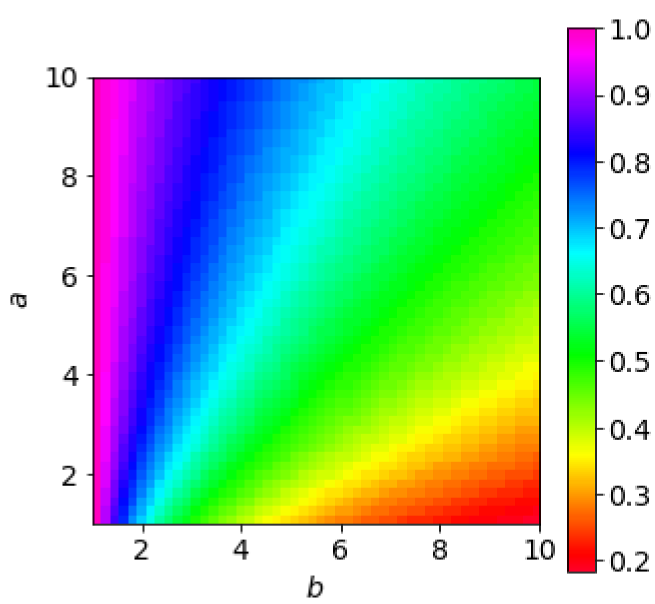
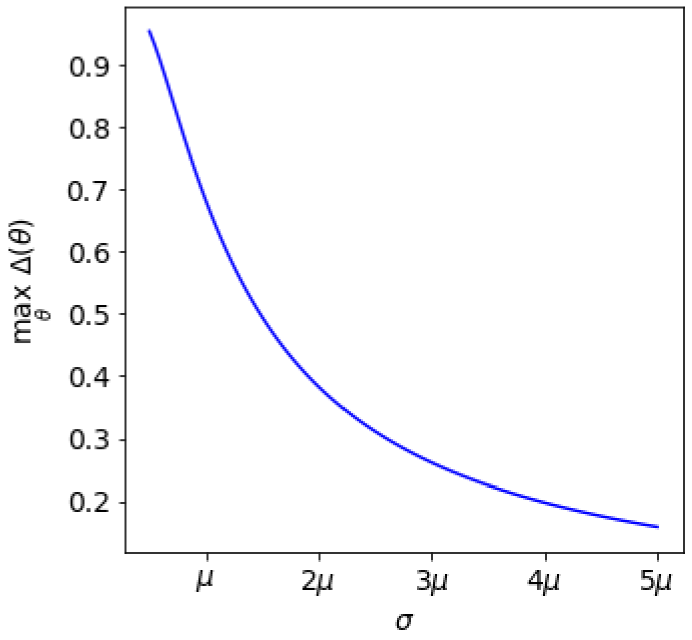
beta distributed cost:
$C \sim Beta(a,b), a \in [1,10], b \in [1,10]$C~Beta(a,b),a∈[1,10],b∈[1,10]

In [55]:
```
from scipy.stats import norm,lognorm,beta
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
matplotlib.rcParams.update({'font.size': 14})

a,b = 10,2
loc, scale = 0,1

x1 = np.linspace(0,1,100)
G_max = np.zeros([50,50])
i = -1
for a in np.linspace(1,10,50):
    i+=1
    j = -1
    for b in np.linspace(1,10,50):
        j+=1
        Phi = beta.cdf(x1, a, b, loc, scale) + x1*beta.pdf(x1, a, b, loc, scale)
        Idx = np.argmax(Phi)
        G_max[i,j] = x1[Idx]
        if j == 11 and i == 38:
            print(a,b)
            print(x1[Idx])

plt.figure(figsize=(5,5))
plt.imshow(G_max,origin='lower',extent=[1,10,1,10],cmap='gist_rainbow')
plt.xlabel(r'$b$')
plt.ylabel(r'$a$')

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

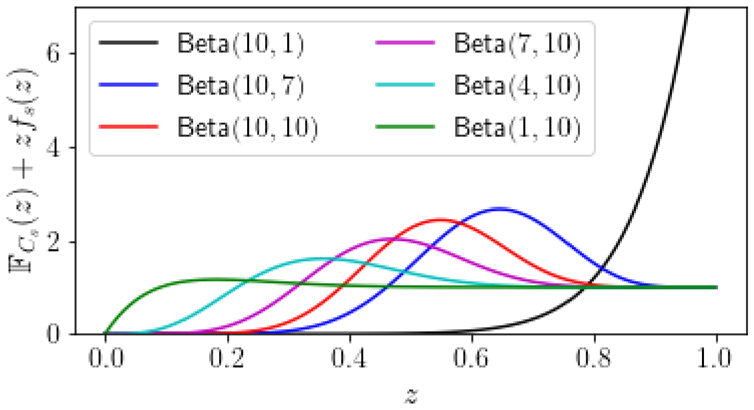
```
7.979591836734694 3.0204081632653064
0.8181818181818182
```



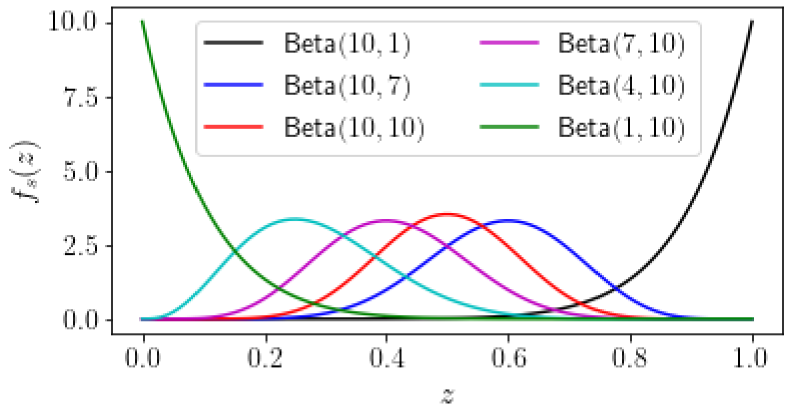
In [6]:
```python
mu0,mu1 = -3,3
G = []
for std0 in np.linspace(mu1*0.5,mu1*5,100):
    std1 = std0
    g0 = norm.cdf(0,mu0,std0) - norm.cdf(0,mu1,std1)
    G.append(g0)
x = np.linspace(0.5,5,100)

plt.figure(figsize=(5,5))
plt.plot(x,G,'b')
plt.ylabel(r'$\max_{\theta} \ \Delta(\theta)$')
plt.xticks([1,2,3,4,5],[r'$\mu$',r'$2\mu$',r'$3\mu$',r'$4\mu$',r'$5\mu$'])

plt.xlabel(r'$\sigma$')
plt.show()
```



In [104]:
```python
loc, scale = 0,1
x1 = np.linspace(0,1,100)
a,b = 10,1
Phi_1 = beta.cdf(x1, a, b, loc, scale) + x1*beta.pdf(x1, a, b, loc, scale)
a,b = 10,4
Phi_2 = beta.cdf(x1, a, b, loc, scale) + x1*beta.pdf(x1, a, b, loc, scale)
a,b = 10,7
Phi_3 = beta.cdf(x1, a, b, loc, scale) + x1*beta.pdf(x1, a, b, loc, scale)
a,b = 10,11
Phi_4 = beta.cdf(x1, a, b, loc, scale) + x1*beta.pdf(x1, a, b, loc, scale)
a,b = 10,10
Phi_5 = beta.cdf(x1, a, b, loc, scale) + x1*beta.pdf(x1, a, b, loc, scale)
a,b = 9,10
Phi_6 = beta.cdf(x1, a, b, loc, scale) + x1*beta.pdf(x1, a, b, loc, scale)
a,b = 7,10
Phi_7 = beta.cdf(x1, a, b, loc, scale) + x1*beta.pdf(x1, a, b, loc, scale)
a,b = 4,10
Phi_8 = beta.cdf(x1, a, b, loc, scale) + x1*beta.pdf(x1, a, b, loc, scale)
a,b = 1,10
Phi_9 = beta.cdf(x1, a, b, loc, scale) + x1*beta.pdf(x1, a, b, loc, scale)

plt.figure(figsize=(6,3))
plt.plot(x1,Phi_1,'k',label=r'Beta$(10,1)$')
plt.plot(x1,Phi_3,'b',label=r'Beta$(10,7)$')
plt.plot(x1,Phi_5,'r',label=r'Beta$(10,10)$')
plt.plot(x1,Phi_7,'m',label=r'Beta$(7,10)$')
plt.plot(x1,Phi_8,'c',label=r'Beta$(4,10)$')
plt.plot(x1,Phi_9,'g',label=r'Beta$(1,10)$')

plt.legend()
plt.ylim([0,7])
plt.xlabel(r'$z$',fontsize=16)
plt.ylabel(r'$\mathbb{F}_{C_s}(z)+zf_s(z)$',fontsize=16)
plt.legend(loc='upper left',ncol=2)
plt.savefig('beta_Phi.pdf',bbox_inches='tight')
plt.show()
```



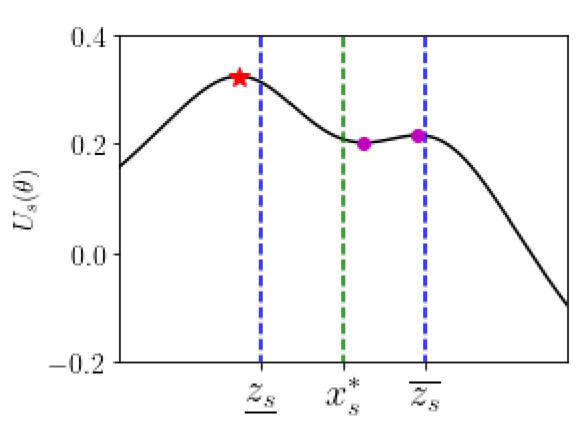
In [105]:
```
from scipy.stats import norm,lognorm,beta
import numpy as np
import matplotlib.pyplot as plt
loc, scale = 0,1
x1 = np.linspace(0,1,100)
a,b = 10,1
Phi_1 =  beta.pdf(x1, a, b, loc, scale)
a,b = 10,7
Phi_2 =  beta.pdf(x1, a, b, loc, scale)
a,b = 10,10
Phi_3 =  beta.pdf(x1, a, b, loc, scale)
a,b = 7,10
Phi_4 =  beta.pdf(x1, a, b, loc, scale)
a,b = 4,10
Phi_5 =  beta.pdf(x1, a, b, loc, scale)
a,b = 1,10
Phi_6 =  beta.pdf(x1, a, b, loc, scale)

plt.figure(figsize=(6,3))
plt.plot(x1,Phi_1,'k',label=r'Beta$(10,1)$')
plt.plot(x1,Phi_2,'b',label=r'Beta$(10,7)$')
plt.plot(x1,Phi_3,'r',label=r'Beta$(10,10)$')
plt.plot(x1,Phi_4,'m',label=r'Beta$(7,10)$')
plt.plot(x1,Phi_5,'c',label=r'Beta$(4,10)$')
plt.plot(x1,Phi_6,'g',label=r'Beta$(1,10)$')

plt.legend()
plt.xlabel(r'$z$',fontsize=16)
plt.ylabel(r'$f_s(z)$',fontsize=16)
plt.legend(loc='upper center',ncol=2)

plt.savefig('beta_pdf.pdf',bbox_inches='tight')
plt.show()
```



institute's utility function

In [63]:
```python
plt.rc('text', usetex=True)

u_cost, u_benefit = 1,1
theta = np.linspace(-20,20,400)
alpha = 0.6
a,b = 10,4
mu0,mu1 = -5,5
std0,std1 = 4.7,4.7
g0 = norm.cdf(theta,mu0,std0) - norm.cdf(theta,mu1,std1)
rho = u_cost*(1-alpha)/(u_benefit*alpha)
U = norm.cdf(theta,mu0,std0)*(1-beta.cdf(g0,a,b,0,1))*rho - norm.cdf(theta,mu1,std1)*(1-rho*beta.cdf(
g0,a,b,0,1))

Id_max = np.argmax(U)
Id_max1 = np.argmax(U[200:])
Id_min = np.argmin(U[Id_max:Id_max1+200])

Phi = beta.cdf(g0,a,b,0,1) + g0*beta.pdf(g0,a,b,0,1)
l = np.abs(Phi-(1/rho))
Id = np.where(l<0.05)
print(theta[Id])

plt.figure(figsize=(4,3))
plt.plot(theta,U,'k')
plt.plot([theta[Id]]*100,np.linspace(-0.4,0.5,100),'b--')
plt.plot([0]*100,np.linspace(-0.4,0.5,100),'g--')

plt.plot(theta[Id_max],U[Id_max],'*',color='r',markersize=10)

plt.plot(theta[Id_max1+200],U[Id_max1+200],'o', color='m')

plt.plot(theta[Id_min+Id_max],U[Id_min+Id_max],'o',color='m')

plt.xticks([theta[Id][0],theta[Id][1],0],[r'$\underline{z_s}$',r'$\overline{z_s}$',r'$x^*_s$'], fonts
ize=20 )
plt.ylabel(r'$U_s(\theta)$')
plt.ylim([-0.2,0.4])
plt.xlim([-8.3,8.3])
plt.savefig('utility.eps',bbox_inches='tight')
plt.show()
```

```
[-3.05764411  3.05764411]
```

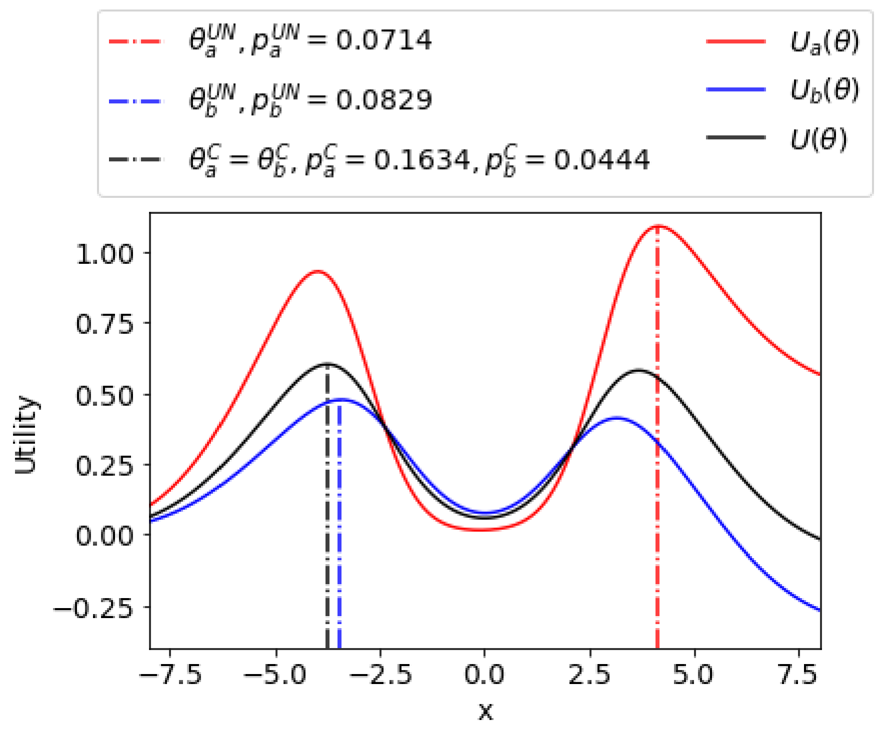


1. when $U_a(\theta)$, $U_b(\theta)$Ua(θ),Ub(θ) both have multiple extreme points

(1). an example when $\alpha_a < \frac{u_-}{u_-+u_+}$αa<u−u−+u+, $\alpha_b > \frac{u_-}{u_-+u_+}$αb>u−u−+u+, the EqOpt fair threholds are decreased for both groups, $p_b^{eqopt} < p_b^{un}$pbeqopt<pbun and $p_a^{eqopt} > p_a^{un}$paeqopt>paun.

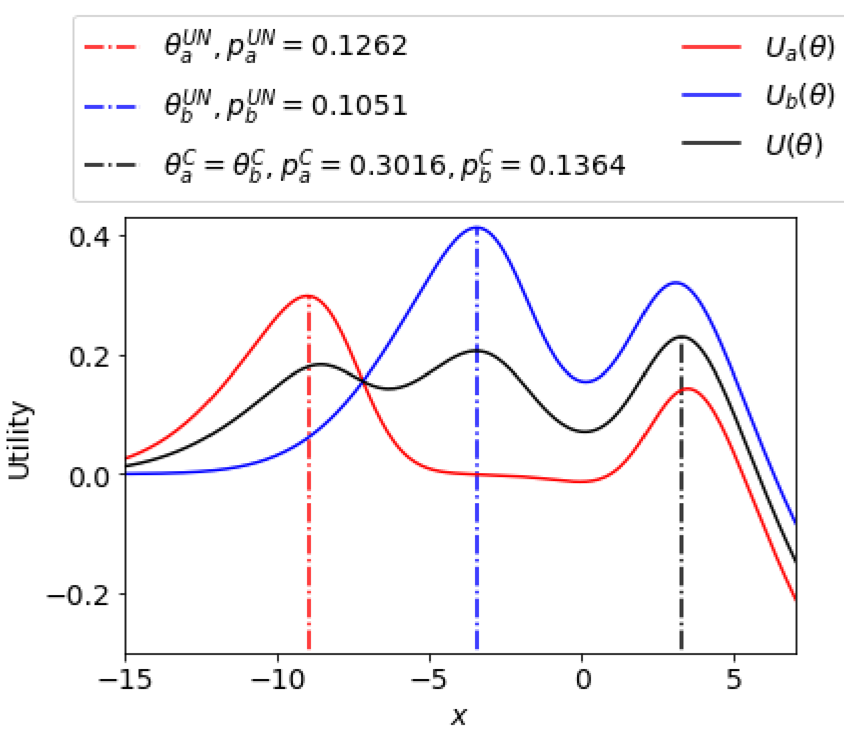
In [3]:

```python
from scipy.stats import norm,lognorm,beta
import matplotlib.pyplot as plt
import numpy as np
import matplotlib
matplotlib.rcParams.update({'font.size': 14})


mu0,mu1 = -5,5
std0,std1 = 2,2
theta = np.linspace(-20,20,400)
g0 = norm.cdf(theta,mu0,std0) - norm.cdf(theta,mu1,std1)
u_cost,u_benefit = 1,1
pa = 0.3

alpha = 0.4
a,b = 10,2
rho = u_cost*(1-alpha)/(u_benefit*alpha)
Ua = norm.cdf(theta,mu0,std0)*(1-beta.cdf(g0,a,b,0,1))*rho - norm.cdf(theta,mu1,std1)*(1-rho*beta.cdf(g0,a,b,0,1))
alpha = 0.6
a,b = 10,1
rho = u_cost*(1-alpha)/(u_benefit*alpha)
Ub = norm.cdf(theta,mu0,std0)*(1-beta.cdf(g0,a,b,0,1))*rho - norm.cdf(theta,mu1,std1)*(1-rho*beta.cdf(g0,a,b,0,1))
theta_a_opt = theta[int(np.argmax(Ua))]
theta_b_opt = theta[int(np.argmax(Ub))]
U_tot = pa*Ua + (1-pa)*Ub
theta_eqopt = theta[int(np.argmax(U_tot))]


g_eqopt = norm.cdf(theta_eqopt,mu0,std0) - norm.cdf(theta_eqopt,mu1,std1)
gb = norm.cdf(theta_b_opt,mu0,std0) - norm.cdf(theta_b_opt,mu1,std1)
ga = norm.cdf(theta_a_opt,mu0,std0) - norm.cdf(theta_a_opt,mu1,std1)


plt.plot([theta_a_opt]*100,np.linspace(-0.4,np.max(Ua),100),'r-.',label=r'$\theta^{UN}_a, p^{UN}_a = 0.0714$')
plt.plot([theta_b_opt]*100,np.linspace(-0.4,np.max(Ub),100),'b-.',label=r'$\theta^{UN}_b, p^{UN}_b = 0.0829$')
plt.plot([theta_eqopt]*100,np.linspace(-0.4,np.max(U_tot),100),'k-.',label=r'$\theta^{C}_a = \theta^{C}_b, p^{C}_a = 0.1634, p^{C}_b = 0.0444$')

plt.plot(theta,Ua,'r',label=r'$U_a(\theta)$')
plt.plot(theta,Ub,'b',label=r'$U_b(\theta)$')
plt.plot(theta,U_tot,'k',label=r'$U(\theta)$')
plt.xlim([-8,8])
plt.ylim([-0.4,1.14])
plt.legend(loc='upper right', bbox_to_anchor=(1.1, 1.5),ncol=2)
plt.xlabel('x')
plt.ylabel('Utility')
plt.show()
```



(2). an example when $\alpha_a > \frac{u_-}{u_-+u_+}$αa>u−u−+u+, $\alpha_b > \frac{u_-}{u_-+u_+}$αb>u−u−+u+, the EqOpt fair threholds are increased for both groups, $p_b^{eqopt} > p_b^{un}$ pbeqopt>pbun and $p_a^{eqopt} > p_a^{un}$ paeqopt>paun.

In [2]:
```
pa = 0.5

alpha = 0.6
a,b = 10,2
mu0b,mu1b = -5,5
std0b,std1b = 3,3
g0b = norm.cdf(theta,mu0b,std0b) - norm.cdf(theta,mu1b,std1b)
rho = u_cost*(1-alpha)/(u_benefit*alpha)
Ub = norm.cdf(theta,mu0b,std0b)*(1-beta.cdf(g0b,a,b,0,1))*rho - norm.cdf(theta,mu1b,std1b)*(1-rho*bet
a.cdf(g0b,a,b,0,1))

alpha = 0.65
a,b = 10,3
mu0a,mu1a = -10,5
std0a,std1a = 3,3
g0a = norm.cdf(theta,mu0a,std0a) - norm.cdf(theta,mu1a,std1a)
rho = u_cost*(1-alpha)/(u_benefit*alpha)
Ua = norm.cdf(theta,mu0a,std0a)*(1-beta.cdf(g0a,a,b,0,1))*rho - norm.cdf(theta,mu1a,std1a)*(1-rho*bet
a.cdf(g0a,a,b,0,1))
theta_a_opt = theta[int(np.argmax(Ua))]
theta_b_opt = theta[int(np.argmax(Ub))]
U_tot = pa*Ua + (1-pa)*Ub
theta_eqopt = theta[int(np.argmax(U_tot))]

g_eqopta = norm.cdf(theta_eqopt,mu0a,std0a) - norm.cdf(theta_eqopt,mu1a,std1a)
g_eqoptb = norm.cdf(theta_eqopt,mu0b,std0b) - norm.cdf(theta_eqopt,mu1b,std1b)

gb = norm.cdf(theta_b_opt,mu0b,std0b) - norm.cdf(theta_b_opt,mu1b,std1b)
ga = norm.cdf(theta_a_opt,mu0a,std0a) - norm.cdf(theta_a_opt,mu1a,std1a)

plt.plot([theta_a_opt]*100,np.linspace(-0.4,np.max(Ua),100),'r-.',label=r'$\theta^{UN}_a, p^{UN}_a =
 0.1262$')
plt.plot([theta_b_opt]*100,np.linspace(-0.4,np.max(Ub),100),'b-.',label=r'$\theta^{UN}_b, p^{UN}_b =
 0.1051$')
plt.plot([theta_eqopt]*100,np.linspace(-0.4,np.max(U_tot),100),'k-.',label=r'$\theta^{C}_a = \theta^
{C}_b, p^{C}_a = 0.3016, p^{C}_b = 0.1364$')

plt.plot(theta,Ua,'r',label=r'$U_a(\theta)$')
plt.plot(theta,Ub,'b',label=r'$U_b(\theta)$')
plt.plot(theta,U_tot,'k',label=r'$U(\theta)$')
plt.xlim([-15,7])
plt.ylim([-0.3,0.43])
plt.legend(loc='upper right', bbox_to_anchor=(1.1, 1.5),ncol=2)
plt.xlabel(r'$x$')
plt.ylabel('Utility')
plt.show()
```



Fair optimal thresholds under strategic manipulation

1. EqOpt fairness: $\int_{-\infty}^{\theta_a} P_{X|Y,S}(x|1,a)dx = \int_{-\infty}^{\theta_b} P_{X|Y,S}(x|1,b)dx$∫−∞θaPX|Y,S(x|1,a)dx=∫−∞θbPX|Y,S(x|1,b)dx

In [8]:
```
# feasible pairs (x,f_eqopt(x)) that satisfy EqOpt
def f_eqopt(x,alpha_a,alpha_b,mu0_a,mu1_a,sigma0_a,sigma1_a,mu0_b,mu1_b,sigma0_b,sigma1_b):
    return norm.ppf(norm.cdf(x, loc=mu1_a, scale=sigma1_a),loc=mu1_b, scale=sigma1_b)

def Peqopt_s(x,alpha_s,mu1_s,sigma1_s,mu0_s,sigma0_s):
    return norm.pdf(x, loc=mu1_s, scale=sigma1_s)
```

2. DP fairness:
$\int_{-\infty}^{\theta_a} \alpha^a P_{X|Y,S}(x|1,a) + (1-\alpha^a)P_{X|Y,S}(x|0,a)dx = \int_{-\infty}^{\theta_b} \alpha^b P_{X|Y,S}(x|1,b) + (1-\alpha^b)P_{X|Y,S}(x|0,b)dx$ ∫−∞θaαaPX|Y

In [9]:
```
# feasible pairs (x,f_dp(x)) that satisfy DP
def f_dp(x,alpha_a,alpha_b,mu0_a,mu1_a,sigma0_a,sigma1_a,mu0_b,mu1_b,sigma0_b,sigma1_b):
    f_b = (lambda x_b: alpha_b*norm.cdf(x_b, loc=mu1_b, scale=sigma1_b)+(1-alpha_b)*norm.cdf(x_b,loc=mu0_b, scale=sigma0_b))
    inv_f_b = inversefunc(f_b)
    return float(inv_f_b(alpha_a*norm.cdf(x, loc=mu1_a, scale=sigma1_a)+(1-alpha_a)*norm.cdf(x, loc=mu0_a, scale=sigma0_a)))

def Pdp_s(x,alpha_s,mu1_s,sigma1_s,mu0_s,sigma0_s):
    return alpha_s*norm.pdf(x, loc=mu1_s, scale=sigma1_s)+(1-alpha_s)*norm.pdf(x, loc=mu0_s, scale=sigma0_s)
```

Optimal equation for finding the optimal thresholds

In [7]:
```
from scipy.stats import norm,beta
import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import newton
from pynverse import inversefunc
import warnings
warnings.filterwarnings("ignore")

def Phi(x,beta_a,beta_b,mu0,mu1,sigma0,sigma1):
    g = norm.cdf(x, loc=mu0, scale=sigma0) - norm.cdf(x, loc=mu1, scale=sigma1)
    return beta.cdf(g,beta_a,beta_b,0,1) + g*beta.pdf(g,beta_a,beta_b,0,1)


# optimal equations
def dir_Us(x,alpha,cost,benefit,beta_a,beta_b,mu0,mu1,sigma0,sigma1,Pc_s):
    term1 = (norm.pdf(x, loc=mu1, scale=sigma1) - norm.pdf(x, loc=mu0, scale=sigma0))/Pc_s(x,alpha,mu1,sigma1,mu0,sigma0)
    term2 = (norm.pdf(x, loc=mu0, scale=sigma0)*cost*(1-alpha) - norm.pdf(x, loc=mu1, scale=sigma1)*benefit*alpha)/Pc_s(x,alpha,mu1,sigma1,mu0,sigma0)
    return term1*Phi(x,beta_a,beta_b,mu0,mu1,sigma0,sigma1)*cost*(1-alpha) + term2



def opt_eqn(x_a,alpha_a,alpha_b,Pa,f_c,cost,benefit,beta_a_a,beta_b_a,mu0_a,mu1_a,sigma0_a,sigma1_a,beta_a_b,beta_b_b,mu0_b,mu1_b,sigma0_b,sigma1_b,Pc_s):
    x_b = f_c(x_a,alpha_a,alpha_b,mu0_a,mu1_a,sigma0_a,sigma1_a,mu0_b,mu1_b,sigma0_b,sigma1_b)
    return Pa*dir_Us(x_a,alpha_a,cost,benefit,beta_a_a,beta_b_a,mu0_a,mu1_a,sigma0_a,sigma1_a,Pc_s) + (1-Pa)*dir_Us(x_b,alpha_b,cost,benefit,beta_a_b,beta_b_b,mu0_b,mu1_b,sigma0_b,sigma1_b,Pc_s)



def get_policy(f_c,alpha_a,alpha_b,Pa,cost,benefit,beta_a_a,beta_b_a,mu0_a,mu1_a,sigma0_a,sigma1_a,beta_a_b,beta_b_b,mu0_b,mu1_b,sigma0_b,sigma1_b,Pc_s):
    root=[]
    range_min,range_max = min(mu0_a,mu0_b)-max(sigma0_a,sigma0_b,sigma1_a,sigma1_b),max(mu1_a,mu1_b)+max(sigma0_a,sigma0_b,sigma1_a,sigma1_b)
    for i in np.arange(range_min,range_max,1):
        try:
            root.append(newton(opt_eqn,x0 = i,maxiter=50,args = (alpha_a,alpha_b,Pa,f_c,cost,benefit,beta_a_a,beta_b_a,mu0_a,mu1_a,sigma0_a,sigma1_a,beta_a_b,beta_b_b,mu0_b,mu1_b,sigma0_b,sigma1_b,Pc_s)))
        except(RuntimeError):
            pass
    root = [float(format(round(r, 4))) for r in root]

    return np.unique(root)[0],f_c(np.unique(root)[0],alpha_a,alpha_b,mu0_a,mu1_a,sigma0_a,sigma1_a,mu0_b,mu1_b,sigma0_b,sigma1_b)

# note that above only finds the fisrt extreme point
```

$U_a(\theta), U_b(\theta)$ Ua(θ),Ub(θ) both have unique extreme point

1. $\alpha_a > \frac{u_+}{u_- + u_+} > \alpha_b$ αa>u+u−+u+>αb

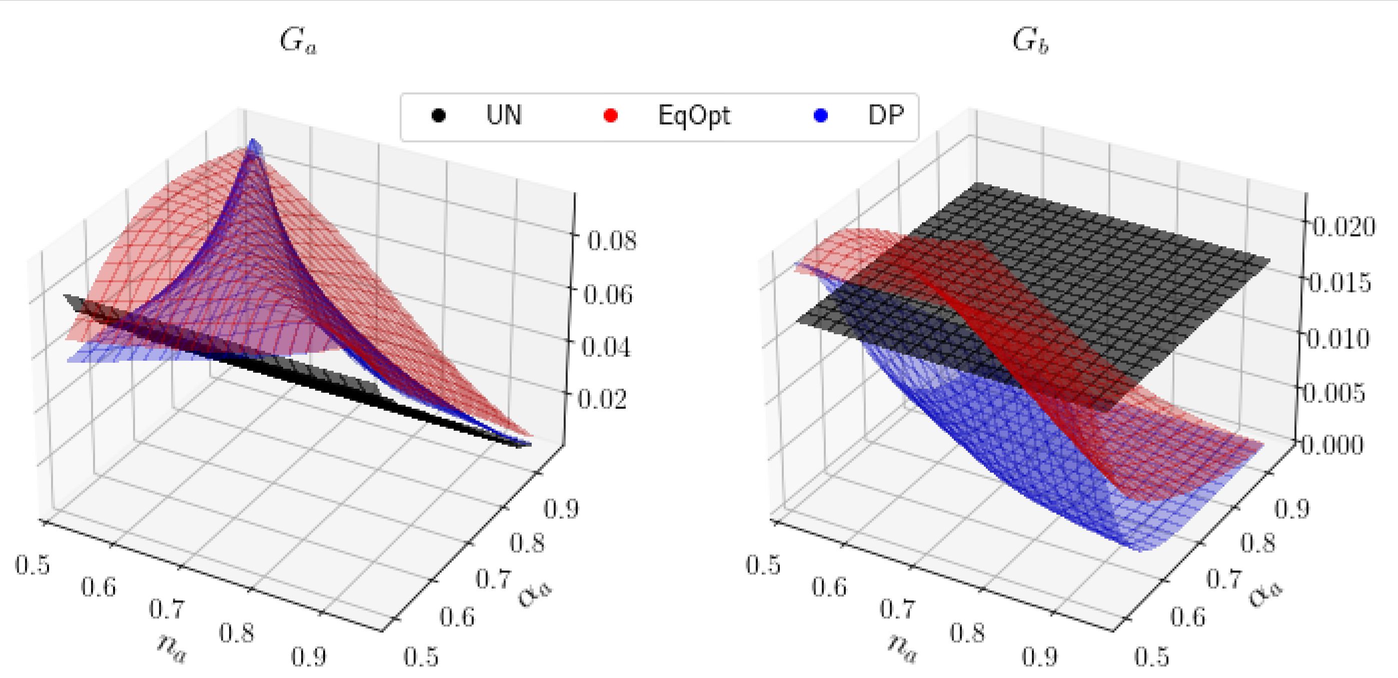
In [5]:

```python
def fun(theta,pa,u_cost,u_benefit,mu0_b,mu1_b,sigma0_b,sigma1_b,alpha_b,a_b,b_b,mu0_a,mu1_a,sigma0_a,sigma1_a,alpha_a,a_a,b_a):

    g0_b = norm.cdf(theta,mu0_b,sigma0_b) - norm.cdf(theta,mu1_b,sigma1_b)
    rho_b = u_cost*(1-alpha_b)/(u_benefit*alpha_b)
    Ub = norm.cdf(theta,mu0_b,sigma0_b)*(1-beta.cdf(g0_b,a_b,b_b,0,1))*rho_b - norm.cdf(theta,mu1_b,sigma1_b)*(1-rho_b*beta.cdf(g0_b,a_b,b_b,0,1))

    g0_a = norm.cdf(theta,mu0_a,sigma0_a) - norm.cdf(theta,mu1_a,sigma1_a)
    rho_a = u_cost*(1-alpha_a)/(u_benefit*alpha_a)
    Ua = norm.cdf(theta,mu0_a,sigma0_a)*(1-beta.cdf(g0_a,a_a,b_a,0,1))*rho_a - norm.cdf(theta,mu1_a,sigma1_a)*(1-rho_a*beta.cdf(g0_a,a_a,b_a,0,1))

    # total utility
    U = Ub*(1-pa)+Ua*pa


    # unconstrained optimal policies
    theta_opt_a = theta[int(np.argmax(Ua))]
    theta_opt_b = theta[int(np.argmax(Ub))]

    # EqOpt fair thresholds
    theta_eqopt_a,theta_eqopt_b = get_policy(f_eqopt,alpha_a,alpha_b,pa,u_cost,u_benefit,a_a,b_a,mu0_a,mu1_a,sigma0_a,sigma1_a,a_b,b_b,mu0_b,mu1_b,sigma0_b,sigma1_b,Peqopt_s)

    # DP fair thresholds
    theta_dp_a,theta_dp_b = get_policy(f_dp,alpha_a,alpha_b,pa,u_cost,u_benefit,a_a,b_a,mu0_a,mu1_a,sigma0_a,sigma1_a,a_b,b_b,mu0_b,mu1_b,sigma0_b,sigma1_b,Pdp_s)

    # consequent manipulation probability
    p_opt_a = beta.cdf(norm.cdf(theta_opt_a,mu0_a,sigma0_a) - norm.cdf(theta_opt_a,mu1_a,sigma1_a),a_a,b_a,0,1)
    p_opt_b = beta.cdf(norm.cdf(theta_opt_b,mu0_b,sigma0_b) - norm.cdf(theta_opt_b,mu1_b,sigma1_b),a_b,b_b,0,1)

    p_eqopt_a = beta.cdf(norm.cdf(theta_eqopt_a,mu0_a,sigma0_a) - norm.cdf(theta_eqopt_a,mu1_a,sigma1_a),a_a,b_a,0,1)
    p_eqopt_b = beta.cdf(norm.cdf(theta_eqopt_b,mu0_b,sigma0_b) - norm.cdf(theta_eqopt_b,mu1_b,sigma1_b),a_b,b_b,0,1)

    p_dp_a = beta.cdf(norm.cdf(theta_dp_a,mu0_a,sigma0_a) - norm.cdf(theta_dp_a,mu1_a,sigma1_a),a_a,b_a,0,1)
    p_dp_b = beta.cdf(norm.cdf(theta_dp_b,mu0_b,sigma0_b) - norm.cdf(theta_dp_b,mu1_b,sigma1_b),a_b,b_b,0,1)

    theta_a = [theta_opt_a,theta_eqopt_a,theta_dp_a]
    theta_b = [theta_opt_b,theta_eqopt_b,theta_dp_b]

    p_a = [p_opt_a,p_eqopt_a,p_dp_a]
    p_b = [p_opt_b,p_eqopt_b,p_dp_b]

    return theta_a,theta_b,p_a,p_b
```

In [10]:

```python
from scipy.stats import norm,beta
import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import newton
from pynverse import inversefunc
import warnings
warnings.filterwarnings("ignore")

theta = np.linspace(-20,20,400)
u_cost,u_benefit = 1,1
# G_b
mu0_b,mu1_b = -5,5
sigma0_b,sigma1_b = 5,5
alpha_b = 0.4
a_b,b_b = 10,1

# G_a
mu0_a,mu1_a = -5,5
sigma0_a,sigma1_a = 4,4
a_a,b_a = 10,1 # 10,2 for multiple

step = 20
p_opt_a,p_eqopt_a,p_dp_a = np.zeros([step,step]),np.zeros([step,step]),np.zeros([step,step])
p_opt_b,p_eqopt_b,p_dp_b = np.zeros([step,step]),np.zeros([step,step]),np.zeros([step,step])

i = -1
for alpha_a in np.linspace(0.52,0.95,step):
    i+=1
    j = -1
    for pa in np.linspace(u_benefit/(u_benefit+u_cost),0.95,step):
        j+=1
        theta_a,theta_b,p_a,p_b = fun(theta,pa,u_cost,u_benefit,mu0_b,mu1_b,sigma0_b,sigma1_b,alpha_b,a_b,b_b,mu0_a,mu1_a,sigma0_a,sigma1_a,alpha_a,a_a,b_a)
        p_opt_a[i,j],p_eqopt_a[i,j],p_dp_a[i,j] = p_a
        p_opt_b[i,j],p_eqopt_b[i,j],p_dp_b[i,j] = p_b
```

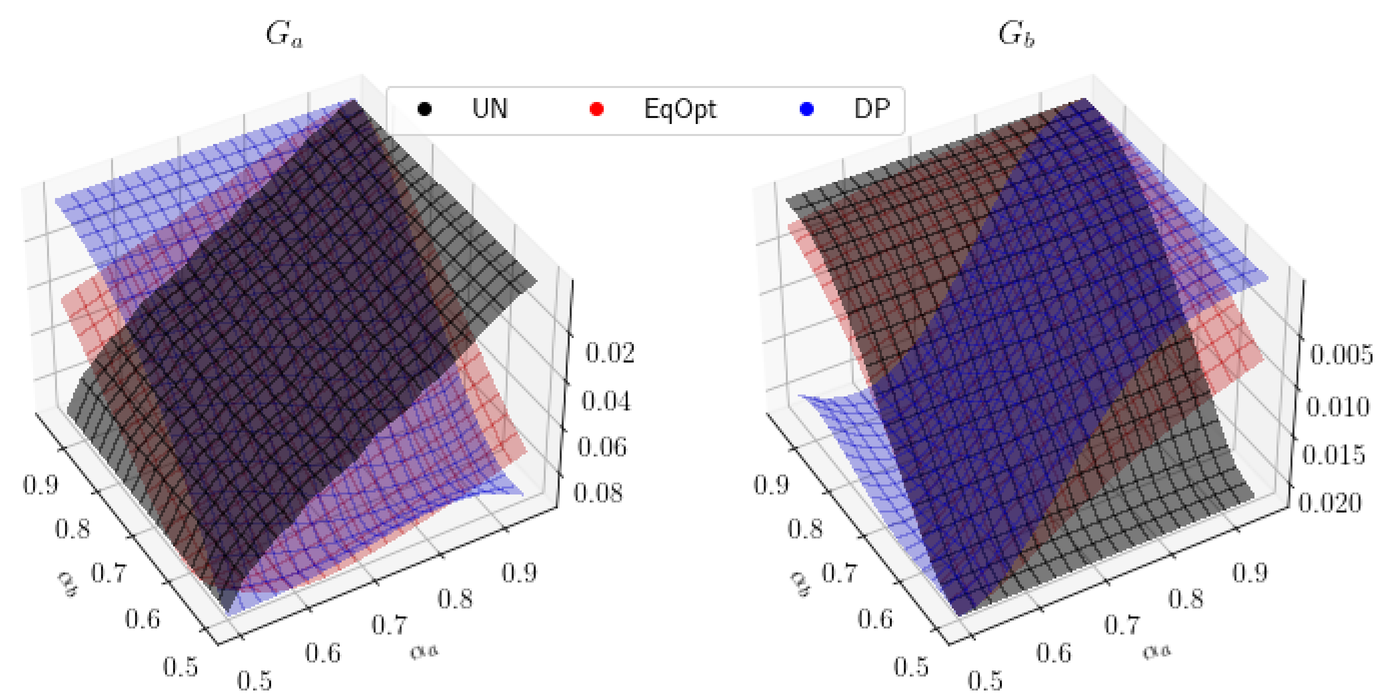
```
In [120]: from matplotlib import cm
from matplotlib.ticker import LinearLocator

step = 20
fig = plt.figure(figsize=(12,6))
ax1 = fig.add_subplot(121, projection='3d')
ax2 = fig.add_subplot(122, projection='3d')

X = np.linspace(0.52,0.95,step)
Y = np.linspace(u_benefit/(u_benefit+u_cost),0.95,step)
X, Y = np.meshgrid(X, Y)

surf = ax1.plot_surface(X, Y, p_opt_a, color='black',linewidth=0, antialiased=False,alpha=0.6,label=
'UN')
surf1 = ax1.plot_surface(X, Y, p_eqopt_a, color='red',linewidth=0, antialiased=False,alpha=0.3,label=
'EqOpt')
surf1 = ax1.plot_surface(X, Y, p_dp_a, color='blue',linewidth=0, antialiased=False,alpha=0.3,label='D
P')


surf2 = ax2.plot_surface(X, Y, p_opt_b,color='black',linewidth=0, antialiased=False,alpha=0.6)
surf3 = ax2.plot_surface(X, Y, p_eqopt_b,color='red',linewidth=0, antialiased=False,alpha=0.3)
surf3 = ax2.plot_surface(X, Y, p_dp_b, 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'$\alpha_a$',fontsize=18)
ax1.set_xlabel(r'$n_a$',fontsize=18)
ax2.set_ylabel(r'$\alpha_a$',fontsize=18)
ax2.set_xlabel(r'$n_a$',fontsize=18)
plt.savefig('manipulation_case1.pdf',bbox_inches='tight')

plt.show()
```



2 $\alpha_s > \frac{u_+}{u_-+u_+}$αs>u+u−+u+ or $\alpha_s < \frac{u_+}{u_-+u_+}$αs<u+u−+u+

```
In [12]: alpha_b = 0.8
alpha_a = 0.53
pa = 0.5

step = 20
p_opt_a,p_eqopt_a,p_dp_a = np.zeros([step,step]),np.zeros([step,step]),np.zeros([step,step])
p_opt_b,p_eqopt_b,p_dp_b = np.zeros([step,step]),np.zeros([step,step]),np.zeros([step,step])

i = -1
for alpha_a in np.linspace(u_benefit/(u_benefit+u_cost),0.95,step):
#for alpha_a in np.linspace(0.05,u_benefit/(u_benefit+u_cost),step):
    i+=1
    j = -1
    for alpha_b in np.linspace(u_benefit/(u_benefit+u_cost),0.95,step):
    #for alpha_b in np.linspace(0.05,u_benefit/(u_benefit+u_cost),step):
        j+=1
        theta_a,theta_b,p_a,p_b = fun(theta,pa,u_cost,u_benefit,mu0_b,mu1_b,sigma0_b,sigma1_b,alpha_b
,a_b,b_b,mu0_a,mu1_a,sigma0_a,sigma1_a,alpha_a,a_a,b_a)
        p_opt_a[i,j],p_eqopt_a[i,j],p_dp_a[i,j] = p_a
        p_opt_b[i,j],p_eqopt_b[i,j],p_dp_b[i,j] = p_b
```

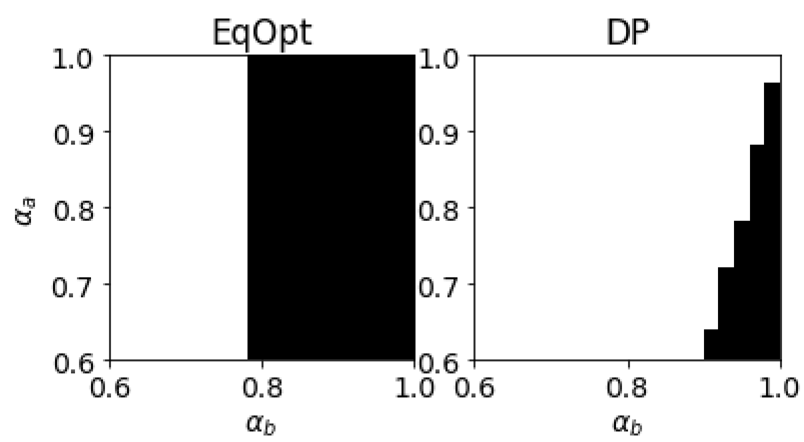
In [191]:
```
fig = plt.figure(figsize=(12,6))
ax1 = fig.add_subplot(121, projection='3d')
ax2 = fig.add_subplot(122, projection='3d')

X = np.linspace(u_benefit/(u_benefit+u_cost),0.95,step)
Y = np.linspace(u_benefit/(u_benefit+u_cost),0.95,step)
#X = np.linspace(0.05,u_benefit/(u_benefit+u_cost),step)
#Y = np.linspace(0.05,u_benefit/(u_benefit+u_cost),step)

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

surf = ax1.plot_surface(X, Y, p_opt_a, color='black',linewidth=0, antialiased=False,alpha=0.5,label=
'UN')
surf1 = ax1.plot_surface(X, Y, p_eqopt_a, color='red',linewidth=0, antialiased=False,alpha=0.3,label=
'EqOpt')
surf1 = ax1.plot_surface(X, Y, p_dp_a, color='blue',linewidth=0, antialiased=False,alpha=0.3,label='D
P')


surf2 = ax2.plot_surface(X, Y, p_opt_b,color='black',linewidth=0, antialiased=False,alpha=0.5)
surf3 = ax2.plot_surface(X, Y, p_eqopt_b,color='red',linewidth=0, antialiased=False,alpha=0.3)
surf3 = ax2.plot_surface(X, Y, p_dp_b, 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'$\alpha_a$')
ax1.set_xlabel(r'$\alpha_b$')
ax2.set_ylabel(r'$\alpha_a$')
ax2.set_xlabel(r'$\alpha_b$')

ax1.view_init(-140, 30)
ax2.view_init(-140, 30)

plt.show()
```



disincentivize both groups

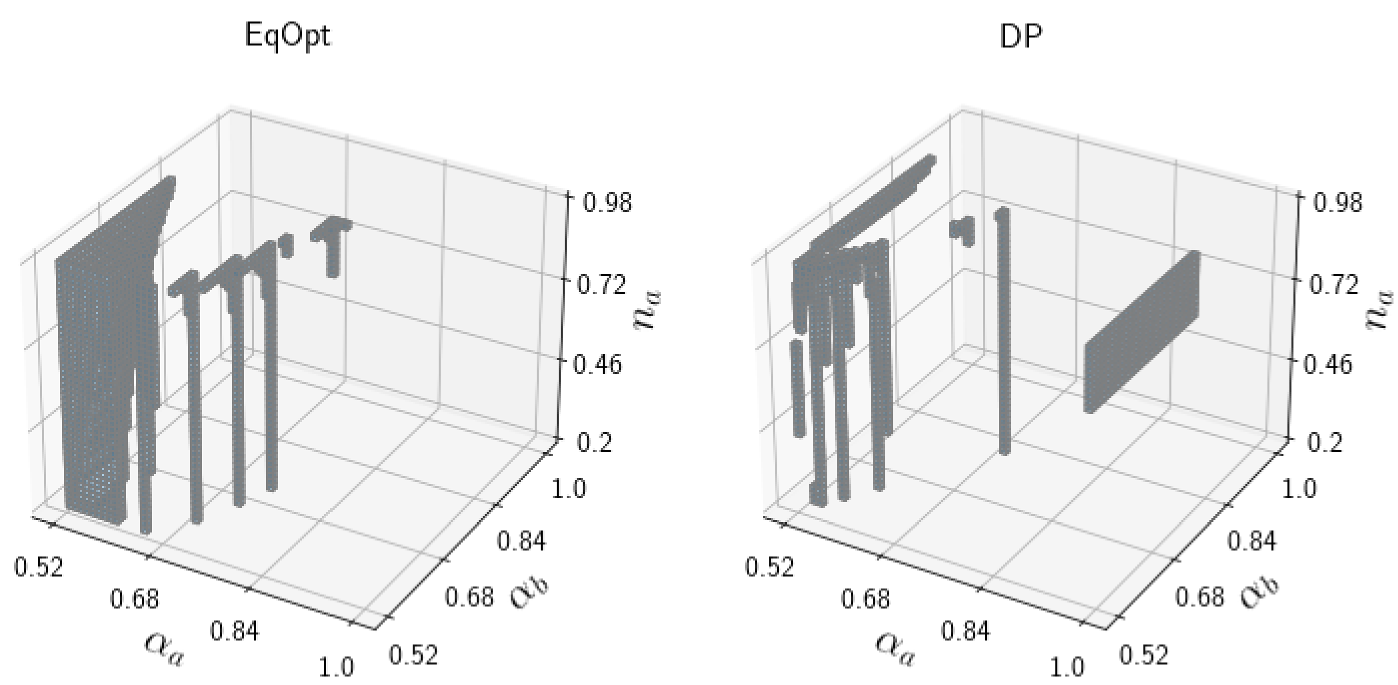
In [14]:
```
def checkFun(u_cost,u_benefit,alpha_a,alpha_b,mu0_b,sigma0_b,mu1_b,sigma1_b,a_b,b_b):

    theta = np.linspace(-20,20,400)
    # check the condition holds or not, only suitable for x_s^* = 0
    g0_b = norm.cdf(theta,mu0_b,sigma0_b) - norm.cdf(theta,mu1_b,sigma1_b)
    rho_b = u_cost*(1-alpha_b)/(u_benefit*alpha_b)
    Ub = norm.cdf(theta,mu0_b,sigma0_b)*(1-beta.cdf(g0_b,a_b,b_b,0,1))*rho_b - norm.cdf(theta,mu1_b,s
igma1_b)*(1-rho_b*beta.cdf(g0_b,a_b,b_b,0,1))
    theta_opt_b = theta[int(np.argmax(Ub))]
    Id = np.argmin(np.abs(g0_b[200:] - (norm.cdf(theta_opt_b,mu0_b,sigma0_b) - norm.cdf(theta_opt_b,m
u1_b,sigma1_b))))
    theta_hat_b = theta[200+Id]

    # EqOpt
    if norm.ppf(norm.cdf(theta_hat_b,mu1_b,sigma1_b),mu1_a,sigma1_a) < 0:
        eqopt_I = 1#print('condition holds for EqOpt')
    else:
        eqopt_I = 0#print('condition does not hold for EqOpt')

    # DP
    f_a = (lambda x_a: alpha_a*norm.cdf(x_a,mu1_a,sigma1_a)+(1-alpha_a)*norm.cdf(x_a,mu0_a,sigma0_a))
    inv_f_a = inversefunc(f_a)
    tmp = float(inv_f_a(alpha_b*norm.cdf(theta_hat_b,mu1_b,sigma1_b)+(1-alpha_b)*norm.cdf(theta_hat_b
,mu0_b,sigma0_b)))
    if tmp < 0:
        dp_I = 1#print('condition holds for DP')
    else:
        dp_I = 0#print('condition does not hold for DP')

    return eqopt_I,dp_I
```

In [17]:
```
u_cost,u_benefit = 1.5,1

# G_b
mu0_b,mu1_b = -5,5
sigma0_b,sigma1_b = 4.5,4.5
alpha_b = 0.67
a_b,b_b = 10,1

# G_a
mu0_a,mu1_a = -2,2
sigma0_a,sigma1_a = 4.5,4.5
a_a,b_a = 10,1
alpha_a = 0.50025

m = 100
eqopt_I,dp_I = np.zeros([m,m]),np.zeros([m,m])
i = -1
for alpha_a in np.linspace(u_cost/(u_cost+u_benefit),1,m):
    i+=1
    j = -1
    for alpha_b in np.linspace(u_cost/(u_cost+u_benefit),1,m):
        j+=1
        eqopt_I[i,j],dp_I[i,j] = checkFun(u_cost,u_benefit,alpha_a,alpha_b,mu0_b,sigma0_b,mu1_b,sigma
1_b,a_b,b_b)

f, axarr = plt.subplots(1,2)
axarr[0].imshow(eqopt_I,origin='lower',extent=[u_cost/(u_cost+u_benefit),1,u_cost/(u_cost+u_benefit),
1],cmap='binary')
axarr[1].imshow(dp_I,origin='lower',extent=[u_cost/(u_cost+u_benefit),1,u_cost/(u_cost+u_benefit),1],
cmap='binary')
axarr[0].set_xlabel(r'$\alpha_b$')
axarr[0].set_ylabel(r'$\alpha_a$')
axarr[1].set_xlabel(r'$\alpha_b$')
axarr[1].set_title('DP')
axarr[0].set_title('EqOpt')
#plt.savefig('corollary2_2.pdf',bbox_inches='tight')
plt.show()
```



In [19]:
```
u_cost,u_benefit = 1.1,1#2,1

# G_b
mu0_b,mu1_b = -5,5
sigma0_b,sigma1_b = 4.5,4.5
a_b,b_b = 10,1

# G_a
mu0_a,mu1_a = -2,2
sigma0_a,sigma1_a = 4.5,4.5
a_a,b_a = 10,1

n_pa = 40
pa_list = list(np.linspace(0.2,0.98,n_pa))

m = 50
eqopt_Pa,dp_Pa = np.zeros([m,m,n_pa]),np.zeros([m,m,n_pa])
i = -1
for alpha_a in np.linspace(u_cost/(u_cost+u_benefit),1,m):
    i+=1
    j = -1
    for alpha_b in np.linspace(u_cost/(u_cost+u_benefit),1,m):
        j+=1
        eqopt_I,dp_I = checkFun(u_cost,u_benefit,alpha_a,alpha_b,mu0_b,sigma0_b,mu1_b,sigma1_b,a_b,b_
b)
        if eqopt_I == 1 or dp_I == 1:
            k=-1
            for pa in pa_list[::-1]:
                k+=1
                _,_,p_a,p_b = fun(theta,pa,u_cost,u_benefit,mu0_b,mu1_b,sigma0_b,sigma1_b,alpha_b,a_b
,b_b,mu0_a,mu1_a,sigma0_a,sigma1_a,alpha_a,a_a,b_a)
                if p_a[1] < p_a[0] and p_b[1] < p_b[0]:
                    eqopt_Pa[i,j,k] = 1
                if p_a[2] < p_a[0] and p_b[2] < p_b[0]:
                    dp_Pa[i,j,k] = 1
                if p_a[1] > p_a[0] or p_b[1] > p_b[0]:
                    if p_a[2] > p_a[0] or p_b[2] > p_b[0]:
                        break
```

In [71]:
```
from mpl_toolkits.mplot3d import Axes3D
def make_ax(grid=False):
    fig = plt.figure()
    ax = fig.gca(projection='3d')
    ax.set_xlabel(r'$\alpha_a$')
    ax.set_ylabel(r'$\alpha_b$')
    ax.set_zlabel(r'$P_S(a)$')
    ax.grid(grid)
    return ax

eqopt_Pa = np.load('eqopt_Pa1.npy')
eqopt_Pa = eqopt_Pa[:, :, ::-1]
dp_Pa = np.load('dp_Pa1.npy')
dp_Pa = dp_Pa[:, :, ::-1]

u_cost,u_benefit = 1.1,1

filled_eqopt = np.array(eqopt_Pa)
filled_dp = np.array(dp_Pa)


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

ax1.voxels(filled_eqopt,facecolors='#1f77b430', edgecolors='gray', shade=False)
ax1.set_xlabel(r'$\alpha_a$',fontsize=22)
ax1.set_ylabel(r'$\alpha_b$',fontsize=22)
ax1.set_zlabel(r'$n_a$',fontsize=22)
ax1.set_xticks(np.linspace(0,50,4))
ax1.set_yticks(np.linspace(0,50,4))
ax1.set_zticks(np.linspace(0,40,4))
ax1.set_xticklabels(np.around(np.linspace(u_cost/(u_cost+u_benefit),1,4),2))
ax1.set_yticklabels(np.around(np.linspace(u_cost/(u_cost+u_benefit),1,4),2))
ax1.set_zticklabels(np.around(np.linspace(0.2,0.98,4),2))
ax1.set_title('EqOpt')

ax2.voxels(filled_dp,facecolors='#1f77b430', edgecolors='gray', shade=False)
ax2.set_xlabel(r'$\alpha_a$',fontsize=22)
ax2.set_ylabel(r'$\alpha_b$',fontsize=22)
ax2.set_zlabel(r'$n_a$',fontsize=22)
ax2.set_xticks(np.linspace(0,50,4))
ax2.set_yticks(np.linspace(0,50,4))
ax2.set_zticks(np.linspace(0,40,4))
ax2.set_xticklabels(np.around(np.linspace(u_cost/(u_cost+u_benefit),1,4),2))
ax2.set_yticklabels(np.around(np.linspace(u_cost/(u_cost+u_benefit),1,4),2))
ax2.set_zticklabels(np.around(np.linspace(0.2,0.98,4),2))
ax2.set_title('DP')

plt.savefig('disincentive1.pdf',bbox_inches='tight')
plt.show()
```



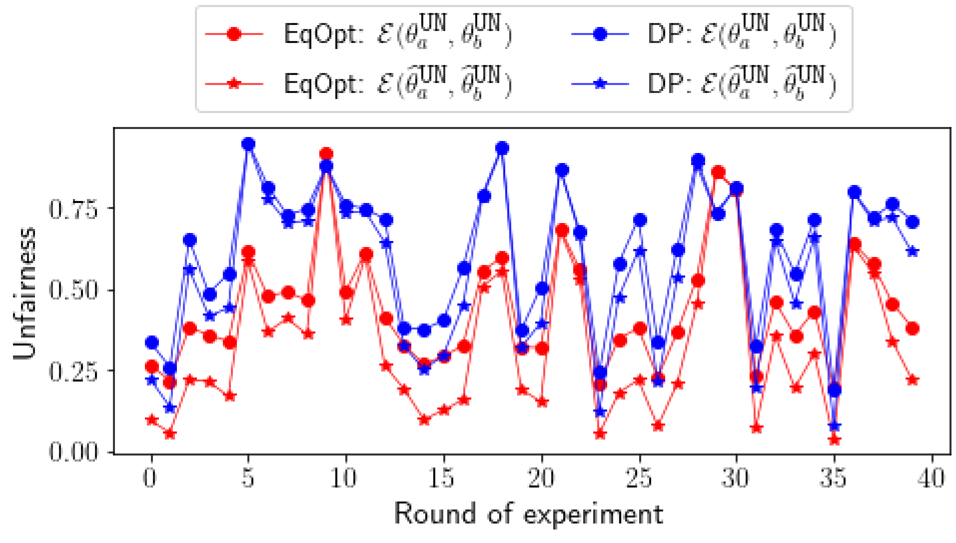
Impact of strategic policies on fairness

In [73]:

```python
def policies(u_cost,u_benefit,mu0,mu1,sigma0,sigma1,alpha,a,b):
    theta = np.linspace(-20,20,400)

    g0 = norm.cdf(theta,mu0,sigma0) - norm.cdf(theta,mu1,sigma1)
    rho = u_cost*(1-alpha)/(u_benefit*alpha)
    U_robust = norm.cdf(theta,mu0,sigma0)*(1-beta.cdf(g0,a,b,0,1))*rho - norm.cdf(theta,mu1,sigma1)*(
1-rho*beta.cdf(g0,a,b,0,1))
    U = norm.cdf(theta,mu0,sigma0)*(1-alpha)*u_cost -  norm.cdf(theta,mu1,sigma1)*alpha*u_benefit
    # strategic policy
    theta_opt_robust = theta[int(np.argmax(U_robust))]

    # non-strategic policy
    theta_opt = theta[int(np.argmax(U))]
    return theta_opt_robust,theta_opt
```

In [87]:

```python
import random
u_cost,u_benefit = 1,1

# G_b
mu0_b,mu1_b = -5,5
sigma0_b,sigma1_b = 5,5
a_b,b_b = 10,3
alpha_b = 0.4

# G_a
mu0_a,mu1_a = -5,5
sigma0_a,sigma1_a = 5,5
a_a,b_a = 10,1
alpha_a = 0.5

i=0
D_eqopt_robust,D_eqopt,D_dp_robust,D_dp=[],[],[],[]
while i < 40:
    i+=1
    alpha_a = random.uniform(u_cost/(u_cost+u_benefit), 1)
    alpha_b = random.uniform(0,u_cost/(u_cost+u_benefit))

    theta_opt_robust_a,theta_opt_a = policies(u_cost,u_benefit,mu0_a,mu1_a,sigma0_a,sigma1_a,alpha_a,
a_a,b_a)
    theta_opt_robust_b,theta_opt_b = policies(u_cost,u_benefit,mu0_b,mu1_b,sigma0_b,sigma1_b,alpha_b,
a_b,b_b)

    # EqOpt disparity
    D_eqopt_robust.append(norm.cdf(theta_opt_robust_b,mu1_b,sigma1_b)-norm.cdf(theta_opt_robust_a,mu1
_a,sigma1_a))
    D_eqopt.append(norm.cdf(theta_opt_b,mu1_b,sigma1_b)-norm.cdf(theta_opt_a,mu1_a,sigma1_a))

    # DP disparity
    D_dp_robust.append(norm.cdf(theta_opt_robust_b,mu1_b,sigma1_b)*alpha_b+ norm.cdf(theta_opt_robust
_b,mu0_b,sigma0_b)*(1-alpha_b)-norm.cdf(theta_opt_robust_a,mu1_a,sigma1_a)*alpha_a- norm.cdf(theta_op
t_robust_a,mu0_a,sigma0_a)*(1-alpha_a))
    D_dp.append(norm.cdf(theta_opt_b,mu1_b,sigma1_b)*alpha_b+ norm.cdf(theta_opt_b,mu0_b,sigma0_b)*(1
-alpha_b)-norm.cdf(theta_opt_a,mu1_a,sigma1_a)*alpha_a- norm.cdf(theta_opt_a,mu0_a,sigma0_a)*(1-alpha
_a))

plt.figure(figsize=(7.5,3))
plt.plot(D_eqopt_robust,'ro-',linewidth=0.7,label=r'EqOpt: $\mathcal{E}(\theta^{\texttt{UN}}_a,\theta
^{\texttt{UN}}_b)$')
plt.plot(D_eqopt,'r*-',linewidth=0.7,label=r'EqOpt: $\mathcal{E}(\widehat{\theta}^{\texttt{UN}}_a,\wi
dehat{\theta}^{\texttt{UN}}_b)$')
plt.plot(D_dp_robust,'bo-',linewidth=0.7,label=r'DP: $\mathcal{E}(\theta^{\texttt{UN}}_a,\theta^{\tex
ttt{UN}}_b)$')
plt.plot(D_dp,'b*-',linewidth=0.7,label=r'DP: $\mathcal{E}(\widehat{\theta}^{\texttt{UN}}_a,\widehat
{\theta}^{\texttt{UN}}_b)$')
plt.ylabel('Unfairness',fontsize = 16)
plt.xlabel('Round of experiment',fontsize = 16)
plt.legend(loc='upper right', bbox_to_anchor=(0.9, 1.42),ncol=2,fontsize = 14)
plt.savefig('unfairness2.pdf',bbox_inches='tight')
plt.show()
```



In [91]:
```
mu0_b,mu1_b = -5,5
sigma0_b,sigma1_b = 5,5
a_b,b_b = 10,1

# G_a
mu0_a,mu1_a = -5,5
sigma0_a,sigma1_a = 5,5
a_a,b_a = 10,5



i=0
D_eqopt_robust,D_eqopt,D_dp_robust,D_dp=[],[],[],[]
while i < 40:
    i+=1
    alpha_a = random.uniform(0,u_cost/(u_cost+u_benefit))
    alpha_b = random.uniform(0,alpha_a)

    theta_opt_robust_a,theta_opt_a = policies(u_cost,u_benefit,mu0_a,mu1_a,sigma0_a,sigma1_a,alpha_a,
a_a,b_a)
    theta_opt_robust_b,theta_opt_b = policies(u_cost,u_benefit,mu0_b,mu1_b,sigma0_b,sigma1_b,alpha_b,
a_b,b_b)

    # EqOpt disparity
    D_eqopt_robust.append(norm.cdf(theta_opt_robust_b,mu1_b,sigma1_b)-norm.cdf(theta_opt_robust_a,mu1
_a,sigma1_a))
    D_eqopt.append(norm.cdf(theta_opt_b,mu1_b,sigma1_b)-norm.cdf(theta_opt_a,mu1_a,sigma1_a))

    # DP disparity
    D_dp_robust.append(norm.cdf(theta_opt_robust_b,mu1_b,sigma1_b)*alpha_b+ norm.cdf(theta_opt_robust
_b,mu0_b,sigma0_b)*(1-alpha_b)-norm.cdf(theta_opt_robust_a,mu1_a,sigma1_a)*alpha_a- norm.cdf(theta_op
t_robust_a,mu0_a,sigma0_a)*(1-alpha_a))
    D_dp.append(norm.cdf(theta_opt_b,mu1_b,sigma1_b)*alpha_b+ norm.cdf(theta_opt_b,mu0_b,sigma0_b)*(1
-alpha_b)-norm.cdf(theta_opt_a,mu1_a,sigma1_a)*alpha_a- norm.cdf(theta_opt_a,mu0_a,sigma0_a)*(1-alpha
_a))

plt.figure(figsize=(7.5,3))
plt.plot(D_eqopt_robust,'ro-',linewidth=0.7,label=r'EqOpt: $\mathcal{E}(\theta^{\texttt{UN}}_a,\theta
^{\texttt{UN}}_b)$')
plt.plot(D_eqopt,'r*-',linewidth=0.7,label=r'EqOpt: $\mathcal{E}(\widehat{\theta}^{\texttt{UN}}_a,\wi
dehat{\theta}^{\texttt{UN}}_b)$')
plt.plot(D_dp_robust,'bo-',linewidth=0.7,label=r'DP: $\mathcal{E}(\theta^{\texttt{UN}}_a,\theta^{\tex
ttt{UN}}_b)$')
plt.plot(D_dp,'b*-',linewidth=0.7,label=r'DP: $\mathcal{E}(\widehat{\theta}^{\texttt{UN}}_a,\widehat
{\theta}^{\texttt{UN}}_b)$')
plt.plot([0]*40,'k--',linewidth=0.8)
plt.ylabel('Unfairness',fontsize = 16)
plt.xlabel('Round of experiment',fontsize = 16)
plt.legend(loc='upper right', bbox_to_anchor=(0.9, 1.42),ncol=2,fontsize = 14)
plt.savefig('unfairness_improve_105.pdf',bbox_inches='tight')
plt.show()
```