MEMORIA

Evaluación de Algoritmos Evolutivos

**Grupo 7**

Jaime Díez-Hochleitner Suárez

Luis Domínguez Romero

**Introducción**

En esta memoria veremos los datos resultantes de la experimentación con las 11 funciones de benchmarking del paquete *SOCO2011.* Hemos ejecutado 3 algoritmos distintos: el algoritmo de Evolución Diferencial que nosotros hemos implementado, el DE de la biblioteca *Scipy* y el SADE de la biblioteca *Pyade*. En nuestro caso, no hemos querido añadir ningún Algoritmo Genético ya que lo que buscamos con esta evaluación de algoritmos es ver la eficacia de nuestra implementación con respecto a algoritmos de la misma índole.

La ejecución de ambos DE’s se ha realizado con un factor de mutación F=0.5 y un factor de cruce Cr=0.5 con estrategias de mutación **de/best/2** y de cruce **binomial**. En todos los algoritmos la población inicial tiene un tamaño de 25 individuos con 10 genes por individuo, cuyos valores rondan entre [0, 10]. El número de iteraciones máximas sobre cada función es de 100 y cada algoritmo se ejecuta 10 veces sobre cada función de benchmark.

**Resultados de la ejecución**

*Función Sphere*

def \_sphere(x):  
 return np.sum(np.power(x, 2))  
  
def sphere(x):  
 return f\_wrap(x, \_sphere)

|  |  |  |  |
| --- | --- | --- | --- |
| **Ejecución** | **Sphere DE**  **(Nuestro)** | **Sphere DE**  **(Scipy)** | **Sphere SADE**  **(Pyade)** |
| **1** | 3.7511972338327175 | 0.0 | 0.00242593487413689 |
| **2** | 0.6747256113582627 | 0.0 | 0.0031722994641388154 |
| **3** | 0.24265402641538952 | 0.0 | 0.0024976900923057285 |
| **4** | 8.814826221348667 | 0.0 | 0.004352575439392395 |
| **5** | 1.328444197097738 | 0.0 | 0.001923501980190291 |
| **6** | 0.000812309596253574 | 0.0 | 0.001402370153307329 |
| **7** | 0.41475018503875244 | 0.0 | 0.003861735998482732 |
| **8** | 3.0897599271968663 | 0.0 | 0.005413986253516438 |
| **9** | 0.621826416826631 | 0.0 | 0.0035321570738812316 |
| **10** | 0.7453245340639049 | 0.0 | 0.004104829153986344 |
| **Media** | 1.9684320662775183 | 0.0 | 0.0032687080483338195 |
| **Desv. Típica** | 2.707296830449243 | 0.0 | 0.0012249262822273226 |
| **Mediana** | 0.7100250727110837 | 0.0 | 0.0033522282690100235 |
| **Min** | 0.000812309596253574 | 0.0 | 0.001402370153307329 |
| **Max** | 8.814826221348667 | 0.0 | 0.005413986253516438 |

*Función Ackley*

def \_ackley(x):  
 dim = len(x)  
 sum1 = 0.0  
 sum2 = 0.0  
  
 for n in range(0, dim):  
 z = np.abs(x[n])  
 sum1 += pow(z, 2)  
 sum2 += np.cos(2 \* np.pi \* z)  
  
 return -20 \* np.exp(-0.2 \* np.sqrt(sum1 / dim)) - np.exp(sum2 / dim) + 20 + np.e

|  |  |  |  |
| --- | --- | --- | --- |
| **Ejecución** | **Ackley DE**  **(Nuestro)** | **Ackley DE**  **(Scipy)** | **Ackley SADE**  **(Pyade)** |
| **1** | 4.440892098500626e-16 | 4.440892098500626e-16 | 0.14034689190248928 |
| **2** | 8.425116087476866 | 2.0133152362567164 | 0.08297121231751037 |
| **3** | 0.9459286561463398 | 4.440892098500626e-16 | 0.05883931180723012 |
| **4** | 4.440892098500626e-16 | 4.440892098500626e-16 | 0.12411213677860689 |
| **5** | 1.5969879387359183 | 4.440892098500626e-16 | 0.07747204599951418 |
| **6** | 1.1790169272030693 | 1.1551485027098392 | 0.07384239467454679 |
| **7** | 3.9098638329822575 | 4.440892098500626e-16 | 0.07691115961356987 |
| **8** | 3.847899178708421 | 4.440892098500626e-16 | 0.06590403385561361 |
| **9** | 0.685792417032197 | 4.440892098500626e-16 | 0.17211387560244562 |
| **10** | 1.6462816573712442 | 4.440892098500626e-16 | 0.058364571373136886 |
| **Media** | 2.2236886695656315 | 0.31684637389665593 | 0.09308776339246636 |
| **Desv. Típica** | 2.574313680406608 | 0.6979247060107421 | 0.038782781111010314 |
| **Mediana** | 1.3880024329694938 | 4.440892098500626e-16 | 0.07719160280654203 |
| **Min** | 4.440892098500626e-16 | 4.440892098500626e-16 | 0.058364571373136886 |
| **Max** | 8.425116087476866 | 2.0133152362567164 | 0.17211387560244562 |

*Función Rosenbrock*

def \_rosenbrock(x):  
 F = 0.0  
 z = [abs(x[n] + 1) for n in range(len(x))]  
  
 for n in range(0, len(x) - 1):  
 F += 100 \* (pow((pow(z[n], 2) - z[n + 1]), 2)) + pow((z[n] - 1), 2)  
  
 return F

|  |  |  |  |
| --- | --- | --- | --- |
| **Ejecución** | **Rosenbrock DE**  **(Nuestro)** | **Rosenbrock DE**  **(Scipy)** | **Rosenbrock SADE (Pyade)** |
| **1** | 0.0 | 0.0 | 10.301449701221445 |
| **2** | 2098.8385671510773 | 1.855302906961063e-12 | 15.446182617920314 |
| **3** | 3137.9553371377838 | 1.0309396407475561e-10 | 1.1276176672901217 |
| **4** | 590.8008956417589 | 4.4504316133711224e-11 | 9.503977666543145 |
| **5** | 0.0 | 4.238214202464481e-11 | 0.2626367514440256 |
| **6** | 563.6313022480872 | 0.0 | 27.869594204211158 |
| **7** | 397.0828220876558 | 8.112318774587342e-11 | 3.1575562063765417 |
| **8** | 1106.040267290261 | 3.4725192045557266e-11 | 15.708681609215533 |
| **9** | 1891.6494039989507 | 5.442159495825945e-13 | 2.121024516710942 |
| **10** | 879.5983869308797 | 4.283807068660394e-11 | 25.33275131465956 |
| **Media** | 1066.5596982486454 | 3.5106639156768996e-11 | 11.083147225559278 |
| **Desv. Típica** | 1015.2256291704297 | 3.606496235901961e-11 | 9.931449176485712 |
| **Mediana** | 735.1996412863193 | 3.855366703510104e-11 | 9.902713683882295 |
| **Min** | 0.0 | 0.0 | 0.2626367514440256 |
| **Max** | 3137.9553371377838 | 1.0309396407475561e-10 | 27.869594204211158 |

*Función Rastrigin*

def \_rastrigin(x):  
 F = 0.0  
 for n in range(0, len(x)):  
 z = x[n]  
 F += (pow(z, 2) - 10 \* np.cos(2 \* np.pi \* z) + 10)  
  
 return F

|  |  |  |  |
| --- | --- | --- | --- |
| **Ejecución** | **Rastrigin DE**  **(Nuestro)** | **Rastrigin DE**  **(Scipy)** | **Rastrigin SADE (Pyade)** |
| **1** | 20.32317317171118 | 12.934462704394257 | 15.531581426878988 |
| **2** | 16.442137000583955 | 21.889058908062694 | 19.949258941467306 |
| **3** | 17.504830867070815 | 19.899145831695 | 19.79711636471839 |
| **4** | 14.044318824852677 | 11.939498609481534 | 18.49992852610082 |
| **5** | 35.442073202252416 | 28.853701459037655 | 20.518963254356862 |
| **6** | 17.694989996875663 | 18.904201933496257 | 7.714984978826578 |
| **7** | 9.976246108176861 | 8.954626476159767 | 6.753942021699379 |
| **8** | 2.526263150985921 | 11.939498609481378 | 12.903070151202977 |
| **9** | 0.23675212249573008 | 20.894120047689096 | 17.626901818879077 |
| **10** | 16.23446083163532 | 15.919329800040812 | 13.340564522390434 |
| **Media** | 15.042524527664053 | 17.212764437953844 | 15.263631200652082 |
| **Desv. Típica** | 9.792493020688815 | 6.007379499847858 | 4.9991541439227705 |
| **Mediana** | 16.338298916109636 | 17.411765866768533 | 16.57924162287903 |
| **Min** | 0.23675212249573008 | 8.954626476159767 | 6.753942021699379 |
| **Max** | 35.442073202252416 | 28.853701459037655 | 20.518963254356862 |

*Función Griewank*

def \_griewank(x):  
 F1 = 0.0  
 F2 = 0.0  
  
 for n in range(0, len(x)):  
 z = abs(x[n])  
 F1 += (pow(z, 2) / 4000)  
 F2 \*= (np.cos(z / np.sqrt(n + 1)))  
  
 return F1 - F2 + 1

|  |  |  |  |
| --- | --- | --- | --- |
| **Ejecución** | **Griewank DE**  **(Nuestro)** | **Griewank DE**  **(Scipy)** | **Griewank SADE (Pyade)** |
| **1** | 1.0 | 1.0000000000015006 | 1.0000208515442737 |
| **2** | 1.00021894788018 | 1.0 | 1.0000296468701984 |
| **3** | 1.0000646517494796 | 1.0 | 1.00004017738462 |
| **4** | 1.0 | 1.0 | 1.0000133668430053 |
| **5** | 1.0002280712536988 | 1.0000000000000258 | 1.0000187258359332 |
| **6** | 1.0 | 1.0000000000000062 | 1.0000325839032633 |
| **7** | 1.001916088852224 | 1.0000000000000178 | 1.000064324009855 |
| **8** | 1.0000552649845997 | 1.0000000000004197 | 1.0000416866753854 |
| **9** | 1.001281171864636 | 1.0000000000003084 | 1.000053404240749 |
| **10** | 1.0003968692968699 | 1.0 | 1.000030726381874 |
| **Media** | 1.0004161065881687 | 1.0000000000002278 | 1.0000345493689158 |
| **Desv. Típica** | 0.0006537111905801868 | 4.719793740640453e-13 | 1.582500805006946e-05 |
| **Mediana** | 1.0001417998148296 | 1.000000000000012 | 1.0000316551425685 |
| **Min** | 1.0 | 1.0 | 1.0000133668430053 |
| **Max** | 1.001916088852224 | 1.0000000000015006 | 1.000064324009855 |

*Función Schwefel\_2\_21*

def \_schwefel\_2\_21(x):  
 F = abs(x[0])  
  
 for n in range(1, len(x)):  
 z = x[n]  
 F = max(F, abs(z))  
  
 return F

|  |  |  |  |
| --- | --- | --- | --- |
| **Ejecución** | **Schwefel\_2\_21 DE (Nuestro)** | **Schwefel\_2\_21 DE (Scipy)** | **Schwefel\_2\_21 SADE (Pyade)** |
| **1** | 0.0 | 0.0 | 0.0483094655102336 |
| **2** | 1.0841182707202182 | 0.0 | 0.03744133025322616 |
| **3** | 1.9753947610574754 | 0.0 | 0.05575811493683536 |
| **4** | 1.9479692010147027 | 0.0 | 0.056071185306722254 |
| **5** | 1.8355479161106507 | 0.0 | 0.03877368543504956 |
| **6** | 2.3868664887400746 | 0.0 | 0.05680902006944151 |
| **7** | 1.1505953343332669 | 0.0 | 0.05891281316728579 |
| **8** | 2.4496860011347152 | 0.0 | 0.0622690874397138 |
| **9** | 0.4641165969902521 | 0.0 | 0.050280852400073586 |
| **10** | 3.4107443792014145 | 0.0 | 0.057802190025164964 |
| **Media** | 1.670503894930277 | 0.0 | 0.052242774454374655 |
| **Desv. Típica** | 1.0113812675995564 | 0.0 | 0.008456474076292119 |
| **Mediana** | 1.8917585585626768 | 0.0 | 0.055914650121778806 |
| **Min** | 0.0 | 0.0 | 0.03744133025322616 |
| **Max** | 3.410744379201414 | 0.0 | 0.0622690874397138 |

*Función Schwefel\_2\_22*

def \_schwefel\_2\_22(x):  
 sum\_ = 0.0  
 prod = 1.0  
  
 for n in range(0, len(x)):  
 val = abs(x[n])   
 sum\_ += val  
 prod \*= val  
  
 return sum\_ + prod

|  |  |  |  |
| --- | --- | --- | --- |
| **Ejecución** | **Schwefel\_2\_22 DE (Nuestro)** | **Schwefel\_2\_22 DE (Scipy)** | **Schwefel\_2\_22 SADE (Pyade)** |
| **1** | 0.0 | 0.0 | 0.18172997994943899 |
| **2** | 0.0 | 0.0 | 0.17833965030761742 |
| **3** | 2.522663494673343 | 0.0 | 0.1998838796837818 |
| **4** | 1.2210507001724045 | 0.0 | 0.13526722972097305 |
| **5** | 2.733207004424902 | 0.0 | 0.1495540963988055 |
| **6** | 0.0 | 0.0 | 0.10355907348238466 |
| **7** | 4.180698423399851 | 0.0 | 0.21506666133979485 |
| **8** | 5.671641967515823 | 0.0 | 0.21353854627802585 |
| **9** | 1.9910792013515124 | 0.0 | 0.2061121955020195 |
| **10** | 6.3922764273247195 | 0.0 | 0.17948466208605368 |
| **Media** | 2.4712617218862554 | 0.0 | 0.17625359747488953 |
| **Desv. Típica** | 2.3267819382303485 | 0.0 | 0.036619013162668655 |
| **Mediana** | 2.2568713480124276 | 0.0 | 0.18060732101774635 |
| **Min** | 0.0 | 0.0 | 0.10355907348238466 |
| **Max** | 6.3922764273247195 | 0.0 | 0.21506666133979485 |

*Función Schwefel\_1\_2*

def \_schwefel\_1\_2(x):  
 sum\_ = 0.0  
 val = 0.0  
  
 for n in range(0, len(x)):  
 val += x[n]  
 sum\_ += val \* val  
  
 return sum\_

|  |  |  |  |
| --- | --- | --- | --- |
| **Ejecución** | **Schwefel\_1\_2 DE (Nuestro)** | **Schwefel\_1\_2 DE (Scipy)** | **Schwefel\_1\_2 SADE (Pyade)** |
| **1** | 0.0 | 0.0 | 0.03317674863535275 |
| **2** | 1.2159393239737075 | 0.0 | 0.08711055893066344 |
| **3** | 81.61993270666179 | 0.0 | 0.052742076139519464 |
| **4** | 0.07548852286543478 | 0.0 | 0.014674303256267942 |
| **5** | 166.18704832556082 | 0.0 | 0.02171212138091982 |
| **6** | 0.7499681933612251 | 0.0 | 0.04870840609294834 |
| **7** | 8.972776666439817 | 0.0 | 0.07585459311138136 |
| **8** | 69.5020030283278 | 0.0 | 0.06569378760965448 |
| **9** | 85.49892461714592 | 0.0 | 0.04071347938389759 |
| **10** | 2.707220614638909 | 0.0 | 0.01369613850052744 |
| **Media** | 41.65293019989754 | 0.0 | 0.04540822130411326 |
| **Desv. Típica** | 56.92174302718393 | 0.0 | 0.025441942530868546 |
| **Mediana** | 5.839998640539363 | 0.0 | 0.044710942738422965 |
| **Min** | 0.0 | 0.0 | 0.013696138500527442 |
| **Max** | 166.18704832556082 | 0.0 | 0.08711055893066344 |

*Función Extended f\_10*

def f\_10(x, y):  
 p = (x\*x + y\*y)  
 z = pow(p, 0.25)  
 t = np.sin(50.0 \* pow(p, 0.1))  
 t = t\*t + 1.0  
  
 return z\*t  
  
def \_extended\_f\_10(x):  
 sum\_ = f\_10(x[len(x)-1], x[0])  
  
 for n in range(0, len(x)-1):  
 sum\_ += f\_10(x[n], x[n+1])  
  
 return sum\_

|  |  |  |  |
| --- | --- | --- | --- |
| **Ejecución** | **Extended f\_10 DE (Nuestro)** | **Extended f\_10 DE (Scipy)** | **Extended f\_10 SADE (Pyade)** |
| **1** | 10.360487348302945 | 6.894336114962439 | 2.7280760263221246 |
| **2** | 6.968209854089093 | 6.652864133265149 | 2.5363290454955814 |
| **3** | 18.488031499176973 | 9.08081617145243 | 2.4767535049427116 |
| **4** | 6.541513836790551 | 8.760645519469872 | 3.5731923957990888 |
| **5** | 11.216183273709767 | 7.921293447914139 | 2.633889371276125 |
| **6** | 2.8643100980509026 | 7.608818055870414 | 2.404112857816869 |
| **7** | 3.420124965615564 | 8.691321134202626 | 3.2629668380292634 |
| **8** | 21.063404159426074 | 5.747895548210275 | 3.1017603926521513 |
| **9** | 8.979560340201186 | 8.178585643418337 | 3.4055404154914237 |
| **10** | 7.135370499315873 | 6.318220684297204 | 3.2805822717840045 |
| **Media** | 9.703719587467893 | 7.585479645306289 | 2.9403203119609342 |
| **Desv. Típica** | 5.960890365365734 | 1.1370103846984934 | 0.4304348990646381 |
| **Mediana** | 8.057465419758529 | 7.765055751892277 | 2.914918209487138 |
| **Min** | 2.8643100980509026 | 5.747895548210275 | 2.404112857816869 |
| **Max** | 21.063404159426074 | 9.08081617145243 | 3.5731923957990888 |

*Función Bohachevsky*

def \_bohachevsky(x):  
 sum\_ = 0.0  
 currentGen = x[0]  
  
 for n in range(1, len(x)):  
 nextGen = x[n]  
 sum\_ += currentGen \* currentGen + 2.0 \* nextGen \* nextGen  
 sum\_ += -0.3 \* np.cos (3.0 \* np.pi \* currentGen) - 0.4 \* np.cos (4.0 \* np.pi \* nextGen) + 0.7  
 currentGen = nextGen  
  
 return sum\_

|  |  |  |  |
| --- | --- | --- | --- |
| **Ejecución** | **Bohachevsky DE**  **(Nuestro)** | **Bohachevsky DE (Scipy)** | **Bohachevsky SADE (Pyade)** |
| **1** | 1.2712461327022804 | 3.6192345761242324 | 0.21436679451702587 |
| **2** | 0.0 | 4.0321614064041835 | 0.2118774478705867 |
| **3** | 20.67016669462263 | 5.2630611659776125 | 0.017962089859151518 |
| **4** | 9.883190814300495 | 0.0 | 2.6383142469779868 |
| **5** | 3.8278866431726635 | 4.669018629858509 | 0.16568100484639597 |
| **6** | 1.0882672003702607 | 1.4627108840100955 | 0.08771768658961085 |
| **7** | 2.6156200819362847 | 0.41292683027581717 | 0.1264396967134566 |
| **8** | 25.50282702958363 | 2.569450522389959 | 0.25359257489060355 |
| **9** | 1.2960152425544065 | 0.0 | 0.16442395060049955 |
| **10** | 0.06427341863029154 | 2.5694505223901047 | 0.13322596806881398 |
| **Media** | 6.621949325787294 | 2.4598014537430513 | 0.40136014609341314 |
| **Desv. Típica** | 9.206300215974503 | 1.9397790275101991 | 0.7888948234514646 |
| **Mediana** | 1.9558176622453456 | 2.569450522390032 | 0.16505247772344778 |
| **Min** | 0.0 | 0.0 | 0.017962089859151518 |
| **Max** | 25.502827029583628 | 5.2630611659776125 | 2.6383142469779868 |

*Función Schaffer*

def \_schaffer(x):  
 sum\_ = 0.0  
 currentGen = x[0]  
 currentGen = currentGen \* currentGen  
  
 for n in range(1, len(x)):  
 nextGen = x[n]  
 nextGen = nextGen \* nextGen  
 aux = currentGen + nextGen  
 currentGen = nextGen  
 aux2 = np.sin(50.0 \* pow(aux, 0.1))  
 sum\_ += pow(aux, 0.25) \* (aux2 \* aux2 + 1.0)  
  
 return sum\_

|  |  |  |  |
| --- | --- | --- | --- |
| **Ejecución** | **Schaffer DE**  **(Nuestro)** | **Schaffer DE**  **(Scipy)** | **Schaffer SADE**  **(Pyade)** |
| **1** | 0.0 | 8.325828324526784 | 2.438692633612565 |
| **2** | 8.225727693739369 | 8.094497616504462 | 2.44726615164237 |
| **3** | 6.303157303028167 | 6.375997645269012 | 3.1354033355816053 |
| **4** | 9.987800262470241 | 8.198238434700265 | 2.322540198136492 |
| **5** | 4.166861933568242 | 7.933075120264752 | 2.5753713685192965 |
| **6** | 6.681139237857854 | 7.994135129836874 | 3.35240927654809 |
| **7** | 18.95357636742272 | 6.641273525191726 | 1.2028608291935952 |
| **8** | 5.127779329146744 | 9.095568071205532 | 2.925548439088487 |
| **9** | 17.292573993395564 | 7.632129435484974 | 2.7073126295018906 |
| **10** | 0.0 | 8.476742131910347 | 2.2761456933396196 |
| **Media** | 7.67386161206289 | 7.876748543489473 | 2.538355055516401 |
| **Desv. Típica** | 6.3668645250286815 | 0.8201218735467214 | 0.5884150959800634 |
| **Mediana** | 6.4921482704430105 | 8.044316373170668 | 2.511318760080833 |
| **Min** | 0.0 | 6.375997645269012 | 1.2028608291935952 |
| **Max** | 18.95357636742272 | 9.095568071205532 | 3.35240927654809 |

*Tests estadísticos de Kruskal y Friedman*

|  |  |  |
| --- | --- | --- |
| **Función** | **Statistic** | **p-value** |
| **Sphere (Kruskal)** | 24.378752886836025 | 5.084181813288585 e-06 |
| **Ackley (Kruskal)** | 9.968498845265595 | 0.00684491373717667 |
| **Rosenbrock (Kruskal)** | 14.18381270903009 | 0.000831810128175261 |
| **Rastrigin (Kruskal)** | 0.5909677419354864 | 0.7441714173911155 |
| **Griewank (Kruskal)** | 11.448434332056776 | 0.0032659089399817397 |
| **Schwefel 2.21 (Kruskal)** | 22.182456140350876 | 1.524547118981126e-05 |
| **Schwefel 2.22 (Kruskal)** | 15.654805131929317 | 0.00039865962873010665 |
| **Schwefel 1.2 (Kruskal)** | 21.786292397660805 | 1.8585177587672292e-05 |
| **Extended f\_10 (Kruskal)** | 17.194838709677427 | 0.00018458151896825312 |
| **Bohachevsky (Kruskal)** | 5.519750612335783 | 0.06329966094484653 |
| **Schaffer (Kruskal)** | 13.52817089452604 | 0.0011545028653648453 |
| **Friedman** | 8.72727272727272 | 0.012732005168250004 |

**Análisis de los resultados**

Observando los resultados, pudimos concluir que nuestro algoritmo es capaz de alcanzar óptimos absolutos o valores cercanos a ellos, al igual que los otros dos algoritmos de biblioteca. Sin embargo, si observamos valores como los fitness o la desviación típica obtenidos a lo largo de todas las ejecuciones, nuestros resultados son bastante más dispares que los obtenido por Scipy o Pyade.

Del mismo modo podemos ver como todos los p-valores que se obtienen en los tests de Kruskal y Friedman son valores en el intervalo [0, 1] y solo el p-valor obtenido en el test de Kruskal sobre los fitness de Rastrigin es mayor a 0.5, por lo que sería posible rechazar la hipótesis nula en la mayoría de ellos

En cuanto a los resultados, sabemos que no son los mejores ya que los resultados no son tan uniformes como los de Scipy o Pyade. Pero, sin duda, estamos bastante orgullosos de que nuestra implementación obtenga valores competentes con respecto a los obtenidos por las implementaciones de biblioteca.