ParadisEO – PEO : Lesson 2

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.

Introduction

By using an evolutionary algorithm, you will be able to use a parallel crossover and a parallel mutation,

Requirements

Before to start this lesson 2, you should read and execute **Lesson2**.

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

Problem

In the lesson 2 you can execute one algorithm with a parallel transformation operators (crossover and mutation) :

- Evolutionary Algorithm (EA)

The problem is : 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 : $X=(1,1)$

EA (mainEA.cpp) :

```
#include <peo>
#include <es.h>
typedef eoReal<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[])
{

    peo :: init( __argc, __argv );
    const unsigned int VEC_SIZE = 2;
    const unsigned int POP_SIZE = 20;
    const unsigned int MAX_GEN = 300;
    const double INIT_POSITION_MIN = -2.0;
    const double INIT_POSITION_MAX = 2.0;
    const float CROSS_RATE = 0.8;
    const double EPSILON = 0.01;
    const float MUT_RATE = 0.3;
    rng.reseed (time(0));
    eoGenContinue < Indi > genContPara (MAX_GEN);
    eoCombinedContinue <Indi> continuatorPara (genContPara);
    eoCheckPoint<Indi> checkpoint(continuatorPara);
    peoEvalFunc<Indi> plainEval(f);
    peoSeqPopEval< Indi > eval(plainEval);  // Here, the evaluation is
                                            // sequential
    eoUniformGenerator < double >uGen (INIT_POSITION_MIN, INIT_POSITION_MAX);
    eoInitFixedLength < Indi > random (VEC_SIZE, uGen);
    eoRankingSelect<Indi> selectionStrategy;
    eoSelectNumber<Indi> select(selectionStrategy,POP_SIZE);
    eoSegmentCrossover<Indi> crossover;
    eoUniformMutation<Indi>  mutation(EPSILON);

 /******************************************************************************/

 /* In this lesson, you can choose between  :
  *
  *    - A sequential transformation (crossover + mutation) :
  *
  *   eoSGATransform<Indi> transform(crossover,CROSS_RATE,mutation,MUT_RATE);
  *
  *   peoSeqTransform<Indi> eaTransform(transform);
  *
  *   OR
  *
  *    - A parallel transformation (crossover + mutation) :
  *
  * peoParaSGATransform <Indi> eaTransform(crossover,CROSS_RATE,mutation,MUT_RATE);
  *
  *  Unfortunately, if you don't use a crossover which creates two children with
  * two parents,
  *  you can't use this operator.
  *  In this case, you should send a mail to : paradiseo-help@lists.gforge.inria.fr
  */
```

```
    peoParaSGATransform <Indi>
    eaTransform(crossover,CROSS_RATE,mutation,MUT_RATE);

/******************************************************************************/

    eoPlusReplacement<Indi> replace;
    eoPop < Indi > pop;
    pop.append (POP_SIZE, random);
    peoEA<Indi> Algo(checkpoint,eval,select,eaTransform,replace);
    Algo(pop);
    peo :: run();
    peo :: finalize();
    if(getNodeRank()==1)
            std::cout << "Final population :\n" << pop << std::endl;
}
```

Launching the program

Your file should be called 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/Lesson2 directory - you should open a console and you should change your current directory to the one of Lesson2.

Compilation :

- make
- make install

Execution (ie Technical Introduction):

mpiexec -n 4 ./ea @param