TFY4235/FY8904 Computational Physics, Spring 2013

Solution Set 11

Problem 1.

This problem is not so simple. That is, the *implementation* of the three algorithms, simulated annealing, tempering and genetic algorithm, is not so difficult, but the problem itself is. The problem consists in getting the algorithms to find the energy minimum of a 5×5 spin glass without getting stuck in local minima. I have set up an implimentation of the spin glass which is identical for all three codes. With the chosen variables, I find $E_{\min} = -32$ using the genetic algorithm, $E_{\min} = -32$ with simulated annealing and $E_{\min} = -14$ with simulated tempering.
The spins s_i take on the values 1 and -1. We represent them in the codes as 0 and 1 — i.e., binary variables. The spin coupling $S_i \cdot S_j$ is then rewritten $(1 - 2 * xor(S_i, S_j))$. Verify that this is correct.
In the genetic algorithm, I use a population of 200 individuals, a evolution rate of 0.75 (i.e., a probability of 0.75 to choose the individual with the least energy), a cross over rate of 0.5 and a mutation rate of 0.01. Here is the program:

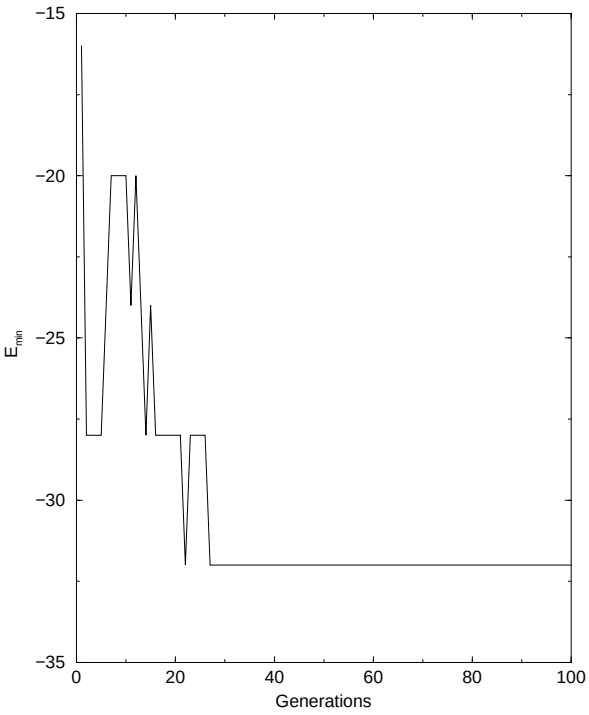
```
      program spgen
c Spin glass energy minimum with genetic algorithm
      parameter(npop=200,nsp=5,nsp2=nsp*nsp,
      c ngen=100,rmut=0.01,pvlg=0.75,qvlg=0.50)
      dimension in(npop,nsp2),nn(npop,nsp2),f(npop),
      c ibn(2,nsp2),ih(nsp),iv(nsp),jo(nsp),jn(nsp)
      ibm=5153
      do i=1,1000
      ibm=ibm*16807
      enddo
      rinv=0.5/(2.**31-1.)
      avlg=0.5+pvlg
      bvlg=0.5+qvlg
      amut=0.5+rmut
c Periodic boundaries
      do i=1,nsp
      ih(i)=i+1
      iv(i)=i-1
      jo(i)=i+1
      jn(i)=i-1
      enddo
      ih(nsp)=1
      iv( 1)=nsp
      jo(nsp)=1
      jn( 1)=nsp
```

```
c Couplings
      do k=1,2
      do ij=1,nsp2
      ibm=ibm*16807
      ibn(k,ij)=1-2*int(rinv*ibm+1.)
      enddo
      enddo
c Initializing spins
      do ipop=1,npop
      do ij=1,nsp2
      ibm=ibm*16807
      in(ipop,ij)=int(rinv*ibm+1.)
      enddo
      enddo
c Gen.  alg.  loop
      do igen=1,ngen
      fmax=-1.e8
c Calculating energy of configuration
c f is minus energy
      do ipop=1,npop
      f(ipop)=0.
      do i=1,nsp
      do j=1,nsp
      ij =i +(j -1)*nsp
      ijh=ih(i)+(j -1)*nsp
      ijo=i +(jo(j)-1)*nsp
      f(ipop)=f(ipop)-ibn(1,ij )*(1-2*xor(in(ipop,ij),in(ipop,ijh)))
      c -ibn(2,ij )*(1-2*xor(in(ipop,ij),in(ipop,ijo)))
      enddo
      enddo
      f(ipop)=-f(ipop)
      fmax=max(fmax,f(ipop))
      enddo
      fmin=-fmax
      write(*,*) igen,fmin
c selection
      do ipop=1,npop
      ibm=ibm*16807
      jpop=npop*(rinv*float(ibm)+0.5)+1
      ibm=ibm*16807
      kpop=npop*(rinv*float(ibm)+0.5)+1
      imin=jpop
      imax=kpop
      if(f(jpop).gt.f(kpop)) then
```

```
      imin=kpop
      imax=jpop
      endif
      ibm=ibm*16807
      ivlg=int(avlg-rinv*ibm)
      if(ivlg.eq.1) then
      do i=1,nsp2
      nn(ipop,i)=in(imax,i)
      enddo
      else
      do i=1,nsp2
      nn(ipop,i)=in(imin,i)
      enddo
      endif
      enddo
      do ipop=1,npop
      do i=1,nsp2
      in(ipop,i)=nn(ipop,i)
      enddo
      enddo
c crossover
      do ipop=1,npop
      ibm=ibm*16807
      ivlg=int(bvlg-rinv*ibm)
      if(ivlg.eq.1) then
      ibm=ibm*16807
      kpop=npop*(rinv*float(ibm)+0.5)+1
      ibm=ibm*16807
      jpop=npop*(rinv*float(ibm)+0.5)+1
      ibm=ibm*16807
      kstr=nsp2*(rinv*float(ibm)+0.5)+1
      do k=1,kstr
      nn(kpop,k)=in(kpop,k)
      enddo
      do k=1,kstr
      in(kpop,k)=in(jpop,k)
      enddo
      do k=1,kstr
      in(jpop,k)=nn(kpop,k)
      nn(kpop,k)=0
      enddo
      endif
      enddo
c mutations
```

```
do ipop=1,npop
do i=1,nsp2
ibm=ibm*16807
imut=int(amut-rinv*ibm)
in(ipop,i)=and(xor(in(ipop,i),imut),1)
enddo
enddo
enddo
end
```

Here is the minimum energy as function of generations:



Here is the simulated annealing program. I have used a starting temperature of 100 (that I have determined by trial and error) and a temperature reduction factor — defined as $T \to T_c$ — equal to 0.999.

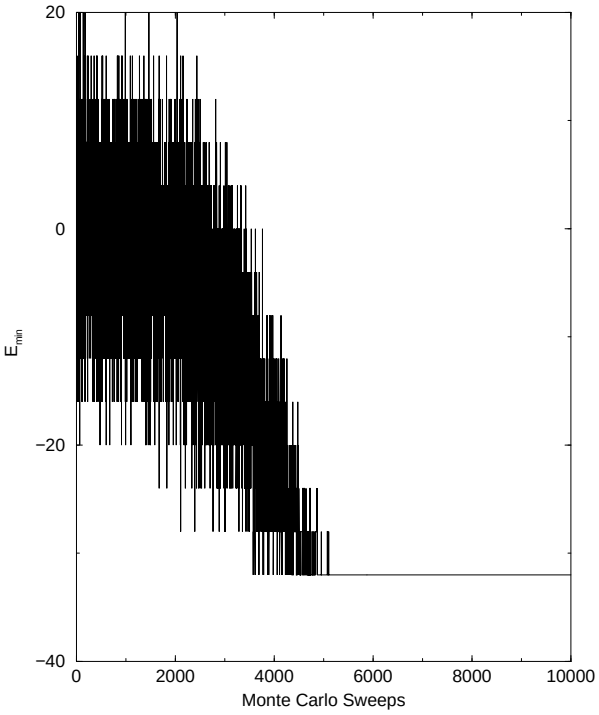
```
      program spann
c Spin glass energy minimum through simulated annealing
```

```
      parameter(nsp=5,nsp2=nsp*nsp,ngen=2000,tstart=100.,ct=0.999)
      dimension in(nsp2),ibn(2,nsp2),ih(nsp),iv(nsp),jo(nsp),jn(nsp)
      ibm=5153
      do i=1,1000
      ibm=ibm*16807
      enddo
      rinv=0.5/(2.**31-1.)
c Periodic boundaries
      do i=1,nsp
      ih(i)=i+1
      iv(i)=i-1
      jo(i)=i+1
      jn(i)=i-1
      enddo
      ih(nsp)=1
      iv( 1)=nsp
      jo(nsp)=1
      jn( 1)=nsp
c Couplings
      do k=1,2
      do ij=1,nsp2
      ibm=ibm*16807
      ran=rinv*ibm
      ibn(k,ij)=1-2*int(rinv*ibm+1.)
      enddo
      enddo
c Initializing spins
      do ij=1,nsp2
      ibm=ibm*16807
      in(ij)=int(rinv*ibm+1.)
      enddo
c Calculating energy of configuration, f
      f=0.
      do i=1,nsp
      do j=1,nsp
      ij =i +(j -1)*nsp
      ijh=ih(i)+(j -1)*nsp
      ijo=i +(jo(j)-1)*nsp
      f=f-ibn(1,ij )*(1-2*xor(in(ij),in(ijh)))
      c -ibn(2,ij )*(1-2*xor(in(ij),in(ijo)))
      enddo
      enddo
c Annealing loop
      temp=tstart
```

```
do igen=1,ngen
do it=1,nsp2
ibm=ibm*16807
ran=ibm*rinv+0.5
i=int(nsp*ran)+1
ibm=ibm*16807
ran=ibm*rinv+0.5
j=int(nsp*ran)+1
ij =i +(j -1)*nsp
ijh=ih(i)+(j -1)*nsp
ijo=i +(jo(j)-1)*nsp
ijv=iv(i)+(j -1)*nsp
ijn=i +(jn(j)-1)*nsp
io=in(ij)
im=and(not(in(ij)),1)
dm=-ibn(1,ij )*(1-2*xor(im,in(ijh)))
c -ibn(2,ij )*(1-2*xor(im,in(ijo)))
c -ibn(1,ijv)*(1-2*xor(im,in(ijv)))
c -ibn(2,ijn)*(1-2*xor(im,in(ijn)))
do=-ibn(1,ij )*(1-2*xor(io,in(ijh)))
c -ibn(2,ij )*(1-2*xor(io,in(ijo)))
c -ibn(1,ijv)*(1-2*xor(io,in(ijv)))
c -ibn(2,ijn)*(1-2*xor(io,in(ijn)))
df=dm-do
p=1.+min(exp(-df/temp),1.)
ibm=ibm*16807
ran=ibm*rinv+0.5
isw=int(p-ran)
jsw=1-isw
in(ij)=isw*im+jsw*io
df=isw*df
f=f+df
enddo
temp=temp*ct
write(*,*) igen,f
enddo
end
```

Note that I do not calculate the total energy at every update. I only need to calculate the energy change. This speeds up the program tremendously. Also note that the choices whether to update or not is done without "if" statements.

In the figure I show energy as a function of monte Carlo sweeps. It fluctuates strongly in the beginning, but then calms down.



Finally, we do simulated tempering. I recommend the original paper where the method was first suggested, E. Marinari og G. Parisi, *Simulated Tempering: A new Monte Carlo Scheme,* Europhys. Lett. **19**, 451–458 (1992) for the best description of how to practically implement the algorithm.

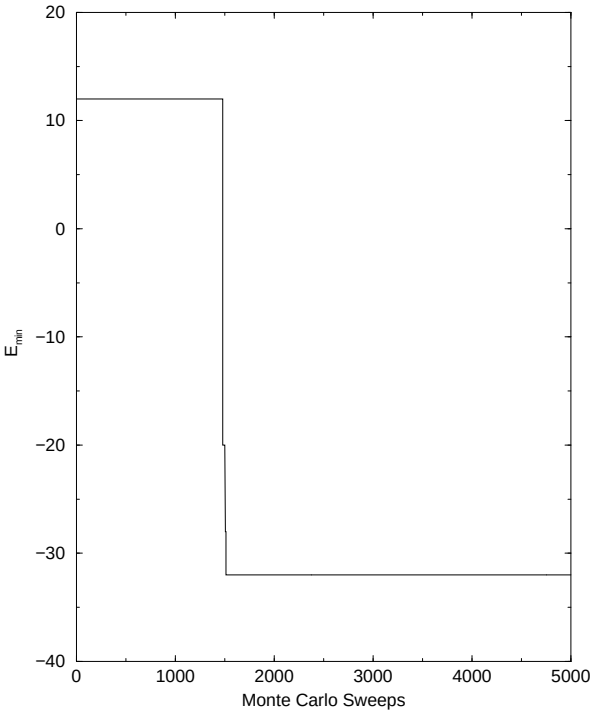
As described in the lectures, the method consists in working in an ensemble where temeprature is a variable. I have in my implementation divided the temperature interval $[0.01, 100]$ into 50 evenly spaced values T_k. To each of these temeperatures one must then associate a weight factor g_k. The optimal choice is to set g_k equal to the free energy at temperature T_k. However, this implies that we know the free energy — which we do not. Hence, we have to make another choice, which will be less efficient. In my implementation, I have simply set $g_k = 0$. I record every time the system passes through T_1, which is the lowest temperature and saves the lowest energy the energy has been so far for this temperature. The program looks as follows:

```
      program sptem
c Spinnglass energiminimum funnet med simulated tempering
      parameter(nsp=5,nsp2=nsp*nsp,ngen=20000,
      c tstart=100.,tstop=0.01,kvar=50)
      dimension in(nsp2),ibn(2,nsp2),ih(nsp),iv(nsp),jo(nsp),jn(nsp),
      c temp(kvar),fg(kvar)
      ibm=5153
      do i=1,1000
```

```
      ibm=ibm*16807
      enddo
      rinv=0.5/(2.**31-1.)
c Periodisk grensebetingelser
      do i=1,nsp
      ih(i)=i+1
      iv(i)=i-1
      jo(i)=i+1
      jn(i)=i-1
      enddo
      ih(nsp)=1
      iv( 1)=nsp
      jo(nsp)=1
      jn( 1)=nsp
c Koblingene
      do k=1,2
      do ij=1,nsp2
      ibm=ibm*16807
      ran=rinv*ibm
      ibn(k,ij)=1-2*int(rinv*ibm+1.)
      enddo
      enddo
c Initialiserer spinnene
      do ij=1,nsp2
      ibm=ibm*16807
      in(ij)=int(rinv*ibm+1.)
      enddo
c Regner ut energien for konfigurasjonen, f
      f=0.
      do i=1,nsp
      do j=1,nsp
      ij =i +(j -1)*nsp
      ijh=ih(i)+(j -1)*nsp
      ijo=i +(jo(j)-1)*nsp
      f=f-ibn(1,ij )*(1-2*xor(in(ij),in(ijh)))
      c -ibn(2,ij )*(1-2*xor(in(ij),in(ijo)))
      enddo
      enddo
      tdel=(tstart-tstop)/(kvar-1)
      do ivar=1,kvar
      temp(ivar)=tstop+(ivar-1)*tdel
      fg(ivar)=0.
      enddo
c Annealing-loekken
```

```
ivar=kvar/2
teo=temp(ivar)
fgo=fg(ivar)
fmin=nsp2/2
do igen=1,ngen
ibm=ibm*16807
ivar=ivar+1-2*int(ibm*rinv+1.0)
ivar=min(ivar,kvar)
ivar=max(ivar, 1)
ten=temp(ivar)
fgn=fg(ivar)
do it=1,nsp2
ibm=ibm*16807
ran=ibm*rinv+0.5
i=int(nsp*ran)+1
ibm=ibm*16807
ran=ibm*rinv+0.5
j=int(nsp*ran)+1
ij =i +(j -1)*nsp
ijh=ih(i)+(j -1)*nsp
ijo=i +(jo(j)-1)*nsp
ijv=iv(i)+(j -1)*nsp
ijn=i +(jn(j)-1)*nsp
p=1.+min(exp(-df/temp(ivar)-(fgo-fgn)),1.)
f=f+df
fgo=fgn
teo=ten
enddo
if(ivar.eq.1) fmin=min(fmin,f)
write(*,*) igen,fmin
enddo
end
```

In the following figure, I show $E_{\min}$as a function of Monte Carlo sweeps.


20
10
0
−10
−20
−30
−40
E_{min}
0
1000
2000
3000
4000
5000
Monte Carlo Sweeps