In [249]:

```
import opengm
import numpy

shape = (10,10)
numVar = shape[0]*shape[1]

data = numpy.random.rand(*shape)
data = numpy.round(data,1)
print data,"\n"
labelsA = (data>0.5).astype(numpy.uint32)
print labelsA
```

```
[[ 0.7  0.9  0.1  0.5  0.8  1.   0.4  0.8  0.2  0.5]
 [ 0.7  0.2  0.7  0.8  0.6  0.5  0.2  0.4  0.5  0.6]
 [ 0.8  1.   0.9  0.5  0.6  0.4  0.3  0.   0.5  1. ]
 [ 0.3  0.9  0.3  0.8  0.6  0.6  0.   0.5  0.8  0.9]
 [ 0.   0.4  0.7  0.4  0.5  0.1  0.5  0.8  0.9  0.5]
 [ 0.9  0.5  0.8  0.2  0.2  1.   0.8  0.5  0.3  0.7]
 [ 0.3  1.   0.3  0.3  0.1  0.3  0.6  0.8  0.5  0.7]
 [ 0.4  0.8  0.7  0.8  0.7  0.5  0.3  0.1  0.5  0.7]
 [ 0.7  0.7  0.1  0.6  0.3  0.5  0.7  0.7  0.9  0.5]
 [ 0.4  0.5  0.7  0.7  0.9  0.6  0.1  0.8  0.4  0.4]]

[[1 1 0 0 1 1 0 1 0 0]
 [1 0 1 1 1 0 0 0 0 1]
 [1 1 1 0 1 0 0 0 0 1]
 [0 1 0 1 1 1 0 0 1 1]
 [0 0 1 0 0 0 0 1 1 0]
 [1 0 1 0 0 1 1 0 0 1]
 [0 1 0 0 0 0 1 1 0 1]
 [0 1 1 1 1 0 0 0 0 1]
 [1 1 0 1 0 0 1 1 1 0]
 [0 0 1 1 1 1 0 1 0 0]]
```

In [250]:

```
beta = 0.3
gm = opengm.gm( [2]*numVar )
```

In [251]:

```
# add unaries
unaries = numpy.ones(shape+(2,))
unaries[:,:,0]=data
unaries[:,:,1]=1.0-data

unaryFunctionIds = gm.addFunctions(unaries.reshape(-1,2))
gm.addFactors(unaryFunctionIds,numpy.arange(numVar))
```

Out[251]: 99

In [252]:

```
pottsFunction = opengm.pottsFunction([2,2],0.0,beta)
pottsFunctionId = gm.addFunction(pottsFunction)

for x in range(shape[0]):
   for y in range(shape[0]):

        if x+1 < shape[1]:
            vi0 = y +x*shape[1]
            vi1 = y +(x+1)*shape[1]
            gm.addFactor(pottsFunctionId,[vi0,vi1])

        if x+1 < shape[1]:
            vi0 = y +x*shape[1]
            vi1 = y+1 + x*shape[1]
            gm.addFactor(pottsFunctionId,[vi0,vi1])



block4Function = numpy.zeros([2,2,2,2])
block4Function[0,0,0,0]=2.0
#block4Function[1,1,1,1]=10.0
block4FunctionId = gm.addFunction(block4Function)

for x in range(shape[0]):
   for y in range(shape[1]):
        if x+1 < shape[0] and y+1 < shape[1]:
            vi0 = y + x*shape[1]
            vi1 = y+1 + x*shape[1]
            vi2 = y + (x+1)*shape[1]
            vi3 = y+1 + (x+1)*shape[1]
            vis = [vi0,vi1,vi2,vi3]
            #gm.addFactor(block4FunctionId,vis)
```

In [253]:

```
Inf  = opengm.inference.BeliefPropagation
parameter = opengm.InfParam(steps=1000,damping=0.9,convergenceBound=0.001)
inf2 = Inf(gm,parameter=parameter)

inf2.infer()
arg=inf2.arg()
labelsB = arg.reshape(shape)
print labelsA,"\n"
print labelsB,"\n"
```

```
[[1 1 0 0 1 1 0 1 0 0]
 [1 0 1 1 1 0 0 0 0 1]
 [1 1 1 0 1 0 0 0 0 1]
 [0 1 0 1 1 1 0 0 1 1]
 [0 0 1 0 0 0 0 1 1 0]
 [1 0 1 0 0 1 1 0 0 1]
 [0 1 0 0 0 0 1 1 0 1]
 [0 1 1 1 1 0 0 0 0 1]
 [1 1 0 1 0 0 1 1 1 0]
 [0 0 1 1 1 1 0 1 0 0]]

[[1 1 1 1 1 1 1 1 1 1]
 [1 1 1 1 1 1 0 0 1 1]
 [1 1 1 1 1 1 0 0 1 1]
 [1 1 1 1 1 1 0 1 1 1]
 [1 1 1 1 1 1 1 1 1 1]
 [1 1 1 0 0 1 1 1 1 1]
 [1 1 1 0 0 1 1 1 1 1]
 [1 1 1 1 1 1 1 1 1 1]
 [1 1 1 1 1 1 1 1 1 1]
 [1 1 1 1 1 1 0 1 1 1]]
```