Генетический алгоритм, основанный на теории естественного отбора для решения задачи оптимизации

Вступление

Реализация основана на описанном в статье (https://www.mdpi.com/2073-8994/12/11/1758) алгоритме. С помощью него решается задача оптимизации, а именно поиск глобального минимума функции.

Генетический алгоритм является одним из самых ранних и популярных алгоритмов, основанных на популяциях. Он состоит из операций селекции, кроссинговера и мутации, как и прочие эволюционные алгоритмы. В решении задач оптимизации каждый набор аргументов соответствует какой-либо хромосоме, где каждый отдельный аргумент является геном.

Описание алгоритма

Вариация основанного в статье генетического алгоритма основана на концепции теории естественного отбора. Естественный отбор – это биологическая теория, впервые предложенная Чарльзом Дарвином. Теория подразумевает, что гены приспосабливаются и выживают на протяжении поколений с помощью нескольких факторов. Другими словами, организм с высокими способностями способен выжить в текущей среде и порождает новые организмы в новом поколении, в то время как организм с низкими способностями имеет два шанса выжить в текущей среде и избежать вымирания: вступить в брак с организмом, обладающим высокой выживаемостью, что может привести к появлению в новом поколении особи с высокими способностями, или генетическая мутация, которая может привести к тому, что организм станет сильнее и сможет выжить в текущей среде. Если же организм, полученный в результате одного из двух шансов, не удовлетворяет требованиям среды, он может со временем вымереть. Однако происходит взаимовлияние среды и популяции, а поэтому со временем среда имеет свойство тоже изменяться, и меняются критерии отбора (в нашем случае, они становятся жестче).

Для моделирования изменения среды была взята идея об элитизме, при которой лучшая особь популяции переходит в следующее поколение и становится образцом для сравнения показателей других особей. Благодаря этому последующее поколение всегда будет лучше предыдущего или останется таким же по характеристикам.

Таким образом, применение идеи теории естественного отбора в генетическом алгоритме улучшит поиск и разнообразие решений обычного генетического алгоритма.

Общий вид работы алгоритма выглядит так:

1. Указать размер популяции и общее количество итераций алгоритма.
2. Случайным образом сгенерировать начальную популяцию, где каждый ген будет лежать в заданном промежутке, а количество генов будет определяться размерностью функции.
3. Вычислить значение функции для каждой хромосомы в популяции.
4. Вычислить среднее значение полученных результатов.
5. Сравнить значение каждой функции со средним:
    a. Если оно меньше, тогда производим операцию мутации над хромосомой, и она переходит в следующее поколение. (1)
    b. Если оно больше, тогда у хромосомы есть два шанса на улучшение:
        i. Вступить в брак с одной из лучших хромосом. Если полученная хромосома удовлетворяет условию 1, тогда она переходит в последующее поколение.
        ii. Пройти через мутацию. Если мутированная хромосома удовлетворяет условию 1, тогда она проходит в следующее поколение. Иначе – в поколении генерируется новая, случайная хромосома.

В данном алгоритме используется равномерная мутация, при которой мы выбираем случайный ген в хромосоме особи и заменяем его на новый, случайно сгенерированный.

Алгоритм кроссинговера выполняется по формуле:

$$new\ gene = \alpha[i] * chromosome[i] + (1 - \alpha[i]) * rschromosome[i]$$

для каждого гена в хромосоме потомка, где α – хромосома, гены которой случайно генерируются в пределах от $-gamma$ до $gamma$ (являющиеся параметрами алгоритма), $chromosome$ – хромосома, вступающая в брак, а $rschromosome$ – одна из пяти лучших хромосом поколения.

Результаты

По результатам работы алгоритма, он достаточно успешен при решении задач оптимизации функций низких размерностей (до 30-го). При большом количестве итераций и размера поколения алгоритм практически точно приближается к глобальному минимуму, что показано на рисунках 1.1–2.2.

$f_{15}(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$	2	[−5, 5]	0.398

Рисунок 1.1 – Одна из функций для тестирования

```
x:  [3.14888236 2.28128372]
f(x):  0.3982855688584035
```

Рисунок 1.2 – Результат работы алгоритма

$f_{14}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	[−5, 5]	−1.0316

Рисунок 2.1 – Одна из функций для тестирования

```
x:  [ 0.08878379 -0.72046716]
f(x):  -1.0311107730836686
```

Рисунок 2.2 – Результат работы алгоритма

Однако на функциях больших размерностей минимум может не достигаться даже приближенно. Результаты работы алгоритма на таких функциях показаны на рисунках 3.1–4.2.

$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	[−600, 600]	0

Рисунок 3.1 – Одна из функций для тестирования

```
x:  [ 165.19108501   47.          -144.           70.           57.
  -25.           58.          106.           33.         -156.
  -40.          111.           29.          171.         -194.
  -82.          -85.           18.           65.          -20.
  -50.           -3.           -6.           91.           76.
 -161.          -64.          -78.          -68.           77.        ]
f(x):  66.7062736413865
```

Рисунок 3.2 – Результат работы алгоритма

$f_{10}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	128	[−32.768, 32.768]	0

Рисунок 4.1 – Одна из функций для тестирования

```
x:  [ -0.75170574  18.40063501  13.07288308 -19.1650116   11.7952687
 -20.84173027  -4.00344968   8.17226983  -3.82424561   7.06235447
  -9.22508609 -18.07956958  -1.91891068  17.22199044   6.41439078
   3.67714944   5.5489127    6.43454817   2.80472658  -1.14594781
  14.96389818  -5.23354748  -3.75324611   8.88279703   3.35913042
   0.84131477   4.41324485 -10.43447055 -11.0473594    0.71171926
  11.17995696  -3.34270096   0.65446714  -3.7453266   -6.83930797
  -7.86799087  20.08222199  -1.20971858  13.76982089  -2.36769249
   3.24108946  -5.04690339  -5.61428663  -2.18529228  -2.48411045
  -7.71485049 -20.00800417   4.2539491   12.69767899  13.15467247
   5.78055407   2.72795395   4.12406779 -15.12954672   1.23059048
  -0.56097158  -3.18817434   3.29584122  -2.14134246 -11.75331889
   3.72992165  -1.58433435  18.67613941   9.70944639  -2.59688631
 -10.62441318   0.93284173   3.56573876  -7.17603801  -5.93504808
   3.70857088  -0.41140929 -20.51352803   1.16663858  -3.00283656
 -14.2114992    3.97413997 -25.95701     10.37435514   2.23631791
  18.60640084  -9.37240688  10.3965107  -11.75542862  -3.34392913
  -4.23614103   1.04876868  -0.75676307 -14.63048539  -7.8976053
  -4.68571944  -9.80813736  -5.83479373  -0.52430885   5.48574273
   5.74464517 -15.33110407   0.85115915  -5.97445841   4.50001025
  -2.47342489  -8.34629044   9.0210843    5.56002278   4.58353665
   5.99473333  -4.23619308   1.84831433  -6.4256451  -16.5676015
  -6.34738172   2.55175227  -1.04561547   5.71571277 -16.88017823
  -3.1145866    8.38280904  -0.5838027  -29.76030306  13.94528686
   1.7184491   -1.24509464 -21.64955572   3.13893341   4.14021538
  -1.3129894   -9.33405545  -6.01970582]
f(x):  18.724263006183367
```

Рисунок 4.2 – Результат работы алгоритма

Таким образом, алгоритм достаточно хорошо показал себя на множестве протестированных функций, хоть и не на всех приблизился к минимуму. При этом алгоритм является расширяемым, и можно достичь лучших результатов при изменении его параметров, а также поиска и разработки новых функций мутации и кроссинговера.