1.

1(a)

Since $K_1, K_2, \cdots K_m$ are valid kernel functions

$K_1, K_2, \cdots K_m$ are symmetric and positive semi-definite function

$K_j(x, x') = K_j(x', x) \quad x, x' \in R, \ j = 1, 2, \cdots m$

$$K(x, x') = \sum_{j=1}^{m} w_j K_j(x, x') = \sum_{j=1}^{m} w_j K_j(x', x) = K(x', x)$$

$\Rightarrow K$ is a symmetric function

$\because K_j$ is positive semi-definite function

$\therefore z^T K_j z \geq 0$ for any z

$$K = \sum_{j=1}^{m} w_j K_j \Rightarrow z^T K z = z^T \sum_{j=1}^{m} w_j K_j \cdot z = \sum_{j=1}^{m} \underbrace{w_j}_{\geq 0} \underbrace{z^T K_j z}_{\geq 0} \geq 0$$

$\Rightarrow K$ is a positive semi-definite function

K is a valid kernel function

1(b)

$$k(x, x') = \exp[-(x-x')^2/2016]$$

$$= \exp\left[-\frac{(x-x')^2}{2\sigma^2}\right], \text{ where } 2\sigma^2 = 2016$$

$$= \exp\left[-\frac{(x/\sigma - x'/\sigma)^2}{2}\right]$$

$$= \exp[-(y-y')^2/2]$$

$$= h(y-y')$$

$\because h(t) = \exp(-t^2/2) = E[e^{itZ}]$, where random variable $Z \sim N(0, 1)$

For any $x_1, x_2 \cdots x_n$ and $a_1, a_2 \cdots a_n \in R$,

we have $y_1, y_2 \cdots y_n \in R$

$$\text{So } \sum_{j,k=1}^{n} a_j a_k h(y_j - y_k) = \sum_{j,k=1}^{n} a_j a_k E[e^{i(y_j - y_k)Z})$$

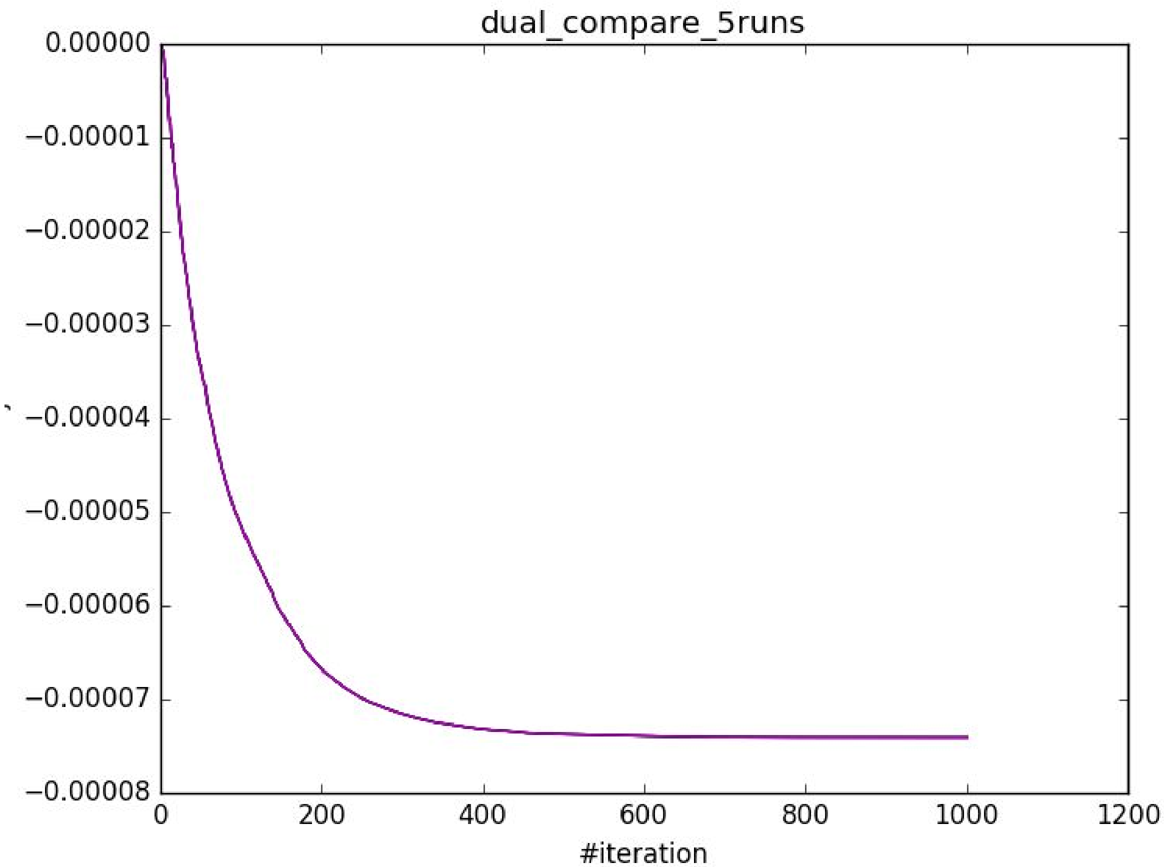
$$= E\left[\sum_{j,k=1}^{n} a_j e^{iy_jZ} a_k e^{-iy_kZ}\right]$$

$$= E\left[\left|\sum_{j=1}^{n} a_j e^{iy_jZ}\right|^2\right] \geq 0$$

2

Each time, we select randomly two records from entire data and update these two alpha. We randomly select two instead of N can save number of gradient computation when updating alpha as well as w.

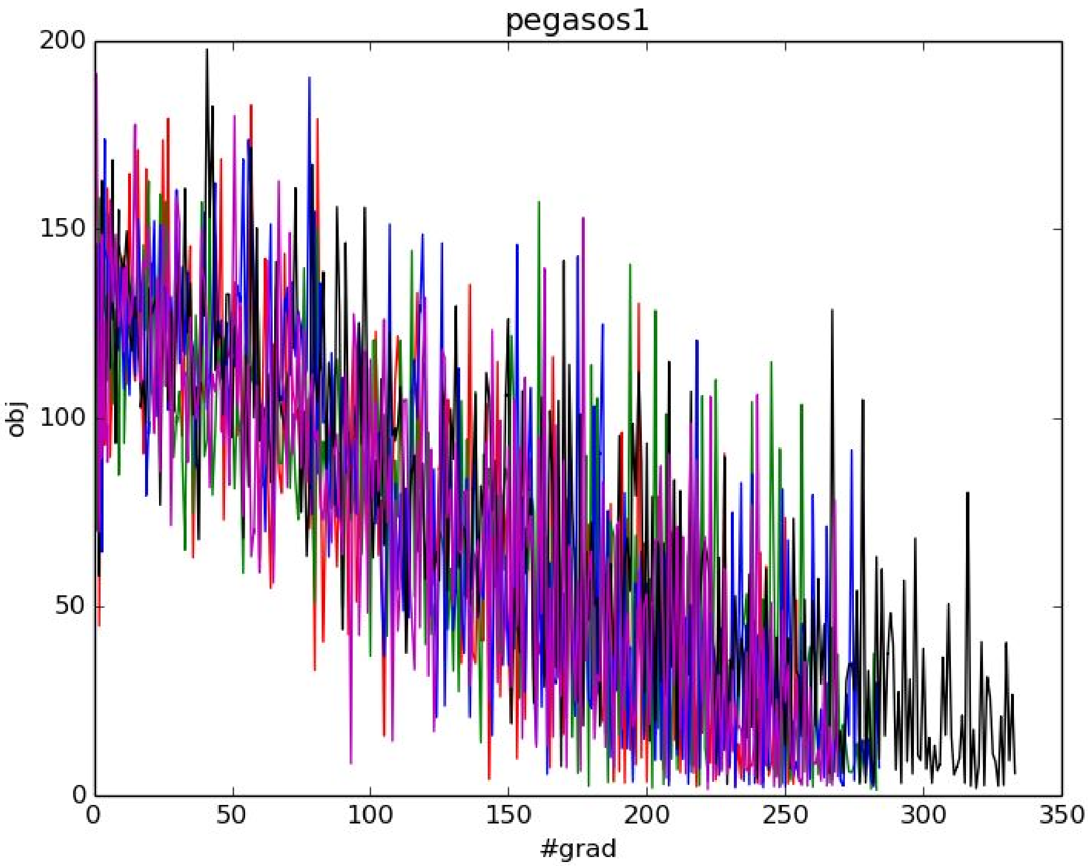
average time= 494.096773052, std= 39.5641404267



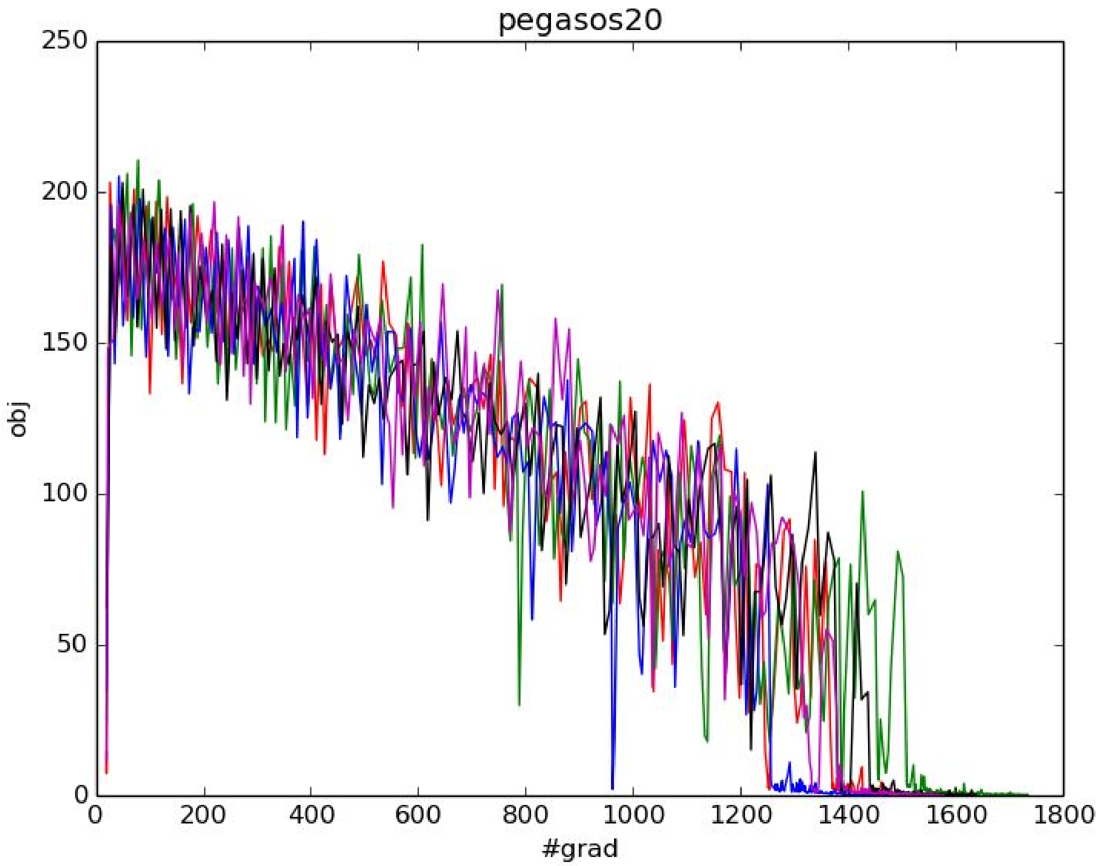
3(1)

k=1,

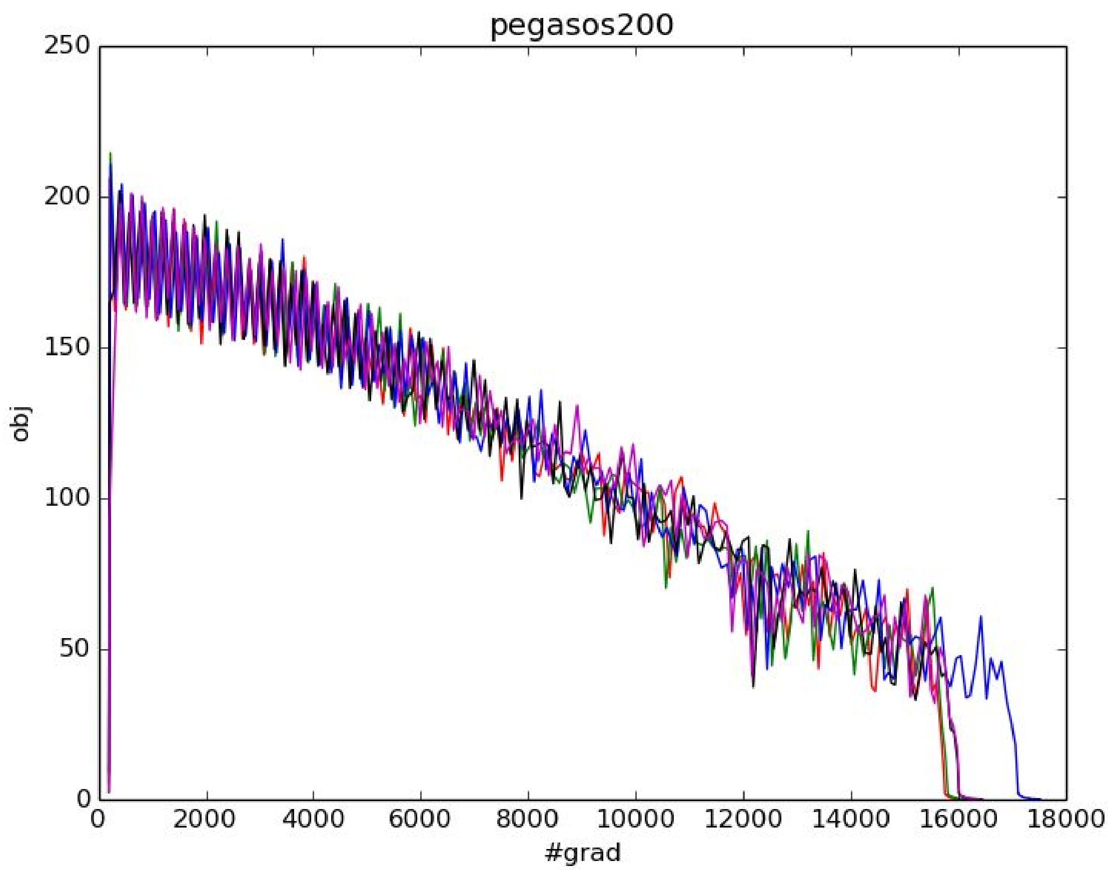
You need to notice that although it looks like random noise, it is because we restrict the number of outer iteration is 1000. If I extend the number of outer iteration to 10,000, the object function value could decrease to less than 1.



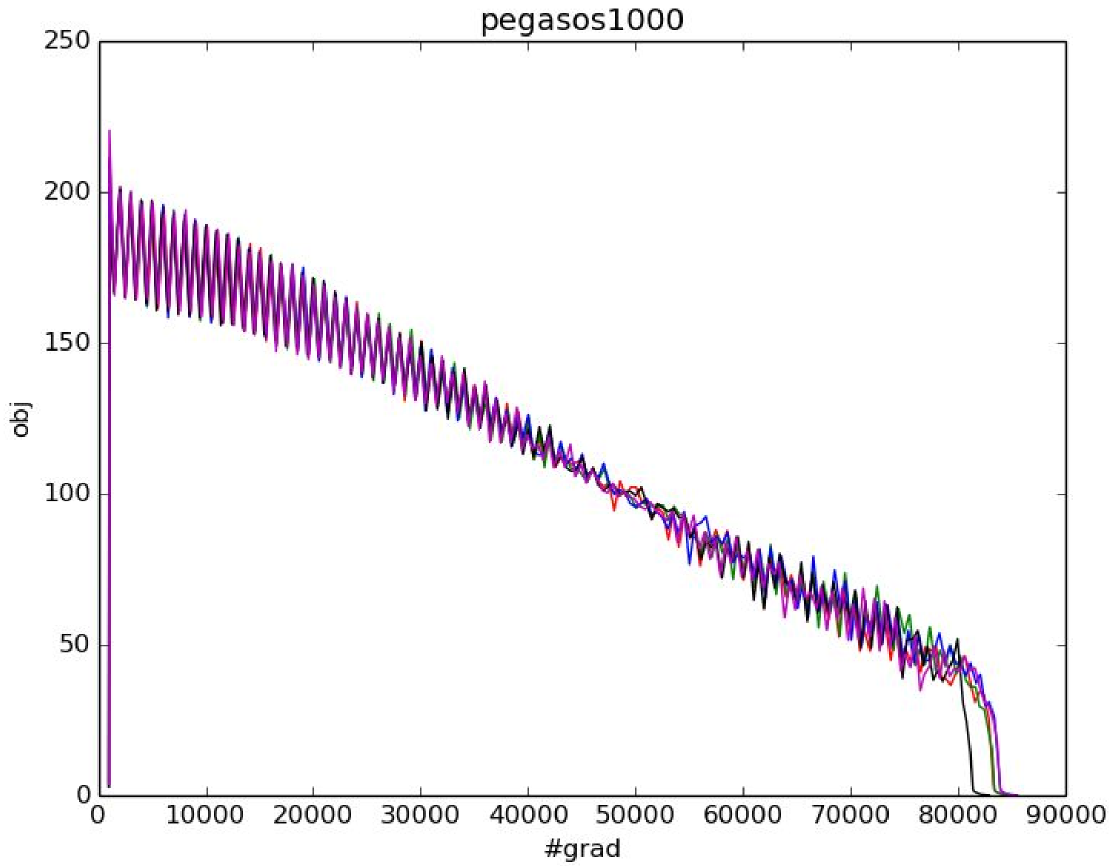
k=20,


pegasos20
250
200
150
100
50
0
obj
0
200
400
600
800
1000
1200
1400
1600
1800
#grad

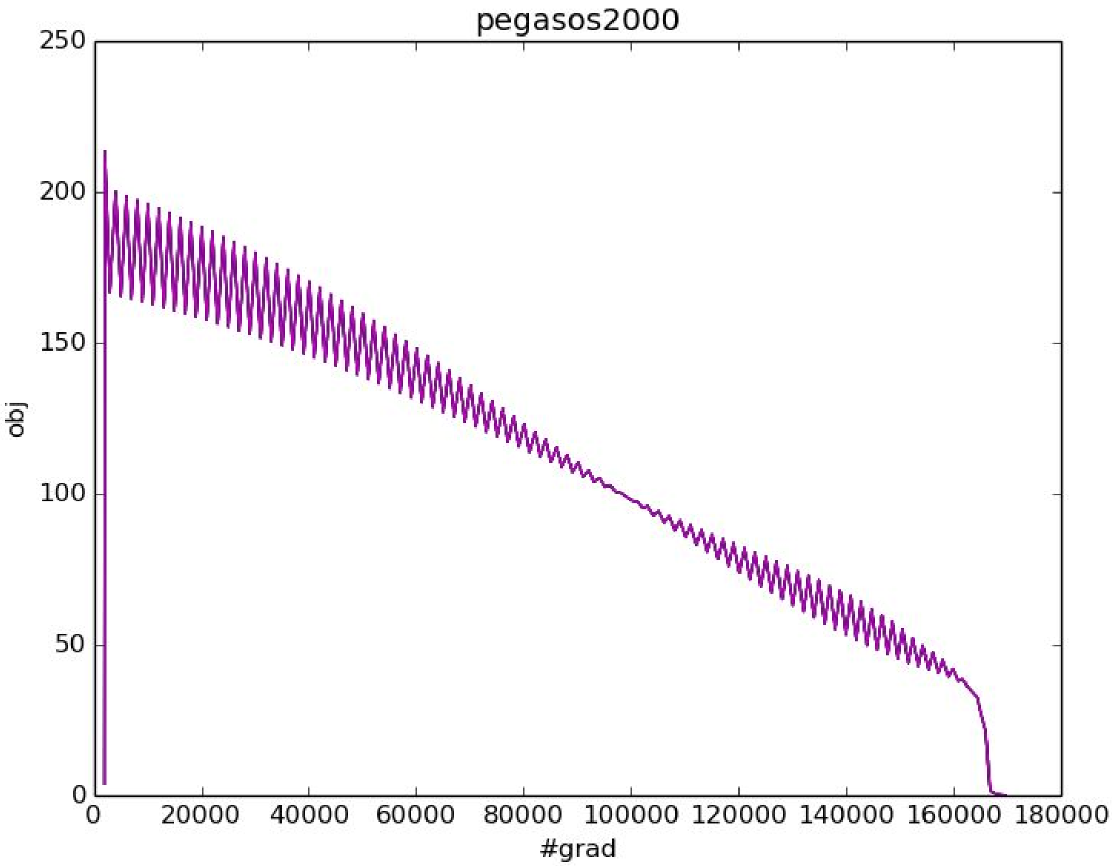
k=200,


pegasos200
obj
#grad
250
200
150
100
50
0
0
2000
4000
6000
8000
10000
12000
14000
16000
18000

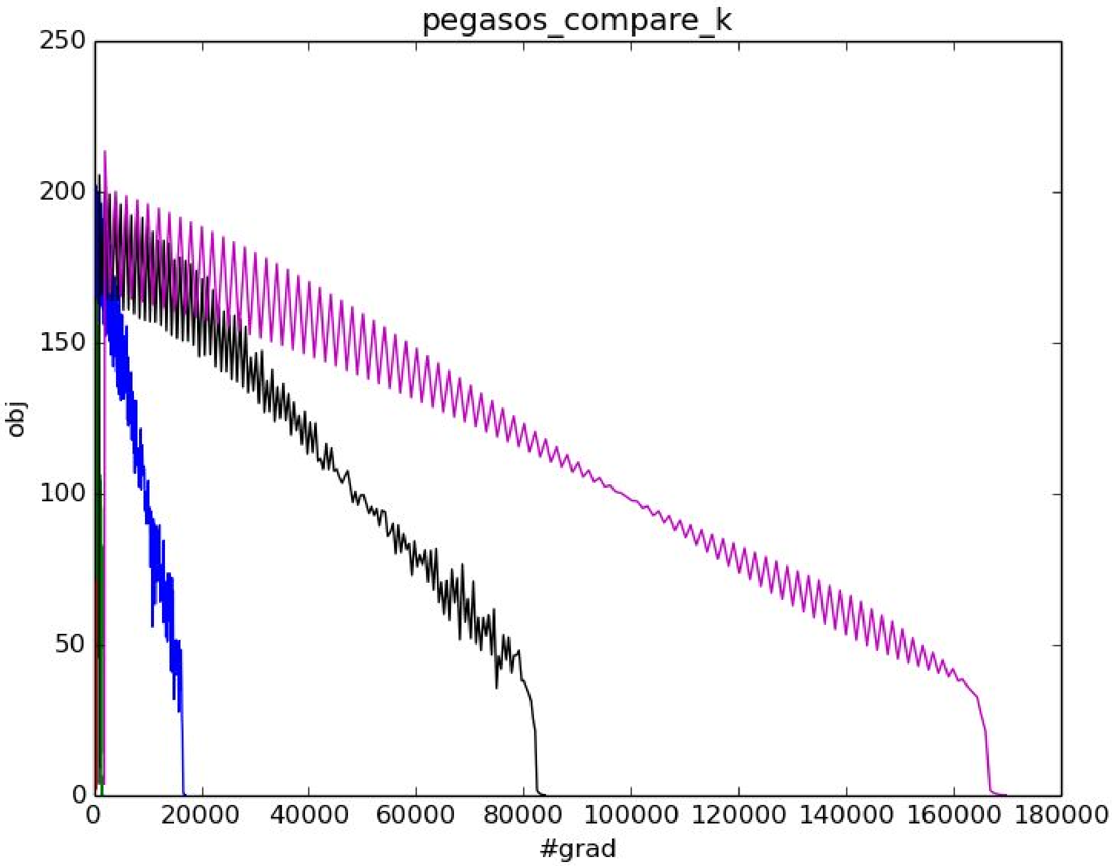
k=1000,


pegasos1000
obj
#grad
250
200
150
100
50
0
0
10000
20000
30000
40000
50000
60000
70000
80000
90000

k=2000,

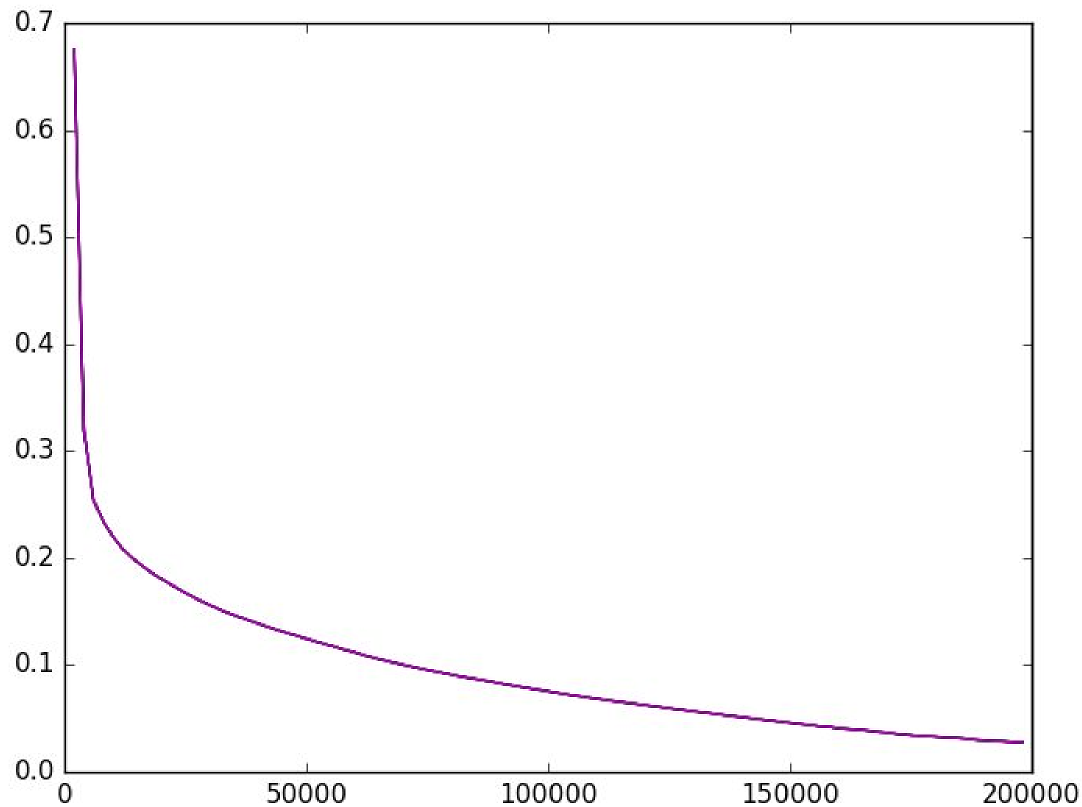


Compare different k
k_list=[1,20,200,1000,2000]
5 colours=['r','g','b','k','m']

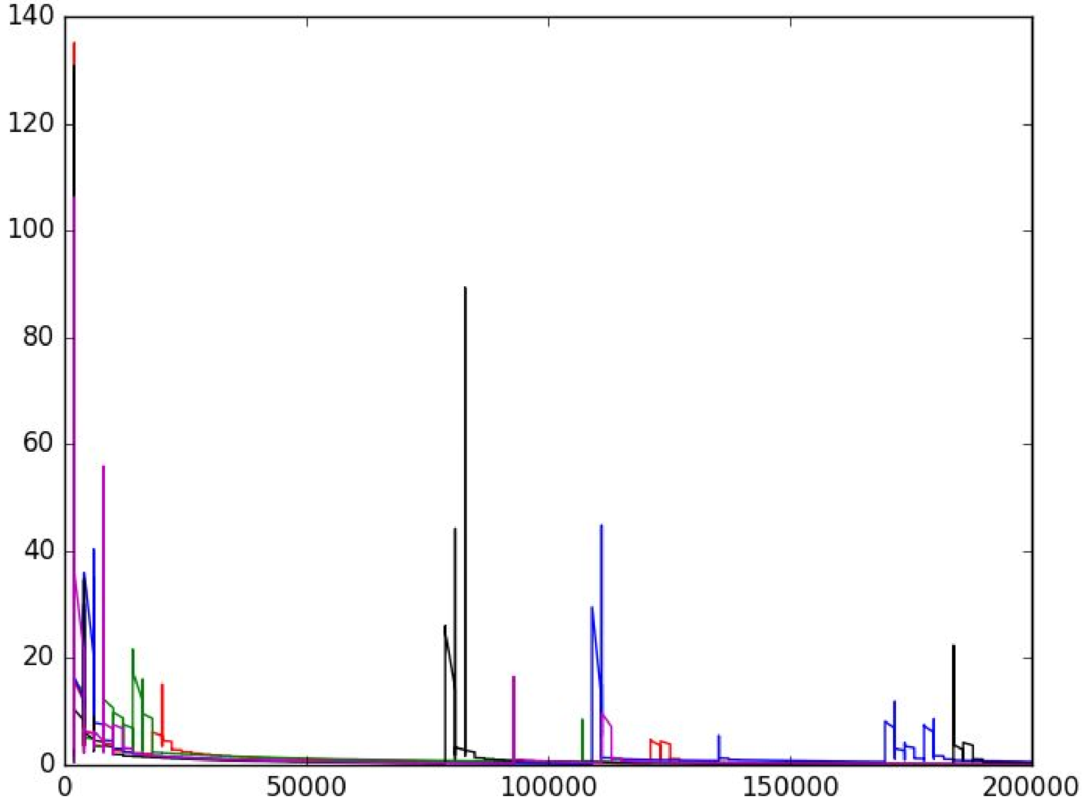


k	Average time	std
1	2.37789278	0.3573610381
20	2.483677816	0.08827807242
200	11.16191392	1.7335725
1000	10.50129437	1.154234946
2000	10.87798982	0.9878795626

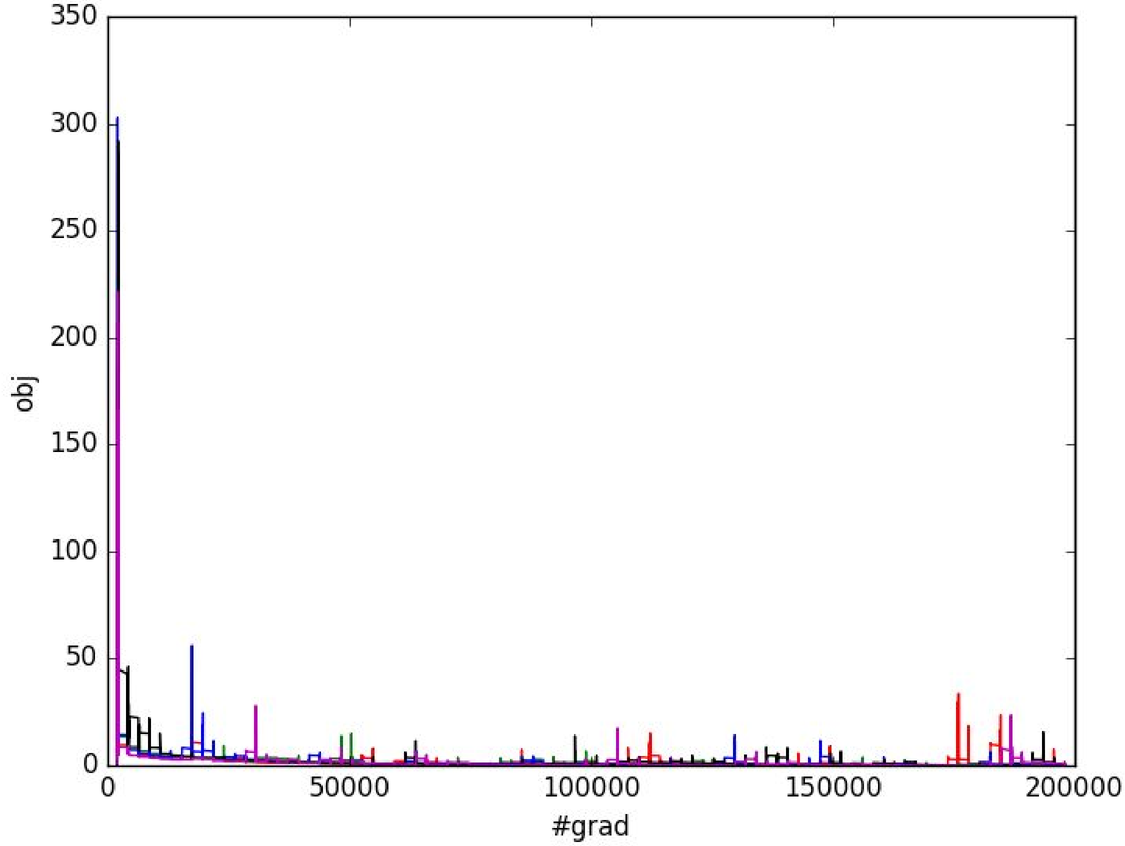
3.(2)
Note that I use k_tot as x_axis
m=1,
It looks like one line. In reality, five line overlay each other. obj= 0.0274177412016,
outer_iter= 100



m=20,

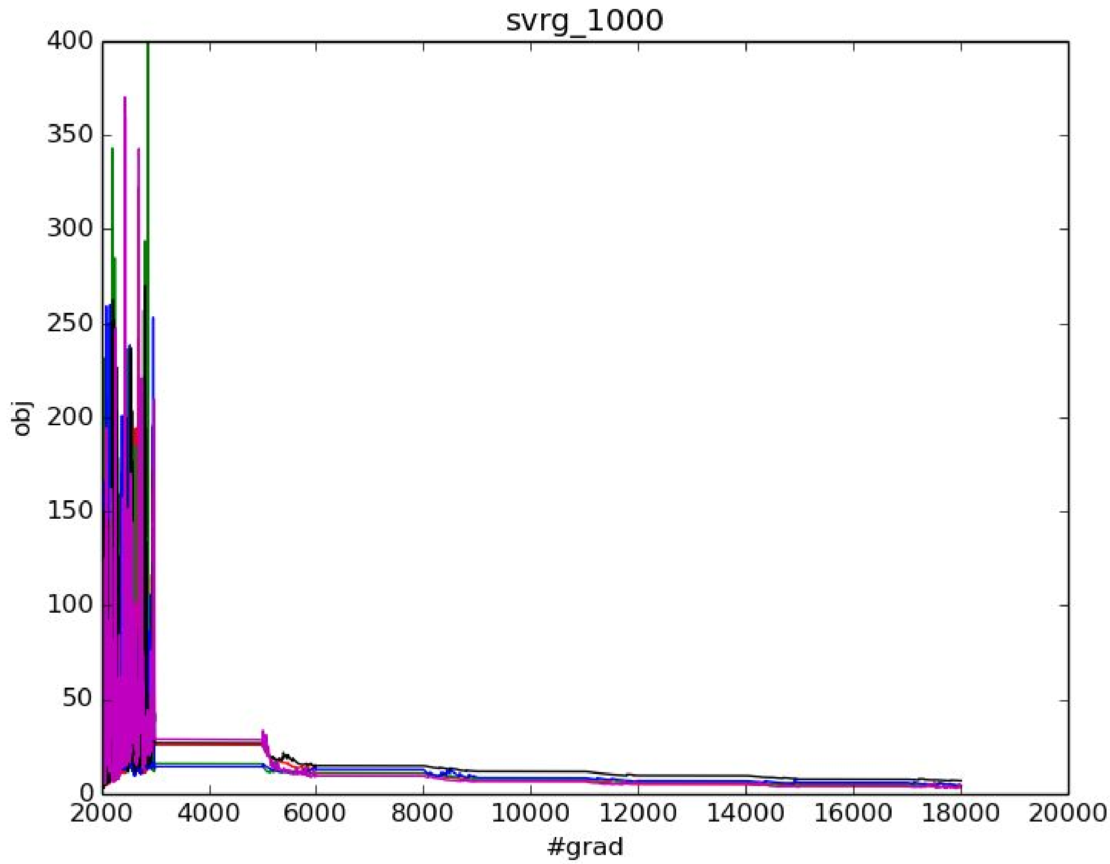


m=200,

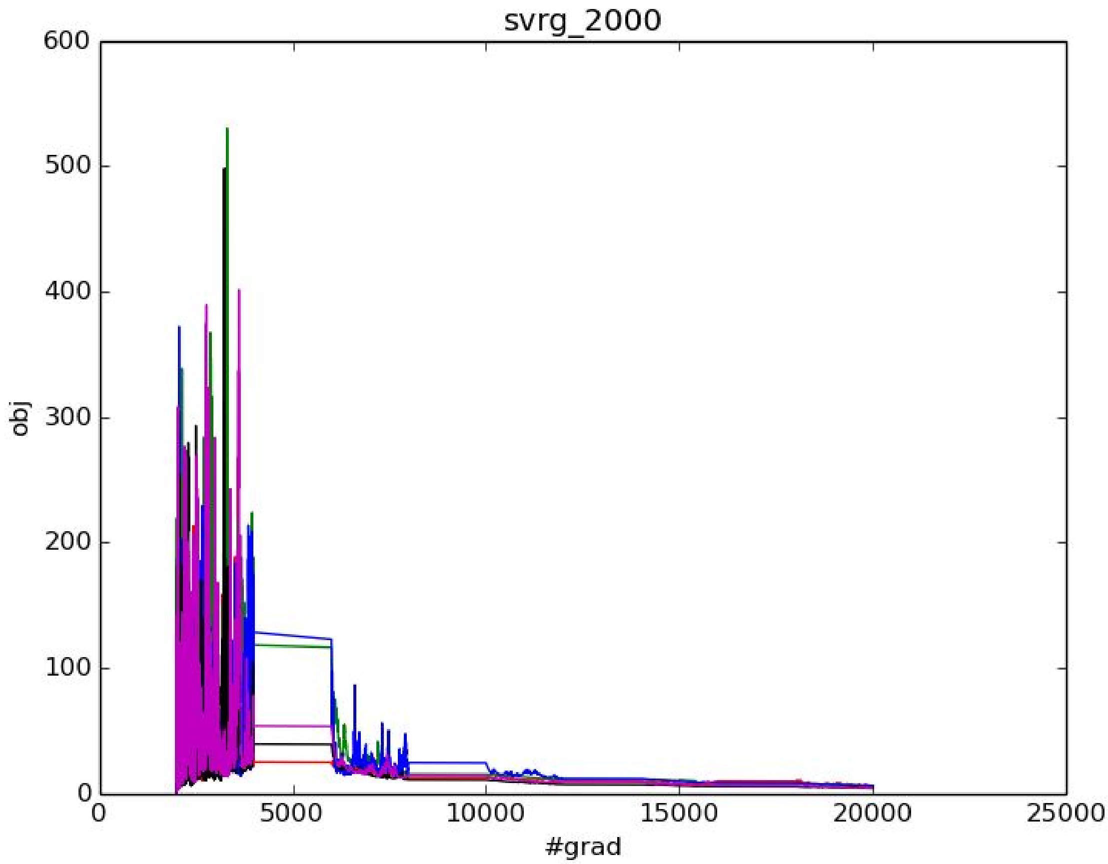


350
300
250
200
150
100
50
0
obj
0
50000
100000
150000
200000
#grad

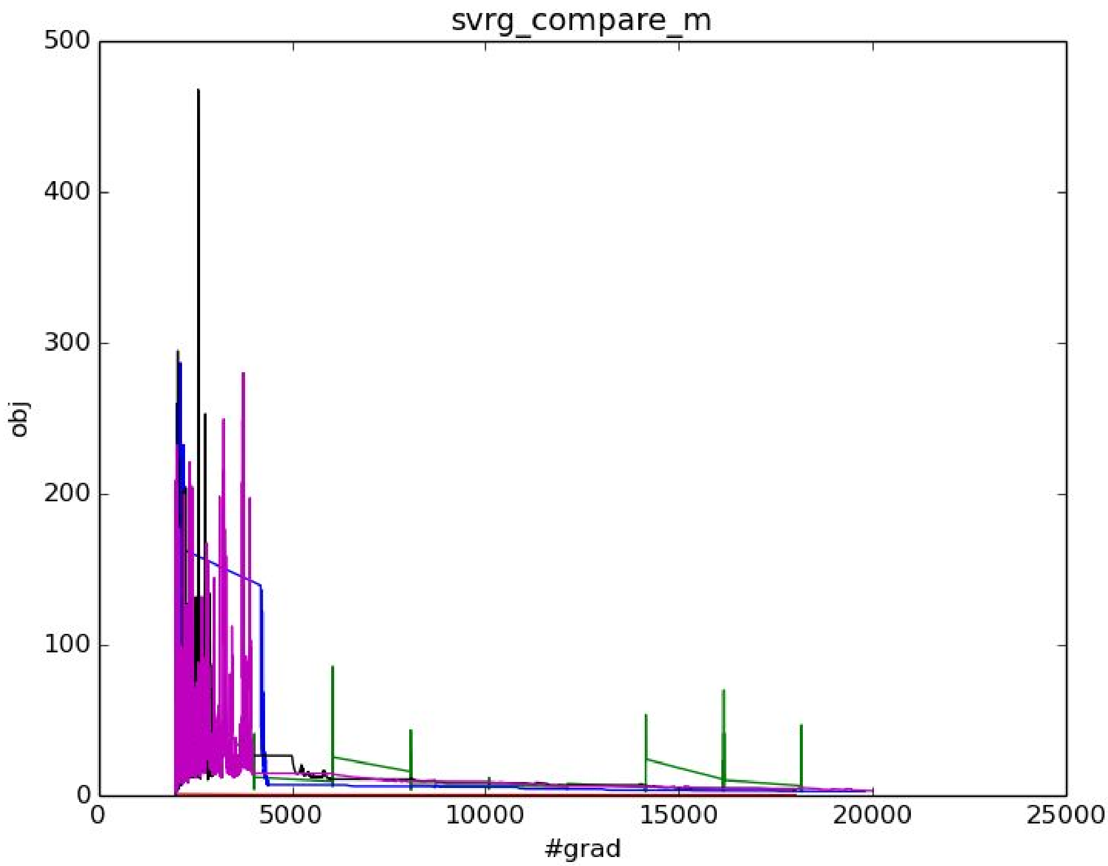
m=1000,


svrg_1000
obj
#grad
400
350
300
250
200
150
100
50
0
2000
4000
6000
8000
10000
12000
14000
16000
18000
20000

m=2000,



compare different m:
colours=['r','g','b','k','m']
m_list=[1,20,200,1000,2000]



m	Average time	std
1	5.729813576	0.4696939852
20	5.438325357	0.1948436057
200	6.271399021	0.8850801084
1000	7.408580923	0.3711026427
2000	8.346722174	0.2600766936