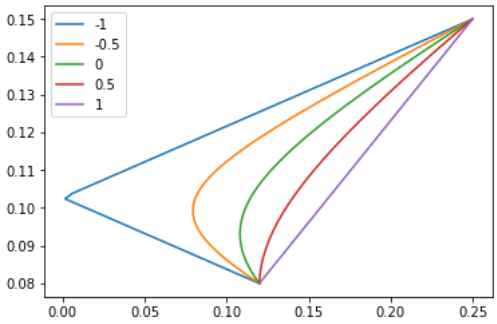
投资组合的收益率和风险

不同相关系数均值标准差关系图

In [3]:
```
import numpy as np
import math
import matplotlib.pyplot as plt
def cal_mean(frac):
    return(0.08*frac+0.15*(1-frac))

mean=list(map(cal_mean,[x/50 for x in range(51)]))
sd_mat=np.array([list(map(lambda x: math.sqrt((x**2)*0.12**2+
((1-x)**2)*0.25**2+2*x*(1-x)*(-1.5+i*0.5)*0.12*0.25),[x/50 for x in range(51)])
) for i in range(1,6)])
#[expression for variable in sequence] list comprehension
plt.plot(sd_mat[0,:],mean,label='-1')
plt.plot(sd_mat[1,:],mean,label='-0.5')
plt.plot(sd_mat[2,:],mean,label='0')
plt.plot(sd_mat[3,:],mean,label='0.5')
plt.plot(sd_mat[4,:],mean,label='1')
plt.legend(loc='upper left')
```

Out[3]: <matplotlib.legend.Legend at 0x1e3723897f0>



Markowitz均值-方差模型

python实现

In [7]:
```
import pandas as pd
stock=pd.read_table('stock.txt',sep='\t',index_col='Trddt')
stock.index=pd.to_datetime(stock.index)

fjgs=stock.ix[stock.Stkcd==600033,'Dretwd']
fjgs.name='fjgs'
zndl=stock.ix[stock.Stkcd==600023,'Dretwd']
zndl.name='zndl'
sykj=stock.ix[stock.Stkcd==600183,'Dretwd']
sykj.name='sykj'
hxyh=stock.ix[stock.Stkcd==600015,'Dretwd']
hxyh.name='hxyh'
byjc=stock.ix[stock.Stkcd==600004,'Dretwd']
byjc.name='byjc'


sh_return=pd.concat([byjc,fjgs,hxyh,sykj,zndl],axis=1)
sh_return.head()
```

```
D:\software\study\Anaconda3\lib\site-packages\ipykernel_launcher.py:5: DeprecationWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprecated
  """
```

Out[7]:

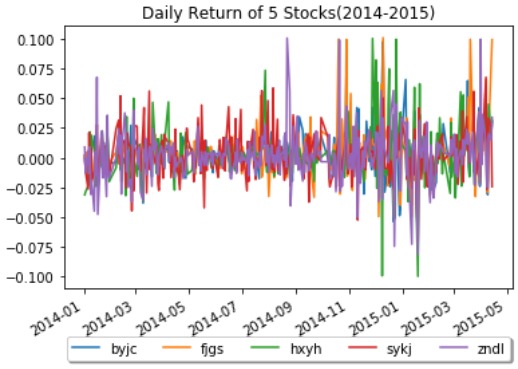
	byjc	fjgs	hxyh	sykj	zndl
Trddt					
2014-01-02	-0.001439	0.000000	-0.031505	0.002024	0.008876
2014-01-03	-0.008646	0.004673	-0.028916	-0.012121	-0.013196
2014-01-06	-0.018895	-0.023256	-0.023573	-0.026585	0.005944
2014-01-07	-0.007407	0.004762	-0.003812	0.021008	-0.013294
2014-01-08	0.005970	-0.014218	0.021684	-0.014403	0.008982

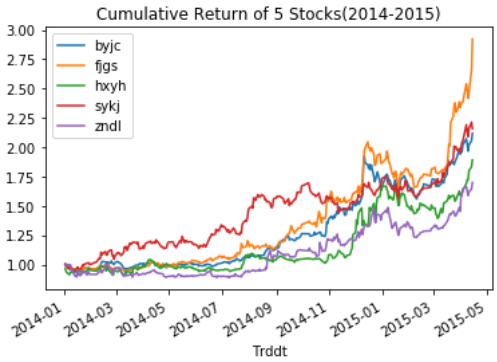
In [8]:
```
sh_return=sh_return.dropna()
#sh_return2=sh_return2.dropna()
sh_return.corr()

cumreturn=(1+sh_return).cumprod()
sh_return.plot()
plt.title('Daily Return of 5 Stocks(2014-2015)')
plt.legend(loc='lower center',bbox_to_anchor=(0.5,-0.3),
           ncol=5, fancybox=True, shadow=True)

cumreturn.plot()
plt.title('Cumulative Return of 5 Stocks(2014-2015)')
```

Out[8]: Text(0.5,1,'Cumulative Return of 5 Stocks(2014-2015)')





In [9]:
```
sh_return.corr()
```

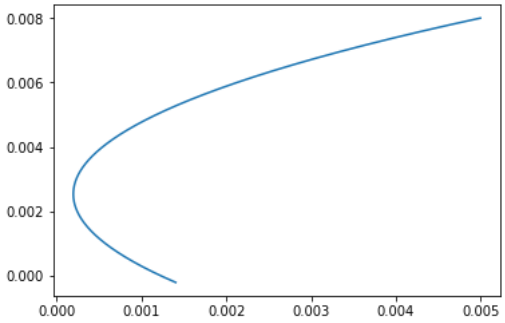
Out[9]:

	byjc	fjgs	hxyh	sykj	zndl
byjc	1.000000	0.591674	0.327680	0.372088	0.429905
fjgs	0.591674	1.000000	0.282111	0.361442	0.397685
hxyh	0.327680	0.282111	1.000000	0.207656	0.369259
sykj	0.372088	0.361442	0.207656	1.000000	0.302859
zndl	0.429905	0.397685	0.369259	0.302859	1.000000

In [15]:
```
from scipy import linalg
class MeanVariance:
    def __init__(self,returns):
        self.returns=returns
    def minVar(self,goalRet):
        covs=np.array(self.returns.cov())
        means=np.array(self.returns.mean())
        L1=np.append(np.append(covs.swapaxes(0,1),[means],0),
                     [np.ones(len(means))],0).swapaxes(0,1)
        L2=list(np.ones(len(means)))
        L2.extend([0,0])
        L3=list(means)
        L3.extend([0,0])
        L4=np.array([L2,L3])
        L=np.append(L1,L4,0)
        results=linalg.solve(L,np.append(np.zeros(len(means)),[1,goalRet],0))
        return(np.array([list(self.returns.columns),results[:-2]]))
    def frontierCurve(self):
        goals=[x/500000 for x in range(-100,4000)]
        variances=list(map(lambda x: self.calVar(self.minVar(x)[1,:].astype(np.float)),goals))
        plt.plot(variances,goals)
    def meanRet(self,fracs):
        meanRisky=ffn.to_returns(self.returns).mean()
        assert len(meanRisky)==len(fracs), 'Length of fractions must be equal to number of assets'
        return(np.sum(np.multiply(meanRisky,np.array(fracs))))
    def calVar(self,fracs):
        return(np.dot(np.dot(fracs,self.returns.cov()),fracs))


minVar=MeanVariance(sh_return)

minVar.frontierCurve()
```

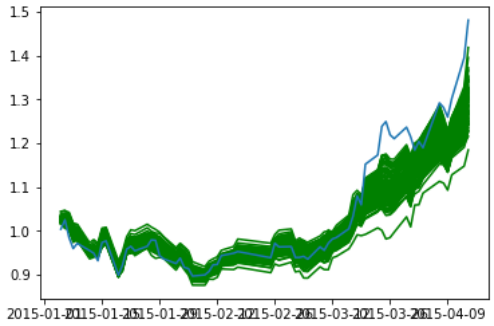


In [16]:
```
train_set=sh_return['2014']
test_set=sh_return['2015']
varMinimizer=MeanVariance(train_set)
goal_return=0.003
portfolio_weight=varMinimizer.minVar(goal_return)
portfolio_weight
test_return=np.dot(test_set,
                   np.array([portfolio_weight[1,:].astype(np.float)]).swapaxes(0,1))
test_return=pd.DataFrame(test_return,index=test_set.index)
test_cum_return=(1+test_return).cumprod()

sim_weight=np.random.uniform(0,1,(100,5))
sim_weight=np.apply_along_axis(lambda x: x/sum(x),1,sim_weight)
sim_return=np.dot(test_set,sim_weight.swapaxes(0,1))
sim_return=pd.DataFrame(sim_return,index=test_cum_return.index)
sim_cum_return=(1+sim_return).cumprod()

plt.plot(sim_cum_return.index,sim_cum_return,color='green')
plt.plot(test_cum_return.index,test_cum_return)
```

Out[16]:
```
[<matplotlib.lines.Line2D at 0x1e373364fd0>]
```



上期所，大期所，郑期所

全样本空间

In [17]:
```
#导入所需包
import pandas as pd
import numpy as np
import math
import datetime as dt
import scipy.optimize as sco
import matplotlib.pyplot as plt
from sqlalchemy import create_engine
pd.set_option("display.precision", 6)
from WindPy import w
w.start()
import pandas as pd
from WindPy import *
import os
from datetime import *
```

```
Welcome to use Wind Quant API for Python (WindPy)!

COPYRIGHT (C) 2017 WIND INFORMATION CO., LTD. ALL RIGHTS RESERVED.
IN NO CIRCUMSTANCE SHALL WIND BE RESPONSIBLE FOR ANY DAMAGES OR LOSSES CAUSED BY USING WIND QUANT API FOR Python.
```

In [19]:
```
names = ['M.DCE','Y.DCE','A.DCE','B.DCE','P.DCE','C.DCE','CS.DCE','JD.DCE','BB.DCE','FB.DCE','L.DCE','V.DCE','PP.DCE','J.DCE','JM.DCE','I.
         'CU.SHF','AL.SHF','ZN.SHF','PB.SHF','NI.SHF','SN.SHF','AU.SHF','AG.SHF','RB.SHF','WR.SHF','HC.SHF','FU.SHF','BU.SHF','RU.SHF',
         'SR.CZC','CF.CZC','ZC.CZC','FG.CZC','TA.CZC','MA.CZC','WH.CZC','PM.CZC','RI.CZC','LR.CZC','JR.CZC','RS.CZC','OI.CZC','RM.CZC','SF
df=w.wsd(names[0], "close", "2013-01-01", "2017-12-31", "")
df = pd.DataFrame([df.Times,df.Data[0]],index=['statistic_date',names[0]],columns=df.Times)
df_t=df.T
for i in names[1:]:
    df=w.wsd(i, "close", "2013-01-01", "2017-12-31", "")
    df = pd.DataFrame([df.Times,df.Data[0]],index=['statistic_date',i],columns=df.Times)
    df=df.T
    df_t=df_t.merge(df,on='statistic_date',how='inner')
```

```
df_t
```

Out[19]:

	statistic_date	M.DCE	Y.DCE	A.DCE	B.DCE	P.DCE	C.DCE	CS.DCE	JD.DCE	BB.DCE	...	WH.CZC	PM.CZC	RI.CZC	LR.CZC	JR.CZC
0	2013-01-04	3248	8768	4740	4674	7028	2446	NaN	NaN	NaN	...	2572	NaN	2706	NaN	NaN
1	2013-01-07	3267	8688	4731	4667	6964	2445	NaN	NaN	NaN	...	2597	2478	2718	NaN	NaN
2	2013-01-08	3291	8660	4739	4664	6844	2456	NaN	NaN	NaN	...	2614	NaN	2721	NaN	NaN
3	2013-01-09	3272	8600	4737	4655	6804	2444	NaN	NaN	NaN	...	2601	2506	2701	NaN	NaN
4	2013-01-10	3268	8562	4750	4640	6726	2445	NaN	NaN	NaN	...	2600	2486	2708	NaN	NaN
5	2013-01-11	3258	8488	4755	4668	6640	2438	NaN	NaN	NaN	...	2593	NaN	2707	NaN	NaN
6	2013-01-14	3361	8530	4806	4657	6698	2434	NaN	NaN	NaN	...	2586	2470	2687	NaN	NaN
7	2013-01-15	3368	8594	4798	4669	6726	2434	NaN	NaN	NaN	...	2594	2463	2683	NaN	NaN
8	2013-01-16	3405	8642	4813	4655	6776	2435	NaN	NaN	NaN	...	2596	2470	2685	NaN	NaN
9	2013-01-17	3381	8628	4791	4645	6712	2431	NaN	NaN	NaN	...	2583	2474	2682	NaN	NaN
10	2013-01-18	3376	8678	4816	4645	6746	2434	NaN	NaN	NaN	...	2583	2464	2715	NaN	NaN
11	2013-01-21	3388	8714	4827	4631	6752	2427	NaN	NaN	NaN	...	2563	2469	2704	NaN	NaN
12	2013-01-22	3426	8848	4844	4627	6818	2426	NaN	NaN	NaN	...	2553	2452	2706	NaN	NaN
13	2013-01-23	3432	8848	4835	4627	6826	2427	NaN	NaN	NaN	...	2553	2455	2714	NaN	NaN
14	2013-01-24	3358	8688	4819	4619	6720	2417	NaN	NaN	NaN	...	2530	2430	2703	NaN	NaN
15	2013-01-25	3383	8736	4832	4604	6724	2419	NaN	NaN	NaN	...	2532	2479	2702	NaN	NaN
16	2013-01-28	3429	8746	4851	4604	6724	2424	NaN	NaN	NaN	...	2535	2462	2702	NaN	NaN
17	2013-01-29	3419	8684	4847	4604	6702	2422	NaN	NaN	NaN	...	2523	NaN	2690	NaN	NaN
18	2013-01-30	3484	8734	4862	4626	6756	2424	NaN	NaN	NaN	...	2530	NaN	2727	NaN	NaN
19	2013-01-31	3508	8820	4855	4635	6836	2424	NaN	NaN	NaN	...	2539	2460	2727	NaN	NaN
20	2013-02-01	3521	8852	4854	4644	6864	2425	NaN	NaN	NaN	...	2544	2477	2726	NaN	NaN
21	2013-02-04	3503	8896	4841	4631	6854	2424	NaN	NaN	NaN	...	2541	NaN	2723	NaN	NaN
22	2013-02-05	3512	8822	4833	4602	7156	2468	NaN	NaN	NaN	...	2530	2476	2722	NaN	NaN
23	2013-02-06	3532	8810	4835	4602	7136	2469	NaN	NaN	NaN	...	2527	2453	2704	NaN	NaN
24	2013-02-07	3519	8764	4816	4602	7116	2464	NaN	NaN	NaN	...	2520	2453	2679	NaN	NaN
25	2013-02-08	3553	8714	4828	4624	7088	2462	NaN	NaN	NaN	...	2524	NaN	2704	NaN	NaN
26	2013-02-18	3286	8646	4782	4456	7056	2440	NaN	NaN	NaN	...	2506	2445	2699	NaN	NaN
27	2013-02-19	3307	8710	4783	4536	7118	2448	NaN	NaN	NaN	...	2518	2449	2703	NaN	NaN
28	2013-02-20	3344	8754	4815	4550	7124	2445	NaN	NaN	NaN	...	2517	2550	2697	NaN	NaN
29	2013-02-21	3351	8630	4794	4540	7024	2447	NaN	NaN	NaN	...	2512	2511	2692	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1185	2017-11-20	2897	5914	3636	3295	5526	1690	2076	4339	131	...	2647	NaN	NaN	NaN	NaN
1186	2017-11-21	2921	5906	3683	3294	5478	1693	2074	4381	132.7	...	2621	NaN	NaN	NaN	NaN
1187	2017-11-22	2826	5918	3684	3328	5494	1698	2077	4490	132.7	...	2630	NaN	NaN	NaN	NaN
1188	2017-11-23	2821	5910	3679	3302	5496	1705	2071	4482	132.7	...	2611	NaN	NaN	NaN	NaN
1189	2017-11-24	2842	5928	3680	3306	5496	1699	2063	4491	132.7	...	2605	NaN	NaN	NaN	NaN
1190	2017-11-27	2823	5882	3667	3304	5402	1702	2071	4498	132.7	...	2565	NaN	NaN	NaN	NaN
1191	2017-11-28	2832	5870	3666	3306	5428	1707	2076	4509	132.7	...	2580	NaN	NaN	NaN	NaN
1192	2017-11-29	2822	5824	3641	3307	5412	1718	2077	4569	132.7	...	2568	NaN	NaN	NaN	NaN
1193	2017-11-30	2816	5854	3560	3309	5414	1715	2052	4493	132.7	...	2544	NaN	NaN	NaN	NaN
1194	2017-12-01	2800	6046	3572	3309	5456	1708	2060	3987	132.7	...	2570	NaN	NaN	3099	NaN
1195	2017-12-04	2871	6056	3600	3327	5492	1719	2106	4007	132.75	...	2579	NaN	NaN	NaN	NaN
1196	2017-12-05	2864	5978	3575	3320	5386	1777	2106	3936	128.65	...	2593	NaN	NaN	NaN	NaN
1197	2017-12-06	2912	5996	3573	3347	5414	1787	2130	3890	129.5	...	2563	NaN	NaN	NaN	NaN
1198	2017-12-07	2880	5880	3453	3339	5316	1769	2091	3846	131.3	...	2528	NaN	NaN	NaN	NaN
1199	2017-12-08	2875	5872	3429	3340	5336	1774	2087	3866	131.3	...	2529	NaN	NaN	NaN	3219
1200	2017-12-11	2850	5890	3621	3331	5324	1769	2066	3820	132.55	...	2530	NaN	NaN	NaN	NaN
1201	2017-12-12	2844	5914	3615	3312	5328	1769	2068	3824	131	...	2530	NaN	NaN	NaN	3335
1202	2017-12-13	2842	5896	3622	3321	5324	1778	2084	3842	131	...	2548	NaN	NaN	NaN	NaN
1203	2017-12-14	2823	5806	3636	3312	5218	1785	2099	3857	131	...	2560	NaN	NaN	NaN	NaN
1204	2017-12-15	2811	5802	3632	3305	5246	1792	2101	3858	131	...	2557	NaN	NaN	NaN	NaN
1205	2017-12-18	2815	5834	3674	3340	5294	1823	2146	3870	129.85	...	2555	NaN	NaN	NaN	3292
1206	2017-12-19	2818	5782	3669	3325	5254	1816	2134	3808	130.35	...	2532	NaN	NaN	NaN	NaN
1207	2017-12-20	2804	5778	3647	3322	5274	1819	2142	3781	129.8	...	2530	NaN	NaN	NaN	3298
1208	2017-12-21	2790	5690	3642	3322	5180	1836	2141	3775	130.45	...	2542	NaN	NaN	NaN	3338
1209	2017-12-22	2786	5622	3601	3300	5134	1823	2138	3789	130.45	...	2545	NaN	NaN	NaN	NaN
1210	2017-12-25	2794	5660	3666	3308	5176	1807	2107	3809	130.45	...	2540	NaN	NaN	3088	NaN

1211	2017-12-26	2820	5662	3670	3301	5178	1810	2117	3781	130.45	...	2544	NaN	NaN	NaN	NaN
1212	2017-12-27	2824	5706	3670	3293	5232	1816	2123	3805	130.45	...	2532	NaN	NaN	NaN	NaN
1213	2017-12-28	2806	5694	3632	3280	5248	1814	2120	3820	130.45	...	2575	NaN	NaN	NaN	NaN
1214	2017-12-29	2761	5668	3626	3266	5226	1816	2123	3815	129.1	...	2578	NaN	NaN	NaN	NaN

statistic_date	M.DCE	Y.DCE	A.DCE	B.DCE	P.DCE	C.DCE	CS.DCE	JD.DCE	BB.DCE	...	WH.CZC	PM.CZC	RI.CZC	LR.CZC	JR.CZC

1215 rows × 47 columns

In [22]:
```
df = df_t
df=df.sort_values(by='statistic_date')
df=df.fillna(method='ffill')
df.index=df['statistic_date']
df=df[names]
df_tmp=df.copy(False)
df=df/df.shift(1)-1
rets=df.dropna()
rets_temp=rets.copy(False)
################################################################################
##############################建立风险平价模型#################################

###################################原始风险平价模型
n_assets=rets.shape[1]
##1.1方差风险平价
var_a=((rets-rets.mean())**2).sum()/(len(rets)-1)
ori_rp_var_w=(1/var_a)/((1/var_a).sum())
ori_rp_var_w
```

Out[22]:
```
M.DCE      0.023543
Y.DCE      0.036850
A.DCE      0.031392
B.DCE      0.029385
P.DCE      0.022258
C.DCE      0.027745
CS.DCE     0.021724
JD.DCE     0.008370
BB.DCE     0.007555
FB.DCE     0.004053
L.DCE      0.017892
V.DCE      0.019601
PP.DCE     0.014865
J.DCE      0.006574
JM.DCE     0.006522
I.DCE      0.005500
CU.SHF     0.025938
AL.SHF     0.032324
ZN.SHF     0.018274
PB.SHF     0.018875
NI.SHF     0.014132
SN.SHF     0.024460
AU.SHF     0.057262
AG.SHF     0.026146
RB.SHF     0.008992
WR.SHF     0.014518
HC.SHF     0.009726
FU.SHF     0.006576
BU.SHF     0.008657
RU.SHF     0.006896
SR.CZC     0.044047
CF.CZC     0.023438
ZC.CZC     0.012586
FG.CZC     0.016084
TA.CZC     0.023595
MA.CZC     0.012969
WH.CZC     0.048839
PM.CZC     0.017718
RI.CZC     0.044720
LR.CZC     0.032657
JR.CZC     0.081124
RS.CZC     0.018726
OI.CZC     0.035635
RM.CZC     0.016859
SF.CZC     0.009933
SM.CZC     0.004464
dtype: float64
```

In [24]:
```
##1.2最大回撤风险平价
df_tmp1 = (df_tmp.cummax() - df_tmp) / df_tmp
dy_dd=df_tmp1[1:].max()
ori_rp_maxDD_w=(1/dy_dd)/((1/dy_dd).sum())
ori_rp_maxDD_w
```

Out[24]:
```
M.DCE      0.019168
Y.DCE      0.019284
A.DCE      0.029428
B.DCE      0.020243
P.DCE      0.018309
C.DCE      0.016269
CS.DCE     0.014959
JD.DCE     0.015213
BB.DCE     0.010826
FB.DCE     0.014904
L.DCE      0.019510
V.DCE      0.025362
PP.DCE     0.012280
J.DCE      0.005802
```

```
JM.DCE     0.008695
I.DCE      0.005612
CU.SHF     0.017356
AL.SHF     0.023723
ZN.SHF     0.030875
PB.SHF     0.032513
NI.SHF     0.018190
SN.SHF     0.026406
AU.SHF     0.023458
AG.SHF     0.012534
RB.SHF     0.008580
WR.SHF     0.009851
HC.SHF     0.012944
FU.SHF     0.006839
BU.SHF     0.007591
RU.SHF     0.007688
SR.CZC     0.039119
CF.CZC     0.013160
ZC.CZC     0.012351
FG.CZC     0.013379
TA.CZC     0.012478
MA.CZC     0.011107
WH.CZC     0.049882
PM.CZC     0.064867
RI.CZC     0.050691
LR.CZC     0.051838
JR.CZC     0.090889
RS.CZC     0.025279
OI.CZC     0.016362
RM.CZC     0.016740
SF.CZC     0.018432
SM.CZC     0.019012
dtype: float64
```

In [29]:
```
##1.3风险价值和条件风险价值的风险平价
dfA=rets_temp
VaR,CVaR=np.ones(n_assets),np.ones(n_assets)

for i in range(n_assets):
    dfi=dfA.ix[:,i].sort_values()
    dfi=np.array(dfi)
    dfi=dfi.flatten()
    alpha=0.05
    M=1000
    T=len(dfi)
    np.random.seed(M)
    df_random=np.random.choice(dfi,[M,T])
    df_random.sort(axis=1)
    j=math.floor((T-1)*alpha+1)
    g=(T-1)*alpha+1-j
    VaR[i]=max(0,-(df_random[:,j-1]+g*(df_random[:,j]-df_random[:,j-1])).mean())
    CVaR[i]=max(0,-df_random[:,0:j-1].sum(axis=1).mean())

ori_rp_VaR_w=(1/VaR)/((1/VaR).sum())
ori_rp_VaR_w
```

Out[29]:
```
array([2.28915775e-04, 2.84078317e-04, 2.63897715e-04, 3.61875308e-04,
       2.10998805e-04, 2.89116303e-04, 2.26046468e-04, 1.98707083e-04,
       1.19321115e-04, 1.26040119e-04, 1.89773757e-04, 1.97390893e-04,
       1.71574396e-04, 1.12964681e-04, 1.10064388e-04, 1.00664706e-04,
       2.41142203e-04, 2.50249823e-04, 1.96873925e-04, 2.05441269e-04,
       1.53133799e-04, 2.18933186e-04, 3.68239939e-04, 2.61220326e-04,
       1.36301865e-04, 2.66627061e-04, 1.41275748e-04, 1.00806465e-04,
       1.20600759e-04, 1.17041736e-04, 3.38264742e-04, 2.34751543e-04,
       1.72945289e-04, 1.68837555e-04, 2.19808868e-04, 1.68113837e-04,
       4.18795206e-04, 4.82891366e-04, 3.65123877e-04, 3.04931733e-02,
       9.59954768e-01, 2.56324375e-04, 2.80569043e-04, 1.82369892e-04,
       1.42073070e-04, 1.51872159e-04])
```

In [30]:
```
ori_rp_CVaR_w=(1/CVaR)/((1/CVaR).sum())
ori_rp_CVaR_w
```

Out[30]:
```
array([0.02240013, 0.03141093, 0.02692391, 0.02389836, 0.0243284 ,
       0.02181593, 0.02194338, 0.01752293, 0.01397591, 0.00978453,
       0.01915382, 0.02113291, 0.01812488, 0.0111169 , 0.01156918,
       0.01070742, 0.02554262, 0.02790118, 0.02125539, 0.02156879,
       0.01747112, 0.02361967, 0.04054574, 0.02334499, 0.01415375,
       0.01458145, 0.01444537, 0.0128168 , 0.01423463, 0.01250373,
       0.03332995, 0.02270741, 0.01552302, 0.01897503, 0.02291541,
       0.01713449, 0.03252752, 0.01822732, 0.02777796, 0.04139707,
       0.06542396, 0.01973424, 0.02984879, 0.01890674, 0.01417536,
       0.01160095])
```

In [32]:
```
##################################最优风险平价模型
#求解成分方差并进行投资组合优化
def min_func_diff_cvar2(weights):
    weights=np.array(weights)
    cvar_w=(weights*np.dot(rets.cov(),weights))
    cvar_w=np.array(cvar_w)
    cvar_A=cvar_w
    for i in range(len(cvar_w)):
        cvar_A[i]=((cvar_w[i]-cvar_w)**2).sum()
    diff_cvar2=cvar_A.sum()
    return np.array(diff_cvar2)

cons=({'type':'eq','fun':lambda x:np.sum(x)-1})
bnds=tuple((0,1) for x in range(n_assets))
```

```
opts=sco.minimize(min_func_diff_cvar2,n_assets*[1./n_assets,],method='SLSQP',\
                  bounds=bnds,constraints=cons,tol=1e-20)
                  #由于该函数数值一般比较小。所以记得改变这里的容忍度
opt_rp_var_w=opts['x']

def perfomance(weights):
    weights=np.array(weights)
    year_ret=(1+np.sum(rets.mean()*weights))**T-1
    year_std=np.sqrt(np.dot(weights.T,np.dot(rets.cov()*T,weights)))
    return np.array([year_ret,year_std,(year_ret)/year_std])

#########################################################################
##不同风险平价模型组合权重比较
df_ori_rp_var_w=pd.DataFrame(ori_rp_var_w)
df_ori_rp_maxDD_w=pd.DataFrame(ori_rp_maxDD_w)
df_ori_rp_VaR_w=pd.DataFrame(ori_rp_VaR_w)
df_ori_rp_VaR_w.index=df_ori_rp_var_w.index
df_ori_rp_CVaR_w=pd.DataFrame(ori_rp_CVaR_w)
df_ori_rp_CVaR_w.index=df_ori_rp_var_w.index
df_opt_rp_var_w=pd.DataFrame(opt_rp_var_w)
df_opt_rp_var_w.index=df_ori_rp_var_w.index
df_weights=pd.concat([df_ori_rp_var_w,ori_rp_maxDD_w,df_ori_rp_VaR_w,df_ori_rp_CVaR_w,df_opt_rp_var_w],axis=1)
df_weights.columns=['原始波动率平价','最大回撤风险平价','原始VaR平价','原始CVaR平价','最优化波动率平价']
df_weights
```

Out[32]:

	原始波动率平价	最大回撤风险平价	原始VaR平价	原始CVaR平价	最优化波动率平价
M.DCE	0.023543	0.019168	0.000229	0.022400	0.015106
Y.DCE	0.036850	0.019284	0.000284	0.031411	0.023845
A.DCE	0.031392	0.029428	0.000264	0.026924	0.028356
B.DCE	0.029385	0.020243	0.000362	0.023898	0.052557
P.DCE	0.022258	0.018309	0.000211	0.024328	0.015799
C.DCE	0.027745	0.016269	0.000289	0.021816	0.035224
CS.DCE	0.021724	0.014959	0.000226	0.021943	0.027340
JD.DCE	0.008370	0.015213	0.000199	0.017523	0.018103
BB.DCE	0.007555	0.010826	0.000119	0.013976	0.018878
FB.DCE	0.004053	0.014904	0.000126	0.009785	0.017258
L.DCE	0.017892	0.019510	0.000190	0.019154	0.013027
V.DCE	0.019601	0.025362	0.000197	0.021133	0.014816
PP.DCE	0.014865	0.012280	0.000172	0.018125	0.010615
J.DCE	0.006574	0.005802	0.000113	0.011117	0.005861
JM.DCE	0.006522	0.008695	0.000110	0.011569	0.006454
I.DCE	0.005500	0.005612	0.000101	0.010707	0.007674
CU.SHF	0.025938	0.017356	0.000241	0.025543	0.011871
AL.SHF	0.032324	0.023723	0.000250	0.027901	0.026163
ZN.SHF	0.018274	0.030875	0.000197	0.021255	0.011315
PB.SHF	0.018875	0.032513	0.000205	0.021569	0.013020
NI.SHF	0.014132	0.018190	0.000153	0.017471	0.009742
SN.SHF	0.024460	0.026406	0.000219	0.023620	0.012222
AU.SHF	0.057262	0.023458	0.000368	0.040546	0.051772
AG.SHF	0.026146	0.012534	0.000261	0.023345	0.018930
RB.SHF	0.008992	0.008580	0.000136	0.014154	0.005065
WR.SHF	0.014518	0.009851	0.000267	0.014581	0.046713
HC.SHF	0.009726	0.012944	0.000141	0.014445	0.006396
FU.SHF	0.006576	0.006839	0.000101	0.012817	0.016718
BU.SHF	0.008657	0.007591	0.000121	0.014235	0.006723
RU.SHF	0.006896	0.007688	0.000117	0.012504	0.003520
SR.CZC	0.044047	0.039119	0.000338	0.033330	0.031202
CF.CZC	0.023438	0.013160	0.000235	0.022707	0.012297
ZC.CZC	0.012586	0.012351	0.000173	0.015523	0.010608
FG.CZC	0.016084	0.013379	0.000169	0.018975	0.010401
TA.CZC	0.023595	0.012478	0.000220	0.022915	0.010200
MA.CZC	0.012969	0.011107	0.000168	0.017134	0.006912
WH.CZC	0.048839	0.049882	0.000419	0.032528	0.052760
PM.CZC	0.017718	0.064867	0.000483	0.018227	0.054508
RI.CZC	0.044720	0.050691	0.000365	0.027778	0.055689
LR.CZC	0.032657	0.051838	0.030493	0.041397	0.062311
JR.CZC	0.081124	0.090889	0.959955	0.065424	0.063101
RS.CZC	0.018726	0.025279	0.000256	0.019734	0.033453
OI.CZC	0.035635	0.016362	0.000281	0.029849	0.011209

RM.CZC	0.016859	0.016740	0.000182	0.018907	0.008106
SF.CZC	0.009933	0.018432	0.000142	0.014175	0.011416
SM.CZC	0.004464	0.019012	0.000152	0.011601	0.014744

In [33]:
```
##########################################################################
##不同风险平价模型组合预期业绩指标
df_ori_rp_var_w=pd.DataFrame(perfomance(ori_rp_var_w ))
df_ori_rp_maxDD_w=pd.DataFrame(perfomance(ori_rp_maxDD_w))
df_ori_rp_VaR_w=pd.DataFrame(perfomance(ori_rp_VaR_w))
df_ori_rp_CVaR_w=pd.DataFrame(perfomance(ori_rp_CVaR_w))
df_opt_rp_var_w=pd.DataFrame(perfomance(opt_rp_var_w))
df_per=pd.concat([df_ori_rp_var_w,df_ori_rp_maxDD_w,df_ori_rp_VaR_w,df_ori_rp_CVaR_w,df_opt_rp_var_w],axis=1)
df_per.columns=['原始波动率平价','原始最大回撤平价','原始VaR平价','原始CVaR平价','最优化波动率平价']
df_per.index=['预期年化收益率','预期年化年化波动率','预期年化夏普比']
df_per
```

Out[33]:

	原始波动率平价	原始最大回撤平价	原始VaR平价	原始CVaR平价	最优化波动率平价
预期年化收益率	0.174850	0.207908	0.085333	0.215980	0.170404
预期年化年化波动率	0.136243	0.131698	0.175707	0.151830	0.112559
预期年化夏普比	1.283368	1.578671	0.485654	1.422509	1.513906

设定目标波动率

In [34]:
```
#组合预期年化波动率小于等于0.1"""
lambda_var=0.1/df_per.ix[1,0]
df_weights1=df_weights[['原始波动率平价','最优化波动率平价']]
df_lambda_weights=df_weights1*lambda_var
df_lambda_weights
```

```
D:\software\study\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: DeprecationWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprecated
```

Out[34]:

	原始波动率平价	最优化波动率平价
M.DCE	0.017280	0.011087
Y.DCE	0.027047	0.017502
A.DCE	0.023041	0.020812
B.DCE	0.021568	0.038576
P.DCE	0.016337	0.011596
C.DCE	0.020364	0.025854
CS.DCE	0.015945	0.020067
JD.DCE	0.006143	0.013287
BB.DCE	0.005545	0.013856
FB.DCE	0.002975	0.012667
L.DCE	0.013133	0.009561
V.DCE	0.014387	0.010875
PP.DCE	0.010911	0.007792
J.DCE	0.004825	0.004302
JM.DCE	0.004787	0.004737
I.DCE	0.004037	0.005633
CU.SHF	0.019038	0.008713
AL.SHF	0.023725	0.019203
ZN.SHF	0.013413	0.008305
PB.SHF	0.013854	0.009556
NI.SHF	0.010372	0.007151
SN.SHF	0.017953	0.008971
AU.SHF	0.042029	0.038000
AG.SHF	0.019190	0.013895
RB.SHF	0.006600	0.003718
WR.SHF	0.010656	0.034286
HC.SHF	0.007138	0.004695
FU.SHF	0.004827	0.012270
BU.SHF	0.006354	0.004934
RU.SHF	0.005062	0.002583
SR.CZC	0.032330	0.022902

CF.CZC	0.017203	0.009026
ZC.CZC	0.009238	0.007786
FG.CZC	0.011805	0.007634
TA.CZC	0.017319	0.007487
MA.CZC	0.009519	0.005073
WH.CZC	0.035847	0.038725
PM.CZC	0.013005	0.040008
RI.CZC	0.032824	0.040874
LR.CZC	0.023970	0.045735
JR.CZC	0.059543	0.046315
RS.CZC	0.013745	0.024554
OI.CZC	0.026155	0.008227
RM.CZC	0.012374	0.005949
SF.CZC	0.007291	0.008379
SM.CZC	0.003277	0.010822