ВИКИПЕДИЯ

Генетический алгоритм

Материал из Википедии — свободной энциклопедии

Генети́ческий алгори́тм (англ. *genetic algorithm*) — это эвристический алгоритм поиска, используемый для решения задач оптимизации и моделирования путём случайного подбора, комбинирования и вариации искомых параметров с использованием механизмов, аналогичных естественному отбору в природе. Является разновидностью эволюционных вычислений, с помощью которых решаются оптимизационные задачи с использованием методов естественной эволюции, таких как наследование, мутации, отбор и кроссинговер. Отличительной особенностью генетического алгоритма является акцент на использование оператора «скрещивания», который производит операцию рекомбинации решений-кандидатов, роль которой аналогична роли скрещивания в живой природе.

Содержание

История

Первые работы по симуляции эволюции были проведены в 1954 году Нильсом Баричелли на компьютере, установленном в Институте перспективных исследований Принстонского университета.[1][2] Его работа, опубликованная в том же году, привлекла широкое внимание общественности. С 1957 года,[3] австралийский генетик Алекс Фразер опубликовал серию работ по симуляции искусственного отбора среди организмов с множественным контролем измеримых характеристик. Положенное начало позволило компьютерной симуляции эволюционных

процессов и методам, описанным в книгах Фразера и Барнелла(1970)[4] и Кросби (1973)[5], с 1960-х годов стать более распространенным видом деятельности среди биологов. Симуляции Фразера включали все важнейшие элементы современных генетических алгоритмов. Вдобавок к этому, Ганс-Иоахим Бремерманн в 1960-х опубликовал серию работ, которые также принимали подход использования популяции решений, подвергаемой рекомбинации, мутации и отбору, в проблемах оптимизации. Исследования Бремерманна также включали элементы современных генетических алгоритмов.[6] Среди прочих пионеров следует отметить Ричарда Фридберга, Джорджа Фридмана и Майкла Конрада. Множество ранних работ были переизданы Давидом Б. Фогелем (1998).[7]

Хотя Баричелли в своей работе 1963 года симулировал способности машины играть в простую игру,[8] искусственная эволюция стала общепризнанным методом оптимизации после работы Инго Рехенберга и Ханса-Пауля Швефеля в 1960-х и начале 1970-х годов двадцатого века — группа Рехенберга смогла решить сложные инженерные проблемы согласно стратегиям эволюции.[9][10][11][12] Другим подходом была техника эволюционного программирования Лоренса Дж. Фогеля, которая была предложена для создания искусственного интеллекта. Эволюционное программирование первоначально использовавшее конечные автоматы для предсказывания обстоятельств, и использовавшее разнообразие и отбор для оптимизации логики предсказания. Генетические алгоритмы стали особенно популярны благодаря работе Джона Холланда в начале 70-х годов и его книге «Адаптация в естественных и искусственных системах» (1975)[13]. Его исследование основывалось на экспериментах с клеточными автоматами, проводившимися Холландом и на его трудах написанных в университете Мичигана. Холланд ввел формализованный подход для предсказывания качества следующего поколения, известный как Теорема схем. Исследования в области генетических алгоритмов оставались в основном теоретическими до середины 80-х годов, когда была, наконец, проведена Первая международная конференция по генетическим алгоритмам в Питтсбурге, Пенсильвания (США).

С ростом исследовательского интереса существенно выросла и вычислительная мощь настольных компьютеров, это позволило использовать новую вычислительную технику на практике. В конце 80-х, компания General Electric начала продажу первого в мире продукта, работавшего с использованием генетического алгоритма. Им стал набор промышленных вычислительных средств. В 1989, другая компания Axcelis, Inc. выпустила Evolver — первый в мире коммерческий продукт на генетическом алгоритме для настольных компьютеров. Журналист The New York Times в технологической сфере Джон Маркофф писал[14] об Evolver в 1990 году.

Описание алгоритма

Задача формализуется таким образом, чтобы её решение могло быть закодировано в виде вектора («генотипа») генов, где каждый ген может быть битом, числом или неким другим объектом. В классических реализациях генетического алгоритма (ГА) предполагается, что генотип имеет фиксированную длину. Однако существуют вариации ГА, свободные от этого ограничения.

Некоторым, обычно случайным, образом создаётся множество генотипов начальной популяции. Они оцениваются с использованием «функции приспособленности», в результате чего с каждым генотипом ассоциируется определённое значение («приспособленность»), которое определяет насколько хорошо фенотип, им описываемый, решает поставленную задачу.

При выборе «функции приспособленности» (или fitness function в англоязычной литературе) важно следить, чтобы её «рельеф» был «гладким».

Из полученного множества решений («поколения») с учётом значения «приспособленности»

выбираются решения (обычно лучшие особи имеют большую вероятность быть выбранными), к которым применяются «генетические операторы» (в большинстве случаев «скрещивание» — crossover и «мутация» — mutation), результатом чего является получение новых решений. Для них также вычисляется значение приспособленности, и затем производится отбор («селекция») лучших решений в следующее поколение.

Этот набор действий повторяется итеративно, так моделируется «эволюционный процесс», продолжающийся несколько жизненных циклов (*поколений*), пока не будет выполнен критерий остановки алгоритма. Таким критерием может быть:

- нахождение глобального, либо субоптимального решения;
- исчерпание числа поколений, отпущенных на эволюцию;
- исчерпание времени, отпущенного на эволюцию.

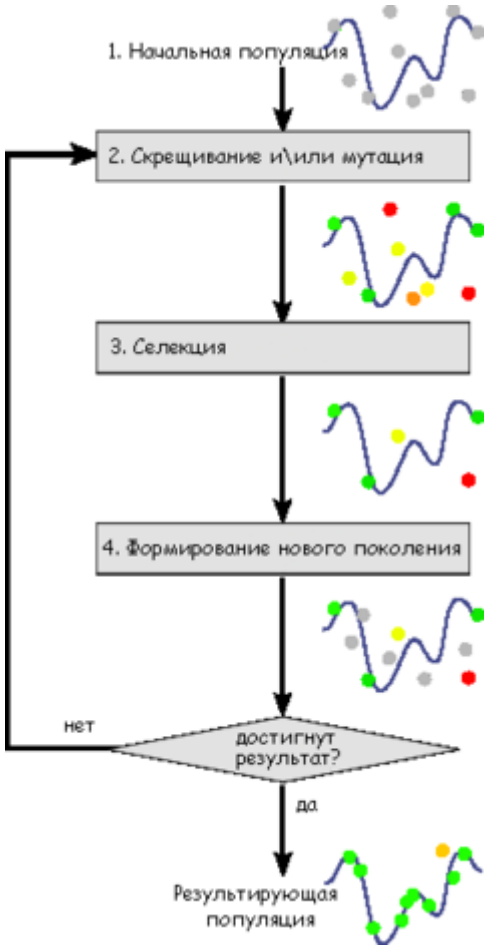

Схема работы генетического алгоритма

Генетические алгоритмы служат, главным образом, для поиска решений в многомерных пространствах поиска.

Таким образом, можно выделить следующие этапы генетического алгоритма:

1. Задать целевую функцию (приспособленности) для особей популяции
2. Создать начальную популяцию

- (Начало цикла)

    1. Размножение (скрещивание)
    2. Мутирование
    3. Вычислить значение целевой функции для всех особей
    4. Формирование нового поколения (селекция)
    5. Если выполняются условия остановки, то (конец цикла), иначе (начало цикла).

Создание начальной популяции

Перед первым шагом нужно случайным образом создать начальную популяцию; даже если она окажется совершенно неконкурентоспособной, вероятно, что генетический алгоритм всё равно достаточно быстро переведёт её в жизнеспособную популяцию. Таким образом, на первом шаге можно особенно не стараться сделать слишком уж приспособленных особей, достаточно, чтобы они соответствовали формату особей популяции, и на них можно было подсчитать функцию приспособленности (Fitness). Итогом первого шага является популяция H, состоящая из N

особей.

Отбор (селекция)

На этапе отбора нужно из всей популяции выбрать определённую её долю, которая останется «в живых» на этом этапе эволюции. Есть разные способы проводить отбор. Вероятность выживания особи h должна зависеть от значения функции приспособленности Fitness(h). Сама доля выживших s обычно является параметром генетического алгоритма, и её просто задают заранее. По итогам отбора из N особей популяции H должны остаться sN особей, которые войдут в итоговую популяцию H'. Остальные особи погибают.

- Турнирная селекция — сначала случайно выбирается установленное количество особей (обычно две), а затем из них выбирается особь с лучшим значением функции приспособленности
- Метод рулетки — вероятность выбора особи тем вероятнее, чем лучше её значение функции приспособленности $p_i = \frac{f_i}{\sum_{i=1}^{N} f_i}$, где p_i — вероятность выбора i особи, f_i — значение функции приспособленности для i особи, N — количество особей в популяции
- Метод ранжирования — вероятность выбора зависит от места в списке особей отсортированном по значению функции приспособленности $p_i = \frac{1}{N}(a - (a-b)\frac{i-1}{N-1})$, где $a \in [1,2]$, $b = 2 - a$, i — порядковый номер особи в списке особей отсортированном по значению функции приспособленности (то есть $\forall\, i\, \forall\, j > i \;\; f_i \leq f_j$ — если мы минимизируем значение функции приспособленности)
- Равномерное ранжирование — вероятность выбора особи определяется выражением: $p_i = \begin{cases} \frac{1}{\mu}, & \text{if } 1 \leq i \leq \mu \\ 0, & \text{if } \mu \leq i \leq N \end{cases}$, где $\mu \leq N$ параметр метода
- Сигма-отсечение — для предотвращения преждевременной сходимости генетического алгоритма используются методы, масштабирующие значение целевой функции. Вероятность выбора особи тем больше, чем оптимальнее значение масштабируемой целевой функции $p_i = \frac{F_i}{\sum_{i=1}^{N} F_i}$, где $F_i = 1 + \frac{f_i - f_{avg}}{2\sigma}$, f_{avg} — среднее значение целевой функции для всей популяции, σ — среднеквадратичное отклонение значения целевой функции[15].

Выбор родителей

Размножение в генетических алгоритмах требует для производства потомка нескольких родителей, обычно двух.

Можно выделить несколько операторов выбора родителей:

1. Панмиксия — оба родителя выбираются случайно, каждая особь

популяции имеет равные шансы быть выбранной
2. Инбридинг — первый родитель выбирается случайно, а вторым выбирается такой, который наиболее похож на первого родителя
3. Аутбридинг — первый родитель выбирается случайно, а вторым выбирается такой, который наименее похож на первого родителя

Инбридинг и аутбридинг бывают в двух формах: фенотипной и генотипной. В случае фенотипной формы похожесть измеряется в зависимости от значения функции приспособленности (чем ближе значения целевой функции, тем особи более похожи), а в случае генотипной формы похожесть измеряется в зависимости от представления генотипа (чем меньше отличий между генотипами особей, тем особи похожее).

Размножение (Скрещивание)

Размножение в разных алгоритмах определяется по-разному — оно, конечно, зависит от представления данных. Главное требование к размножению — чтобы потомок или потомки имели возможность унаследовать черты обоих родителей, «смешав» их каким-либо способом.

Почему особи для размножения обычно выбираются из всей популяции H, а не из выживших на первом шаге элементов H' (хотя последний вариант тоже имеет право на существование)? Дело в том, что главный недостаток многих генетических алгоритмов — отсутствие разнообразия (diversity) в особях. Достаточно быстро выделяется один-единственный генотип, который представляет собой локальный максимум, а затем все элементы популяции проигрывают ему отбор, и вся популяция «забивается» копиями этой особи. Есть разные способы борьбы с таким нежелательным эффектом; один из них — выбор для размножения не самых приспособленных, но вообще всех особей. Однако такой подход вынуждает хранить всех существовавших ранее особей, что увеличивает вычислительную сложность задачи. Поэтому часто применяют методы отбора особей для скрещивания таким образом, чтобы «размножались» не только самые приспособленные, но и другие особи, обладающие плохой приспособленностью. При таком подходе для разнообразия генотипа возрастает роль мутаций.

Мутации

К мутациям относится все то же самое, что и к размножению: есть некоторая доля мутантов m, являющаяся параметром генетического алгоритма, и на шаге мутаций нужно выбрать mN особей, а затем изменить их в соответствии с заранее определёнными операциями мутации.

Критика

Существует несколько поводов для критики насчёт использования генетического алгоритма по сравнению с другими методами оптимизации:

- Повторная оценка функции приспособленности (фитнесс-функции) для сложных проблем, часто является фактором, ограничивающим использование алгоритмов искусственной эволюции. Поиск оптимального решения для сложной задачи высокой размерности зачастую требует очень затратной оценки функции приспособленности. В реальных задачах, таких как задачи структурной оптимизации, единственный запуск функциональной оценки требует от нескольких часов до

нескольких дней для произведения необходимых вычислений. Стандартные методы оптимизации не могут справиться с проблемами такого рода. В таком случае, может быть необходимо пренебречь точной оценкой и использовать аппроксимацию пригодности, которая способна быть вычислена эффективно. Очевидно, что применение аппроксимации пригодности может стать одним из наиболее многообещающих подходов, позволяющих обоснованно решать сложные задачи реальной жизни с помощью генетических алгоритмов.

- Генетические алгоритмы плохо масштабируемы под сложность решаемой проблемы. Это значит, что число элементов, подверженных мутации очень велико, если велик размер области поиска решений. Это делает использование данной вычислительной техники чрезвычайно сложным при решении таких проблем, как, например, проектирование двигателя, дома или самолёта. Для того чтобы сделать так, чтобы такие проблемы поддавались эволюционным алгоритмам, они должны быть разделены на простейшие представления данных проблем. Таким образом, эволюционные вычисления используются, например, при разработке формы лопастей, вместо всего двигателя, формы здания, вместо подробного строительного проекта и формы фюзеляжа, вместо разработки вида всего самолёта. Вторая проблема, связанная со сложностью, кроется в том, как защитить части, которые эволюционировали с высокопригодными решениями от дальнейшей разрушительной мутации, в частности тогда, когда от них требуется хорошая совместимость с другими частями в процессе оценки пригодности. Некоторыми разработчиками было предложено, что подход, предполагающий развитие пригодности эволюционирующих решений, смог бы преодолеть ряд проблем с защитой, но данный вопрос всё ещё остаётся открытым для исследования.
- Решение является более пригодным лишь по сравнению с другими решениями. В результате условие остановки алгоритма неясно для каждой проблемы.
- Во многих задачах генетические алгоритмы имеют тенденцию сходиться к локальному оптимуму или даже к произвольной точке, а не к глобальному оптимуму для данной задачи. Это значит, что они «не знают», каким образом пожертвовать кратковременной высокой пригодностью для достижения долгосрочной пригодности. Вероятность этого зависит от формы ландшафта пригодности: отдельные проблемы могут иметь выраженное направление к глобальному минимуму, в то время как остальные могут указывать направление для фитнесс-функции на локальный оптимум. Эту проблему можно решить использованием иной фитнесс-функции, увеличением вероятности мутаций, или использованием методов отбора, которые поддерживают разнообразие решений в популяции, хотя Теорема об отсутствии бесплатного обеда при поиске и оптимизации[16] доказывает, что не существует общего решения данной проблемы. Общепринятым методом поддержания популяционного разнообразия является установка уровневого ограничения на численность элементов с высоким сродством, которое снизит число представителей сходных решений в последующих поколениях, позволяя другим, менее сходным элементам оставаться в популяции. Данный приём, тем не менее, может не увенчаться успехом в зависимости от ландшафта конкретной проблемы. Другим возможным методом может служить простое замещение части популяции случайно

сгенерированными элементами, в момент, когда элементы популяции становятся слишком сходны между собой. Разнообразие важно для генетических алгоритмов (и генетического программирования) потому, что перекрёст генов в гомогенной популяции не несёт новых решений. В эволюционных стратегиях и эволюционном программировании, разнообразие не является необходимостью, так как большая роль в них отведена мутации.

Имеется много скептиков относительно целесообразности применения генетических алгоритмов. Например, Стивен С. Скиена, профессор кафедры вычислительной техники университета Стоуни —Брук, известный исследователь алгоритмов, лауреат премии института IEEE, пишет[17]:

> *Я лично никогда не сталкивался ни с одной задачей, для решения которой генетические алгоритмы оказались бы самым подходящим средством. Более того, я никогда не встречал никаких результатов вычислений, полученных посредством генетических алгоритмов, которые производили бы на меня положительное впечатление.*

Применение генетических алгоритмов

Генетические алгоритмы применяются для решения следующих задач:

1. Оптимизация функций
2. Оптимизация запросов в базах данных
3. Разнообразные задачи на графах (задача коммивояжера, раскраска, нахождение паросочетаний)
4. Настройка и обучение искусственной нейронной сети
5. Задачи компоновки
6. Составление расписаний
7. Игровые стратегии
8. Теория приближений
9. Искусственная жизнь
10. Биоинформатика (фолдинг белков)
11. Синтез конечных автоматов
12. Настройка ПИД регуляторов

Пример простой реализации на C++

Поиск в одномерном пространстве, без скрещивания.

```
#include <cstdlib>
#include <ctime>
#include <algorithm>
#include <iostream>
#include <numeric>

int main()
{
    srand((unsigned int)time(NULL));
    const size_t N = 1000;
```

```
    int a[N] = { 0 };
    for ( ; ; )
    {
        //мутация в случайную сторону каждого элемента:
        for (size_t i = 0; i < N; ++i)
            a[i] += ((rand() % 2 == 1) ? 1 : -1);

        //теперь выбираем лучших, отсортировав по возрастанию
        std::sort(a, a + N);
        //и тогда лучшие окажутся во второй половине массива.
        //скопируем лучших в первую половину, куда они оставили потомство, а первые умерли:
        std::copy(a + N / 2, a + N, a);
        //теперь посмотрим на среднее состояние популяции. Как видим, оно всё лучше и лучше.
        std::cout << std::accumulate(a, a + N, 0) / N << std::endl;
    }
}
```

Пример простой реализации на Delphi

Поиск в одномерном пространстве с вероятностью выживания, без скрещивания. *(проверено на Delphi XE)*

```
program Program1;

{$APPTYPE CONSOLE}
{$R *.res}

uses
  System.Generics.Defaults,
  System.Generics.Collections,
  System.SysUtils;

const
  N = 1000;
  Nh = N div 2;
  MaxPopulation = High(Integer);
var
  A: array [1..N] of Integer;
  I, R, C, Points, BirthRate: Integer;
  Iptr: ^Integer;
begin
  Randomize;
  // Частичная популяция
  for I := 1 to N do
    A[I] := Random(2);
  repeat
    // Мутация
    for I := 1 to N do
      A[I] := A[I] + (-Random(2) or 1);
    // Отбор, лучшие в конце
    TArray.Sort<Integer>(A, TComparer<Integer>.Default);
    // Предустановка
    Iptr := Addr(A[Nh + 1]);
    Points := 0;
    BirthRate := 0;
    // Результаты скрещивания
    for I := 1 to Nh do
    begin
      Inc(Points, Iptr^);
      // Случайный успех скрещивания
      R := Random(2);
      Inc(BirthRate, R);
      A[I] := Iptr^ * R;
      Iptr^ := 0;
      Inc(Iptr,1);
    end;
    // Промежуточный итог
    Inc(C);
```

```
    until (Points / N >= 1) or (C >= MaxPopulation);
    Writeln(Format('Population %d (rate:%f) score:%f', [C, BirthRate / Nh, Points / N]));
  end.
```

В культуре

- В фильме 1995 года «Виртуозность» мозг главного злодея выращен генетическим алгоритмом с использованием воспоминаний и поведенческих черт преступников.

Примечания

1. *Barricelli, Nils Aall.* Esempi numerici di processi di evoluzione (неопр.) // Methodos. — 1954. — С. 45—68.
2. *Barricelli, Nils Aall.* Symbiogenetic evolution processes realized by artificial methods (англ.) // Methodos : journal. — 1957. — P. 143—182.
3. *Fraser, Alex.* Simulation of genetic systems by automatic digital computers. I. Introduction (англ.) // Aust. J. Biol. Sci. : journal. — 1957. — Vol. 10. — P. 484—491.
4. *Fraser, Alex; Donald Burnell.* Computer Models in Genetics (неопр.). — New York: McGraw-Hill Education, 1970. — ISBN 0-07-021904-4.
5. *Crosby, Jack L.* Computer Simulation in Genetics (неопр.). — London: John Wiley & Sons, 1973. — ISBN 0-471-18880-8.
6. 02.27.96 — UC Berkeley’s Hans Bremermann, professor emeritus and pioneer in mathematical biology, has died at 69 (http://berkeley.edu/news/media/releases/96legacy/releases.96/14319.html)
7. *Fogel, David B. (editor).* Evolutionary Computation: The Fossil Record (англ.). — New York: Institute of Electrical and Electronics Engineers, 1998. — ISBN 0-7803-3481-7.
8. *Barricelli, Nils Aall.* Numerical testing of evolution theories. Part II. Preliminary tests of performance, symbiogenesis and terrestrial life (англ.) // Acta Biotheoretica : journal. — 1963. — No. 16. — P. 99—126.
9. *Rechenberg, Ingo.* Evolutionsstrategie (неопр.). — Stuttgart: Holzmann-Froboog, 1973. — ISBN 3-7728-0373-3.
10. *Schwefel, Hans-Paul.* Numerische Optimierung von Computer-Modellen (PhD thesis) (нем.). — 1974.
11. *Schwefel, Hans-Paul.* Numerische Optimierung von Computor-Modellen mittels der Evolutionsstrategie : mit einer vergleichenden Einführung in die Hill-Climbing- und Zufallsstrategie (нем.). — Basel; Stuttgart: Birkhäuser, 1977. — ISBN 3-7643-0876-1.
12. *Schwefel, Hans-Paul.* Numerical optimization of computer models (Translation of 1977 Numerische Optimierung von Computor-Modellen mittels der Evolutionsstrategie (англ.). — Chichester ; New York: Wiley, 1981. — ISBN 0-471-09988-0.
13. J. H. Holland. Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor, 1975.

.4. *Markoff, John*. What's the Best Answer? It's Survival of the Fittest (https://www.nytimes.com/1990/08/29/business/business-technology-what-s-the-best-answer-it-s-survival-of-the-fittest.html), New York Times (29 августа 1990). Дата обращения 9 августа 2009.
.5. *Melanie Mitchell.* An Introduction to Genetic Algorithms (https://books.google.com.sg/books?id=0eznlz0TF-IC&pg=PA167#v=onepage&q&f=false). — MIT Press, 1998. — С. 167. — 226 с. — ISBN 9780262631853.
.6. Wolpert, D.H., Macready, W.G., 1995. No Free Lunch Theorems for Optimisation. Santa Fe Institute, SFI-TR-05-010, Santa Fe.
.7. Steven S. Skiena. The Algorithm Design Manual. Second Edition. Springer, 2008.

Книги

- *Саймон Д.* Алгоритмы эволюционной оптимизации. — М: ДМК Пресс, 2020. — 940 с. — ISBN 978-5-97060-812-8.
- *Емельянов В. В., Курейчик В. В., Курейчик В. М.* Теория и практика эволюционного моделирования. — М: Физматлит, 2003. — 432 с. — ISBN 5-9221-0337-7.
- *Курейчик В. М., Лебедев Б. К., Лебедев О. К.* Поисковая адаптация: теория и практика. — М: Физматлит, 2006. — 272 с. — ISBN 5-9221-0749-6.
- *Гладков Л. А., Курейчик В. В., Курейчик В. М.* Генетические алгоритмы: Учебное пособие. — 2-е изд. — М: Физматлит, 2006. — 320 с. — ISBN 5-9221-0510-8.
- *Гладков Л. А., Курейчик В. В, Курейчик В. М. и др.* Биоинспирированные методы в оптимизации: монография. — М: Физматлит, 2009. — 384 с. — ISBN 978-5-9221-1101-0.
- *Рутковская Д., Пилиньский М., Рутковский Л.* Нейронные сети, генетические алгоритмы и нечеткие системы = Sieci neuronowe, algorytmy genetyczne i systemy rozmyte. — 2-е изд. — М: Горячая линия-Телеком, 2008. — 452 с. — ISBN 5-93517-103-1.
- *Скобцов Ю. А.* Основы эволюционных вычислений. — Донецк: ДонНТУ, 2008. — 326 с. — ISBN 978-966-377-056-6.

Ссылки

- Эволюционные вычисления (http://www.getinfo.ru/article31_1.html)
- Использование генетических алгоритмов в проблеме автоматического написания программ (https://web.archive.org/web/20070928141507/http://andyceo.ruware.com/study/ispolzovanie-geneticheskih-algoritmov-v-probleme-avtomaticheskogo-napisaniya-programm)
- Реализация генетических алгоритмов в среде MATLAB v6.12 (https://web.archive.org/web/20070904012934/http://andyceo.ruware.com/study/realizatsiya-geneticheskih-algoritmov-v-srede-matlab-v612)
- Сергей Николенко. Генетические алгоритмы (http://logic.pdmi.ras.ru/~sergey/teaching/ml/04-genetic.pdf) (слайды) — лекция № 4 из курса «Самообучающиеся системы» (https://web.archive.org/web/20071016161230/http://www.csin.ru/courses/samoobuchayushchiesya-sistemy-i-nechetkaya-logika)

- geneticprogramming.us (https://web.archive.org/web/20100404012527/http://www.geneticprogramming.us/)
- Генерирование автоматов состояний с помощью ГА (http://is.ifmo.ru/genalg/)
- Субботін С. О., Олійник А. О., Олійник О. О. Неітеративні, еволюційні та мультиагентні методи синтезу нечіткологічних і нейромережних моделей: Монографія / Під заг. ред. С. О. Субботіна. — Запоріжжя: ЗНТУ, 2009. — 375 с. (http://www.csit.narod.ru/subject/mag_SShI/mono.pdf) (укр.)
- *Poli, R., Langdon, W. B., McPhee, N. F.* A Field Guide to Genetic Programming (неопр.). — Lulu.com, freely available from the internet, 2008. — ISBN 978-1-4092-0073-4.
- Подборка статей по использованию генетических алгоритмов в задачах многокритериальной оптимизации (http://delta.cs.cinvestav.mx/~ccoello/EMOO/) (англ.)
- Lakhmi C. Jain; N.M. Martin Fusion of Neural Networks, Fuzzy Systems and Genetic Algorithms: Industrial Applications. — CRC Press, CRC Press LLC, 1998 (http://science-library.at.ua/load/sistemy_iskusstvennogo_intellekta/nechetkie_sistemy_i_nechetkaja_logika/fusion_of_neural_networks_fuzzy_systems_and_genetic_algorithms_industrial_applications/4-1-0-5)
- Лекция по генетическим алгоритмам (http://ai.lector.ru/?go=lection06)
