='tight')

plt.show()
```



In [537]:
```
pDiff_eqopt_a1 = p_un_aList1-p_eqopt_aList1
pDiff_dp_a1 = p_un_aList1-p_dp_aList1
pDiff_eqopt_b1 = p_un_bList1-p_eqopt_bList1
pDiff_dp_b1 = p_un_bList1-p_dp_bList1

fig = plt.figure(figsize=(19,6))
ax1 = fig.add_subplot(141)
ax2 = fig.add_subplot(142)
ax3 = fig.add_subplot(143)
ax4 = fig.add_subplot(144)


im1 = ax1.imshow(pDiff_dp_a1,cmap = 'gist_rainbow',origin='lower',vmin=-0.45, vmax=0.45)
ax1.set_xlabel(r'$\overline{c}_b$',fontsize=14)
ax1.set_ylabel(r'$\overline{c}_a$',fontsize=14)

im2 = ax2.imshow(pDiff_dp_b1,cmap = 'gist_rainbow',origin='lower',vmin=-0.45, vmax=0.45)
ax2.set_xlabel(r'$\overline{c}_b$',fontsize=14)


im3 = ax3.imshow(pDiff_eqopt_a1,cmap = 'gist_rainbow',origin='lower',vmin=-0.45, vmax=0.45)
ax3.set_xlabel(r'$\overline{c}_b$',fontsize=14)

im4 = ax4.imshow(pDiff_eqopt_b1,cmap = 'gist_rainbow',origin='lower',vmin=-0.45, vmax=0.45)
ax4.set_xlabel(r'$\overline{c}_b$',fontsize=14)


ax2.set_xticklabels(list(np.round(np.linspace(0.2,2,5),1)))
ax1.set_xticklabels(list(np.round(np.linspace(0.2,2,5),1)))
ax2.set_yticklabels(list(np.round(np.linspace(0.2,2,5),1)))
ax1.set_yticklabels(list(np.round(np.linspace(0.2,2,5),1)))
ax1.set_xticks(np.linspace(0,29,5))
ax2.set_xticks(np.linspace(0,29,5))
ax1.set_yticks(np.linspace(0,29,5))
ax2.set_yticks(np.linspace(0,29,5))

ax4.set_xticklabels(list(np.round(np.linspace(0.2,2,5),1)))
ax3.set_xticklabels(list(np.round(np.linspace(0.2,2,5),1)))
ax4.set_yticklabels(list(np.round(np.linspace(0.2,2,5),1)))
ax3.set_yticklabels(list(np.round(np.linspace(0.2,2,5),1)))
ax3.set_xticks(np.linspace(0,29,5))
ax4.set_xticks(np.linspace(0,29,5))
ax3.set_yticks(np.linspace(0,29,5))
ax4.set_yticks(np.linspace(0,29,5))

ax1.title.set_text(r'$G_a$: DP')
ax2.title.set_text(r'$G_b$: DP')

ax3.title.set_text(r'$G_a$: EqOpt')
ax4.title.set_text(r'$G_b$: EqOpt')

fig.subplots_adjust(right=0.84)
cbar_ax = fig.add_axes([0.85, 0.26, 0.015, 0.49])
fig.colorbar(im1,cax=cbar_ax)


plt.savefig('Diff_uniform_a_h.eps',bbox_inches='tight')
plt.show()
```