Para hacerlo correr

1. Llvm tiene que estar instalado

```
    $ clang ––version
    Debian clang version 10.0.1-++20200529024432+a634a80615b-
    1~exp1~20200529005028.166
    Target: x86_64-pc-linux-gnu
    Thread model: posix
    InstalledDir: /usr/bin
```

10.0.1 → tiene que ser mínimo la versión 9
/usr/bin → es importante saber donde está instalado llvm

```
    Para asegurar comprueba que existen estos 3 ejecutables:
        $ clang --version
        $ opt --version
        $ llc --version

    Los 3 tienen que estar instalados en la misma carpeta y tienen que llamarse
exactamente así. Si los ejecutables se llaman de otra forma p.e. clang-9, opt-9,
etc lo mas sencillo es hacer enlaces simbólicos a esos ejecutables:
    /usr/bin$ ln –s clang-9 clang
    /usr/bin$ ln –s opt-9 opt
    /usr/bin$ ln –s llc-9 llc
```

2. En llvmMultiobjetiveProblem.py, línea 16:

```
self.llvm = LlvmUtils(
    llvmpath='/usr/bin/',
    source="polybench_small/polybench_small_original.bc" ,
    jobid='llvm_multiobjetive',
    useperf=False)
```

/usr/bin/ → carpeta de instalación de llvm. Tiene que acabar en /
polybench_small/polybench_small_original.bc → este fichero tiene que existir

3. Ejecutar algoritmo genético:

```
    $ pyhton3 main_llvm_multiobjetive.py
```

Teniendo llvm 9+ instalado y la ruta de instalación bien puesta (con la barra / del final) tiene que funcionar.

Para añadir nuevas métricas

1. Añadir a LlvmUtils.py una nueva función que mida algo del .ll o .bc optimizado. En este caso va a ser una función tonta que siempre devuelva el mismo valor:

```
    def get_test(self):
        return 1;
```

2. En llvmMultiobjetiveProblem.py

Localizar estas líneas:

```
20      self.obj_directions = [self.MINIMIZE, self.MAXIMIZE]
21      self.obj_labels = ['runtime', 'codelines']
22      self.number_of_objectives = 2
```

Y modificarlas para que admitan un nuevo objetivo

```
20      self.obj_directions = [self.MINIMIZE, self.MAXIMIZE, self.MAXIMIZE]
21      self.obj_labels = ['runtime', 'codelines', 'test']
22      self.number_of_objectives = 3
```

En este caso se ha añadido un nuevo objetivo de MAXIMIZACION (como las codelines) llamado ‘test’. Podría haber sido de minimización (como el runtime) y llamarse de cualquier otra forma. Importante que el number_of_objetives ha pasado de 2 a 3.

3. En llvmMultiobjetiveProblem.py

Localizar estas líneas:

```
60      solution.objectives[0] = self.llvm.get_codelines(passes=passes)
61      solution.objectives[1] = self.llvm.get_runtime(passes=passes)
62      self.dictionary.update({key: solution.objectives})
63  else:
64      solution.objectives[0] = value[0]
65      solution.objectives[1] = value[1]
```

Añadir el tercer objetivo:

```
60      solution.objectives[0] = self.llvm.get_codelines(passes=passes)
61      solution.objectives[1] = self.llvm.get_runtime(passes=passes)
62      solution.objectives[2] = self.llvm.get_test()
63      self.dictionary.update({key: solution.objectives})
64  else:
65      solution.objectives[0] = value[0]
66      solution.objectives[1] = value[1]
67      solution.objectives[2] = value[2]
```

4. Ejecutar main

```
$ pyhton3 main_llvm_multiobjetive.py
```

```
evaluated solution 1 from epoch 1 : variables=[4,46,17,54,10,10,43,62,58,52],fitness=[0.1320042610168457,6002,1]'
evaluated solution 2 from epoch 1 : variables=[63,84,53,84,36,46,52,9,80,42],fitness=[0.15546941757202148,5270,1]'
evaluated solution 3 from epoch 1 : variables=[64,4,78,41,73,81,12,18,56,10],fitness=[0.1372377872467041,5167,1]'
…
Settings:
     Algorithm: NSGAII
     Problem: Llvm Multiobjetive Problem
     Computing time: 9.094732522964478 seconds
     Max evaluations: 20
     Population size: 10
     Offspring population size: 10
     Probability mutation: 0.1
     Probability crossover: 0.3
     Solution length: 10
     Opt executed one by one: 0 times

Results:
          [48, 30, 84, 81, 21, 3, 32, 23, 48, 47]      [0.14612269401550293, 5262, 1]
          [35, 73, 25, 24, 49, 71, 71, 9, 77, 21]      [0.156358003616333, 5005, 1]
          [72, 41, 2, 6, 45, 67, 22, 45, 60, 42]       [0.14722251892089844, 5231, 1]
          [77, 71, 6, 73, 50, 18, 81, 4, 13, 3]        [0.14569497108459473, 5263, 1]
          [77, 71, 8, 20, 50, 72, 81, 7, 47, 4]        [0.15433907508850098, 5067, 1]
```

El 1 del final es el valor que está devolviendo *get_test* por cada evaluación que hace el algoritmo genético.

Para programar una fución que haga algo más que devolver siempre 1 fijarse en get_runtime y get_codelines e inspirarse para buscar nuevas métricas.

En Results en algunas ejecuciones salen mas soluciones, en otras ejecuciones salen menos. Eso es normal.