ParadisEO – PEO : Lesson 4

Note : All the components are not presented in this lesson (binary, topology, asynchronous or synchronous ...). To know the completeness of components refer to API documentation of ParadisEO – EO and ParadisEO – PEO.

Requirements

Before to start this lesson 4, you should read and execute **Lesson 3**.

Of course, to execute the lesson 4, you should be in the directory of this lesson.

Problem

In the lesson 4 you can execute two different algorithms with a parallel evaluation function, a parallel transformation and an island model :

- Particle Swarm Optimization (PSO)
- Evolutionary Algorithm (EA)

The problem is the same in the two cases : minimizing the Rosenbrock function.

$$f(x_1,x_2)=100*(x_2-x_1^2)^2+(1-x_1)^2$$

The optimal is :

$$f(x_1,x_2)=0$$

with : $X=(1,1)$

PSO (mainPSO.cpp) :

```
#include <peo>
typedef eoRealParticle < double >Indi;
double f (const Indi & _indi)
```

```
{
    double sum;
    sum=_indi[1]-pow(_indi[0],2);
    sum=100*pow(sum,2);
    sum+=pow((1-_indi[0]),2);
    return (-sum);
}
int main (int __argc, char *__argv[])
{
// In this lesson, we can see an example of a PSO with three islands.
// The strategy of migration is the replacement.
// The evaluation is parallel.
    peo :: init( __argc, __argv );
    const unsigned int VEC_SIZE = 2;
    const unsigned int POP_SIZE = 20;
    const unsigned int NEIGHBORHOOD_SIZE= 6;
    const unsigned int MAX_GEN = 150;
    const double INIT_POSITION_MIN = -2.0;
    const double INIT_POSITION_MAX = 2.0;
    const double INIT_VELOCITY_MIN = -1.;
    const double INIT_VELOCITY_MAX = 1.;
    const double C1 = 0.5;
    const double C2 = 2.;
    const double C3 = 2.;
    const unsigned int  MIG_FREQ = 10;
    rng.reseed (time(0));
    RingTopology topologyMig;
    peoEvalFunc<Indi, double, const Indi& > plainEval(f);
    peoParaPopEval< Indi > eval(plainEval);
    eoUniformGenerator < double >uGen (INIT_POSITION_MIN, INIT_POSITION_MAX);
    eoInitFixedLength < Indi > random (VEC_SIZE, uGen);
    eoUniformGenerator < double >sGen (INIT_VELOCITY_MIN, INIT_VELOCITY_MAX);
    eoVelocityInitFixedLength < Indi > veloRandom (VEC_SIZE, sGen);
    eoFirstIsBestInit < Indi > localInit;
    eoRealVectorBounds bndsFlight(VEC_SIZE,INIT_POSITION_MIN,INIT_POSITION_MAX);
    eoStandardFlight < Indi > flight(bndsFlight);
    eoPop < Indi > pop;
    pop.append (POP_SIZE, random);
    peoInitializer <Indi> init(eval,veloRandom,localInit,pop);
    eoLinearTopology<Indi> topology(NEIGHBORHOOD_SIZE);
    eoRealVectorBounds bnds(VEC_SIZE,INIT_VELOCITY_MIN,INIT_VELOCITY_MAX);
    eoStandardVelocity < Indi > velocity (topology,C1,C2,bnds);
    eoGenContinue < Indi > genContPara (MAX_GEN);
    eoCheckPoint<Indi> checkpoint(genContPara);
    eoPeriodicContinue< Indi > mig_cont( MIG_FREQ );
    peoPSOSelect<Indi> mig_selec(topology);
    eoSelectNumber< Indi > mig_select(mig_selec);
    peoPSOReplacement<Indi> mig_replace;
    peoEvalFunc<Indi, double, const Indi& > plainEval2(f);
    peoParaPopEval< Indi > eval2(plainEval2);
    eoUniformGenerator < double >uGen2 (INIT_POSITION_MIN, INIT_POSITION_MAX);
    eoInitFixedLength < Indi > random2 (VEC_SIZE, uGen2);
    eoUniformGenerator < double >sGen2 (INIT_VELOCITY_MIN, INIT_VELOCITY_MAX);
    eoVelocityInitFixedLength < Indi > veloRandom2 (VEC_SIZE, sGen2);
    eoFirstIsBestInit < Indi > localInit2;
    eoRealVectorBounds
    bndsFlight2(VEC_SIZE,INIT_POSITION_MIN,INIT_POSITION_MAX);
    eoStandardFlight < Indi > flight2(bndsFlight2);
    eoPop < Indi > pop2;
    pop2.append (POP_SIZE, random2);
    peoInitializer <Indi> init2(eval2,veloRandom2,localInit2,pop2);
    eoLinearTopology<Indi> topology2(NEIGHBORHOOD_SIZE);
```

```
    eoRealVectorBounds bnds2(VEC_SIZE,INIT_VELOCITY_MIN,INIT_VELOCITY_MAX);
    eoStandardVelocity < Indi > velocity2 (topology2,C1,C2,bnds2);
    eoGenContinue < Indi > genContPara2 (MAX_GEN);
    eoCheckPoint<Indi> checkpoint2(genContPara2);
    eoPeriodicContinue< Indi > mig_cont2( MIG_FREQ );
    peoPSOSelect<Indi> mig_selec2(topology2);
    eoSelectNumber< Indi > mig_select2(mig_selec2);
    peoPSOReplacement<Indi> mig_replace2;
    peoEvalFunc<Indi, double, const Indi& > plainEval3(f);
    peoParaPopEval< Indi > eval3(plainEval3);
    eoUniformGenerator < double >uGen3 (INIT_POSITION_MIN, INIT_POSITION_MAX);
    eoInitFixedLength < Indi > random3 (VEC_SIZE, uGen3);
    eoUniformGenerator < double >sGen3 (INIT_VELOCITY_MIN, INIT_VELOCITY_MAX);
    eoVelocityInitFixedLength < Indi > veloRandom3 (VEC_SIZE, sGen3);
    eoFirstIsBestInit < Indi > localInit3;
    eoRealVectorBounds
    bndsFlight3(VEC_SIZE,INIT_POSITION_MIN,INIT_POSITION_MAX);
    eoStandardFlight < Indi > flight3(bndsFlight3);
    eoPop < Indi > pop3;
    pop3.append (POP_SIZE, random3);
    peoInitializer <Indi> init3(eval3,veloRandom3,localInit3,pop3);
    eoLinearTopology<Indi> topology3(NEIGHBORHOOD_SIZE);
    eoRealVectorBounds bnds3(VEC_SIZE,INIT_VELOCITY_MIN,INIT_VELOCITY_MAX);
    eoStandardVelocity < Indi > velocity3 (topology3,C1,C2,bnds3);
    eoGenContinue < Indi > genContPara3 (MAX_GEN);
    eoCheckPoint<Indi> checkpoint3(genContPara3);
    eoPeriodicContinue< Indi > mig_cont3( MIG_FREQ );
    peoPSOSelect<Indi> mig_selec3(topology3);
    eoSelectNumber< Indi > mig_select3(mig_selec3);
    peoPSOReplacement<Indi> mig_replace3;
    peoAsyncIslandMig< Indi > mig( mig_cont, mig_select, mig_replace,
topologyMig, pop, pop2);
    checkpoint.add( mig );
    peoAsyncIslandMig< Indi > mig2( mig_cont2, mig_select2, mig_replace2,
topologyMig, pop2, pop3);
    checkpoint2.add( mig2 );
    peoAsyncIslandMig< Indi > mig3( mig_cont3, mig_select3, mig_replace3,
topologyMig, pop3, pop);
    checkpoint3.add( mig3 );
    peoPSO < Indi > psa(init,checkpoint, eval, velocity, flight);
    mig.setOwner( psa );
    psa(pop);
    peoPSO < Indi > psa2(init2,checkpoint2, eval2, velocity2, flight2);
    mig2.setOwner( psa2 );
    psa2(pop2);
    peoPSO < Indi > psa3(init3,checkpoint3, eval3, velocity3, flight3);
    mig3.setOwner( psa3 );
    psa3(pop3);

    peo :: run();
    peo :: finalize();
    if(getNodeRank()==1)
    {
            std::cout << "Population 1 :\n" << pop << std::endl;
      std::cout << "Population 2 :\n" << pop2 << std::endl;
      std::cout << "Population 3 :\n" << pop3 << std::endl;
    }
}
```

Launching the program

Your file should be called mainPSO.cpp or mainEA.cpp - please make sure you do not rename the file (we will be using a pre-built makefile, thus you are required not to change the file names). Please make sure you are in the paradiseo-peo/tutorial/build/Lesson4 directory - you should open a console and you should change your current directory to the one of Lesson4.

Compilation :

- make
- make install

Execution (ie Technical Introduction):

mpiexec -n 4 ./pso @param

or

mpiexec -n 4 ./ea @param