Karolina Tatarczyk

Projekt 5 - Ewolucja różnicowa

Mutacja

In[94]:=
```
mutation[ListOfPoints_, weight_] := Module[{
w = weight,
Points = ListOfPoints,
n, mutatedTable, r, i
},
mutatedTable = {};
For[i = 1, i ≤ Length[Points], i++,
PointsToChoose = Delete[Points, i];
n = 0;
(*srawdzenie czy wszystkie 3 wektory są różne*)
While[n ≠ 3, r = RandomChoice[PointsToChoose, 3];
n = CountDistinct[r]];
AppendTo[mutatedTable, r[[1]] + w * (r[[2]] - r[[3]])];
];
Return[mutatedTable]
]
```

Krzyżowanie

In[12]:=
```
crossing[ListOfPoints_, ListOfMutated_, ProbabilityOfCrossing_] := Module[{
pc = ProbabilityOfCrossing,
Points = ListOfPoints,
Mutated = ListOfMutated, Crossed, Jr, wektor, i},
Crossed = {};
Jr = RandomInteger[{1, 2}];
wektor = {RandomReal[{0, 1}], RandomReal[{0, 1}]};
For[i = 1, i ≤ Length[Points], i++,
AppendTo[Crossed, {If[wektor[[1]] ≤ pc || Jr == 1, Mutated[[i, 1]], Points[[i, 1]]],
      If[wektor[[2]] ≤ pc || Jr == 2, Mutated[[i, 2]], Points[[i, 2]]]}];
];
Return[Crossed]
]
```

Selekcja

In[13]:=
```
selection[basePopulation_,
  posteriorPopulation_, function_, rangex_, rangey_] := Module[{
PreviousPoints = basePopulation,
NewPoints = posteriorPopulation,
f = function,
dx = rangex,
dy = rangey,
afterSelection, newpoint, i
},
afterSelection = {};
For[i = 1, i ≤ Length[PreviousPoints], i++,
newpoint = If[f[NewPoints[[i]]] > f[PreviousPoints[[i]]] &&
      IntervalMemberQ[Interval[dx], NewPoints[[i, 1]]] && IntervalMemberQ[
       Interval[dy], NewPoints[[i, 2]]], NewPoints[[i]], PreviousPoints[[i]]];
AppendTo[afterSelection, newpoint];
];
Return[afterSelection]
]
```

Algorytm Ewolucji różnicowej

In[14]:=
```
evolution[function_, numberofpopulation_,
  probabilityC_, rangex_, rangey_, maxIteration_, weight_] := Module[{
f = function, n = numberofpopulation,
   pC = probabilityC, dx = rangex, dy = rangey, maxIt = maxIteration,
w = weight, firstPopulation, xBest, average, startF,
   newPopulation, r, OT, OTC, ALL, newF, xBestValues, bestOfAllBest, i},
firstPopulation = {};
(*losowanie początkowej populacji*)
For[i = 1, i ≤ n, i++,
AppendTo[firstPopulation, {Random[Real, dx], Random[Real, dy]}]];
xBest = {};
average = {};
(*dodanie najlepszego punktu oraz wartości średniej*)
AppendTo[xBest, SortBy[firstPopulation, f][[Length[firstPopulation]]]];
startF := Table[f[firstPopulation[[d]]], {d, 1, Length[firstPopulation]}];
AppendTo[average, Mean[startF]];
newPopulation = firstPopulation;
For[r = 1, r ≤ maxIt, r++,
OT = mutation[newPopulation, w];
OTC = crossing[newPopulation, OT, pC];
ALL = selection[newPopulation, OTC, f, dx, dy];
AppendTo[xBest, SortBy[ALL, f][[Length[ALL]]]];
newF = Table[f[ALL[[d]]], {d, 1, Length[ALL]}];
AppendTo[average, Mean[newF]];
newPopulation = ALL;
];
xBestValues = Table[f[xBest[[t]]], {t, 1, Length[xBest]}];
bestOfAllBest = SortBy[xBest, f][[Length[xBest]]];
Return[{xBest, xBestValues, average, bestOfAllBest}]]
```

In[152]:=

```
f1[{x_, y_}] := 20 - x - y;
n = 10;
pC = 0.9;
dx = {-3, 3};
dy = {-3, 3};
it = 7;
w = 0.9;
evolution[f1, n, pC, dx, dy, it, w]
```

Out[159]=

```
{{{-1.89184, -0.540875}, {-1.89184, -0.540875},
   {-1.89184, -0.540875}, {-0.908866, -2.50629}, {-0.908866, -2.50629},
   {-1.91811, -2.80833}, {-2.25673, -2.5784}, {-2.25673, -2.5784}},
 {22.4327, 22.4327, 22.4327, 23.4152, 23.4152, 24.7264, 24.8351, 24.8351},
 {19.8328, 19.9942, 20.1073, 21.2066, 21.825, 22.718, 23.3541, 23.3541},
 {-2.25673, -2.5784}}
```

Procedura wyświetlania

In[15]:=
```
Drawing[function_, sol_, rangex_, rangey_] :=
 Module[{fun = function, solution = sol, dx = rangex, dy = rangey},
plot = Plot3D[fun[{x, y}], {x, dx[[1]], dx[[2]]},
    {y, dy[[1]], dy[[2]]}, ColorFunction → "BlueGreenYellow"];
tabBestIteration = Table[{i, solution[[2, i]]}, {i, 1, Length[solution[[2]]]}];
tabAverIteration = Table[{i, solution[[3, i]]}, {i, 1, Length[solution[[3]]]}];
tabPoints = Table[Take[solution[[1]], i], {i, 1, Length[solution[[1]]]}];
density = DensityPlot[fun[{x, y}], {x, dx[[1]], dx[[2]]},
    {y, dy[[1]], dy[[2]]}, ColorFunction → "BlueGreenYellow"];
Linesplot =
   Table[ListLinePlot[tabPoints[[i]], PlotStyle → Pink], {i, 1, Length[solution[[1]]]}];
Pointsplot =
   Table[ListPlot[tabPoints[[i]], PlotStyle → Pink], {i, 1, Length[solution[[1]]]}];
animacja = ListAnimate[Table[Show[density, Linesplot[[i]], Pointsplot[[i]],
     {i, 1, Length[solution[[1]]]}], AnimationRunning → False];

iterBest = ListPlot[tabBestIteration, PlotRange → Full, ImageSize → 360];
iterAver = ListLinePlot[tabAverIteration, PlotRange → Full, ImageSize → 360];
Grid[{{{"EWOLUCJA RÓŻNICOWA", SpanFromLeft},
{"Wzór funkcji", "Wykres funkcji"}, {fun[{x, y}], plot},
{"Przebieg wartości maksymalnej w zależności od iteracji",
     "Przebieg wartości średniej w zależności od iteracji"}, {iterBest, iterAver},
{"Najlepsza znaleziona wartość", "Punkt, w którym znaleziono maksimum"},
    {fun[solution[[4]]], solution[[4]]},
{"Przebieg poszukiwania rozwiązania", SpanFromLeft}, {animacja, SpanFromLeft}},
Frame → All,
   Background → {None, {Pink}, {{2, 1} → LightPink, {2, 2} → LightPink, {4, 1} → LightPink,
     {4, 2} → LightPink, {6, 1} → LightPink, {6, 2} → LightPink, {8, 1} → LightPink}}]]
```

Przykład 1

In[96]:=
```
Clear[n, pC, dx, dy, it, solution, solution1]
AckleyFunction2[{x_, y_}] :=
  100 - (-20 * Exp[-0.2 * Sqrt[(x ^ 2 + y ^ 2) / 2]] - Exp[(Cos[2 * Pi * x] + Cos[2 * Pi * y]) / 2] + 20 + E);
n = 10;
pC = 0.9;
dx = {-3, 3};
dy = {-3, 3};
it = 20;
w = 0.9;
(*funkcja,ilość osobników w populacji, prawdopodobieństwo krzyżowania,
przedział dx, przedział dy, ilość iteracji, liczba F*)
solution = evolution[AckleyFunction2, n, pC, dx, dy, it, 1.8];
Drawing[AckleyFunction2, solution, dx, dy]
solution1 = evolution[AckleyFunction2, n, pC, dx, dy, it, 0.2];
Drawing[AckleyFunction2, solution1, dx, dy]
```

Przykład 2

```
Clear[n, pC, dx, dy, it, solution, solution1]
SphericalFunction[{x_, y_}] := 300 - (x ^ 2 + y ^ 2);
n = 10;
pC = 0.7;
dx = {-10, 10};
dy = {-10, 10};
it = 20;
solution = evolution[SphericalFunction, n, pC, dx, dy, it, 1.8];
Drawing[SphericalFunction, solution, dx, dy]
solution1 = evolution[SphericalFunction, n, pC, dx, dy, it, 0.3];
Drawing[SphericalFunction, solution1, dx, dy]
```

In[264]:=

Przykład 3

In[49]:=
```
Clear[n, pC, dx, dy, it, solution, solution1]
ShubertFunction2[{x_, y_}] :=
  200 - Sum[i * Cos[(i + 1) * x + i], {i, 5}] * Sum[i * Cos[(i + 1) * y + i], {i, 5}];
n = 10;
pC = 0.9;
dx = {-10, 10};
dy = {-10, 10};
it = 20;
solution = evolution[ShubertFunction2, n, pC, dx, dy, it, 1.7];
Drawing[ShubertFunction2, solution, dx, dy]
solution1 = evolution[ShubertFunction2, n, pC, dx, dy, it, 0.2];
Drawing[ShubertFunction2, solution1, dx, dy]
```

Wnioski

Zwiększenie ilości losowanych punktów zwiększa prawdopodobieństwo uzyskiwania dobrych wyników już w pierwszej iteracji.
Analizując powyższe przykłady, po kilku testach, można wywnioskować, iż mniejsza wartość F zapewnia lepsze wyniki i można zauważyć fakt, że wykres dla średnich rośnie szybciej dla mniejszego F. Dzieje się tak być może dlatego, że większa wartość F zwiększa wartości współrzędnych dla wektora dawcy, a co za tym idzie istnieje większe prawdopodobieństwo wyjścia poza zakres dziedziny.

Algorytm różnicowy w porównaniu z algorytmem ewolucyjnym daje nam tylko niegorsze wyniki w każdej kolejnej iteracji. Wykres wartości dla kolejnych iteracji zbiega bezpośrednio do maksimum funkcji w podanym przedziale, podstawały wykres

jest zawsze niemalejący.
W algorytmie ewolucyjnym mutowany był nie każdy osobnik. W algorytmie różnicowym z kolei każdy osobnik przechodzi przez pewnego rodzaju mutację. Dodatkowo mutacja następuje poprzez przemieszanie się 3 losowo wybranych osobników, z udziałem stałej F. Jednakże niska wartość F, w przypadku funkcji o wielu ekstremach lokalnych, może nam nie zapewnić wyjścia z ekstremum lokalnego, gdyż mogą być brane pod uwagę tylko punkty z bardzo bliskiego sąsiedztwa.

Zaletą dla algorytmu ewolucyjnego może być to, że w kroku mutacji oraz krzyżowania za każdym razem został sprawdzany warunek wyjścia poza przedział. Gdy nowy osobnik był poza przedziałem to losowany był nowy. W przypadku ewolucji różnicowej ten warunek jest sprawdzany dopiero na samym końcu i nie mamy tutaj warunku że w takim przypadku losowany jest inny osobnik, więc iteracja utrzymuje tę samą wartość co poprzednia.