

Поиск экстремума функций одной переменной с помощью генетических алгоритмов

Афанасьева Елена
Департамент компьютерных наук
Институт Транспорта и Связи
Ломоносова 1-1, Рига, Латвия
20 апреля 2001 г.

Содержание:

1. Теоретическое обоснование	1
1.1. <i>Естественный отбор</i>	1
1.2. <i>Генетические алгоритмы</i>	2
1.3. <i>Генетические операторы</i>	3
1.4. <i>Цикл выполнения операторов</i>	3
1.5. <i>Особенности генетических алгоритмов</i>	4
2. Описание разработанного ПО	5
2.1. <i>Руководство по эксплуатации</i>	5
2.2. <i>Пример решения конкретной задачи</i>	8
3. Заключение	11
4. Используемая литература	11

1. Теоретическое обоснование

1.1. *Естественный отбор*

Согласно эволюционной теории каждый биологический вид целенаправленно развивается и изменяется для того, чтобы наилучшим образом приспособиться к окружающей среде. Эволюция в этом смысле представляет процесс оптимизации всех живых организмов. Природа решает эту задачу оптимизации путем естественного отбора. Его суть состоит в том, что более приспособленные особи имеют больше возможностей для выживания и размножения и, следовательно, приносят больше потомства, чем плохо приспособленные особи. При этом благодаря передаче генетической информации (*генетическому наследованию*) потомки наследуют от родителей основные их качества. Таким образом, потомки сильных особей также будут относительно хорошо приспособленными, а их доля в общей массе особей будет возрастать. После смены нескольких десятков или сотен поколений средняя приспособленность особей данного вида заметно возрастает.

Дадим краткую справку о том, как устроены механизмы генетического наследования. В каждой клетке любого животного содержится вся генетическая информация данной особи. Эта информация записана в виде набора молекул ДНК, каждая из которых представляет собой цепочку, состоящую из молекул *нуклеотидов* четырех типов, обозначаемых А, Т, С и Г. Собственно информацию несет порядок следования нуклеотидов в ДНК. Таким образом, генетический код особи - это длинная строка, где используются всего 4 символа. В животной клетке каждая молекула ДНК окружена оболочкой, такое образование называется *хромосомой*.

Каждое врожденное качество особи (цвет глаз, наследственные болезни, тип волос и т. д.) кодируется определенной частью хромосомы, которая называется *геном* этого свойства. Например, ген цвета глаз содержит информацию, кодирующую определенный цвет глаз. Различные значения гена называются его *аллелями*.

При размножении особей происходит слияние двух родительских половых клеток, и их ДНК взаимодействуют, образуя ДНК потомка. Основной способ взаимодействия - *кроссовер* (crossover, *скрещивание*). При кроссовере ДНК предков делятся на две части, а затем обмениваются своими половинками.

При наследовании возможны мутации, в результате которых могут измениться некоторые гены в половых клетках одного из родителей. Измененные гены передаются потомку и придают ему новые свойства. Если эти новые свойства полезны, они, скорее всего, сохранятся в данном виде. При этом произойдет скачкообразное повышение приспособленности вида.

1.2. Генетические алгоритмы

Пусть дана некоторая сложная целевая функция, зависящая от нескольких переменных, и требуется решить задачу оптимизации, т. е. найти такие значения переменных, при которых значение функции максимально или минимально.

Эту задачу можно решить, применяя известные биологические эволюционные подходы к оптимизации. Будем рассматривать каждый вариант (набор значений переменных) как особь, а значение целевой функции для этого варианта - как приспособленность данной особи. Тогда в процессе эволюции приспособленность особей будет возрастать, а значит, будут появляться все более и более оптимальные варианты. Остановив эволюцию в некоторый момент и выбрав лучший вариант, можно получить достаточно хорошее решение задачи.

Генетический алгоритм (ГА) - это последовательность управляющих действий и операций, моделирующая эволюционные процессы на основе аналогов механизмов генетического наследования и естественного отбора.

При этом сохраняется биологическая терминология в упрощенном виде.

Хромосома - вектор (последовательность) из нулей и единиц, каждая позиция (бит) которого называется *геном*.

Особь (индивидуум) = *генетический код* - набор хромосом = вариант решения задачи.

Кроссовер - операция, при которой две хромосомы обмениваются своими частями.

Мутация - случайное изменение одной или нескольких позиций в хромосоме.

Генетические алгоритмы представляют собой скорее подход, чем единые алгоритмы. Они требуют содержательного наполнения для решения каждой конкретной задачи.

Математически, изложенное можно представить следующим образом. Пусть имеется некоторая целевая функция от многих переменных, у которой необходимо найти глобальный максимум или минимум:

$$f(x_1, x_2, x_3, \dots, x_n).$$

Представим независимые переменные в виде хромосом. Для этого выполним кодирование независимых переменных либо в двоичном формате, либо в формате с плавающей запятой.

В случае двоичного кодирования используется n бит для каждого параметра, причем n может быть различным.

Хромосомы в формате с плавающей запятой задаются путем последовательного размещения закодированных параметров друг за другом.

Наиболее хорошие результаты дает вариант представления хромосом в двоичном формате (особенно при использовании кодов Грея). Однако в этом случае необходимо постоянно осуществлять кодирование/декодирование параметров (генов).

По скорости определения оптимума целевой функции генетические алгоритмы на несколько порядков превосходят случайный поиск. Причина этому заключается в том, что большинство систем имеют довольно независимые подсистемы. Вследствие чего при обмене генетическим материалом от каждого из родителей берутся гены, соответствующие наиболее удачному варианту определенной подсистемы (неудачные варианты постепенно погибают). Генетический алгоритм позволяет накапливать удачные решения для таких систем в целом.

Генетические алгоритмы менее применимы для систем, которые сложно разбить на подсистемы. Кроме того, они могут давать сбои из-за неудачного порядка расположения генов (например, если рядом расположены параметры, относящиеся к различным подсистемам), при котором преимущества обмена генетическим материалом сводятся к нулю. Это замечание несколько сглаживается в системах с диплоидным (двойным) генетическим набором.

Данные, которые закодированы в генотипе, могут представлять собой команды какой-либо виртуальной машины. В таком случае можно говорить об эволюционном или генетическом программировании. В простейшем случае можно ничего не менять в генетическом алгоритме. Однако в таком случае длина получаемой последовательности действий (программы) получается не отличающейся от той (или тех), которая является «затравкой» на этапе инициализации. Современные алгоритмы генетического программирования функционируют в системах с переменной длиной генотипа.

1.3. Генетические операторы

Существуют три генетических оператора: кроссовер, мутация и инверсия, порядок применения которых не важен.

Из трех генетических операторов кроссовер является наиболее важным. Он генерирует новую хромосому потомка, объединяя генетический материал двух родителей. Существует несколько вариантов кроссовера. Наиболее простым является одноточечный, в котором берутся две хромосомы и «перерезаются» в случайно выбранной точке. Хромосома потомка получается из начала одной и конца другой родительских хромосом:

```
001100101110010|11000    →    00110010111001011100 .
110101101101000|11100
```

Мутация представляет собой случайное изменение хромосомы (обычно простым изменением состояния одного из битов на противоположное). Данный оператор позволяет, во-первых, более быстро находить локальные экстремумы и, во-вторых, перейти на другой локальный экстремум:

```
00110010111001011000    →    00110010111001111000
```

Инверсия изменяет порядок бит в хромосоме путем циклической перестановки (случайное количество раз). Многие модификации ГА обходятся без данного генетического оператора:

```
00110010111001011000    →    11000001100101110010
```

1.4. Цикл выполнения операторов

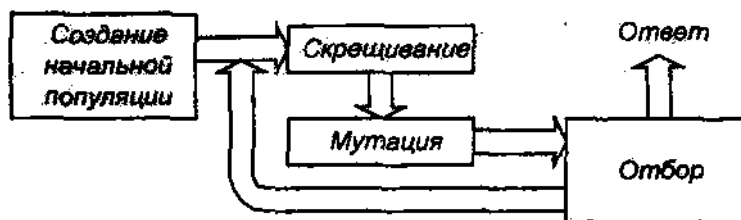


Рис. 1. Вариант структуры генетического алгоритма

На рис. 1 показан один из вариантов структуры генетического алгоритма. Вначале генерируется случайная популяция - несколько особей со случайным набором хромосом (числовых векторов). Генетический алгоритм имитирует эволюцию этой популяции как циклический процесс скрещивания особей, мутации и смены поколений (отбора).

В течение жизненного цикла популяции, т. е. в результате нескольких случайных скрещиваний (посредством кроссовера) и мутаций, к ней добавляется какое-то количество новых вариантов. Далее происходит отбор, в результате которого из старой популяции формируется новая, после чего старая популяция погибает. После отбора к новой популяции опять применяются операции кроссовера и мутации, затем опять происходит отбор, и так далее.

Отбор в генетическом алгоритме тесно связан с принципами естественного отбора следующим образом: приспособленность особи соответствует значению целевой функции на заданном варианте;

выживание наиболее приспособленных особей соответствует тому, что популяция следующего поколения вариантов формируется с учетом целевой функции. Чем приспособленнее особь, тем больше вероятность ее участия в кроссовере, т. е. в размножении.

Таким образом, модель отбора определяет, как следует строить популяцию следующего поколения. Как правило, вероятность участия особи в скрещивании берется пропорциональной ее приспособленности. Часто используется так называемая *стратегия элитизма*, при которой несколько лучших особей переходят в следующее поколение без изменений, не участвуя в кроссовере и отборе. В любом случае каждое следующее поколение будет в среднем лучше предыдущего. Когда приспособленность особей перестает заметно увеличиваться, процесс останавливают и в качестве решения задачи оптимизации берут наилучший из найденных вариантов.

1.5. Особенности генетических алгоритмов

Генетические алгоритмы - не единственный способ решения задач оптимизации. Кроме него существуют два основных подхода для решения таких задач - переборный и локально-градиентный, каждый из которых имеет свои достоинства и недостатки.

Сравним стандартные подходы с генетическими алгоритмами.

Переборный метод наиболее прост в программировании. Для поиска оптимального решения (максимума целевой функции) требуется последовательно вычислить значения целевой функции во всех возможных точках, запоминая максимальное из них. Недостатком метода является большая вычислительная сложность.

Однако, если перебор всех вариантов за разумное время возможен, то найденное решение является оптимальным.

Второй подход основан на методе градиентного спуска. Вначале выбираются некоторые случайные значения параметров, а затем эти значения постепенно изменяют, добиваясь наибольшей скорости роста целевой функции. При достижении локального максимума такой метод останавливается, поэтому для поиска глобального оптимума требуются дополнительные меры.

Градиентные методы работают быстро, но не гарантируют оптимальности найденного решения. Они идеальны для применения в так называемых *унимодальных* задачах, где целевая функция имеет единственный локальный максимум (он же - глобальный).

Практические задачи, как правило, *мультимодальны* и многомерны, т. е. содержат много параметров. Для них не существует универсальных методов, позволяющих достаточно быстро найти абсолютно точные решения. Комбинируя переборный и градиентный методы, можно получить приближенные решения, точность которых будет возрастать с увеличением времени расчета.

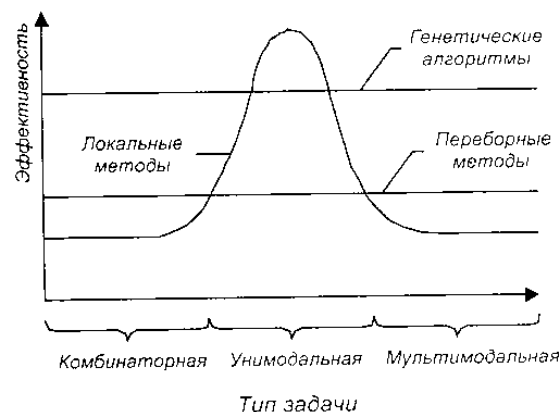


Рис. 2. Эффективность генетических алгоритмов

Генетический алгоритм представляет собой именно такой комбинированный метод. Механизмы скрещивания и мутации в каком-то смысле реализуют переборную часть метода, а отбор лучших решений – градиентный спуск. На рис. 2 показано, что такое сочетание обеспечивает устойчиво хорошую эффективность генетического поиска для любых типов оптимизационных задач.

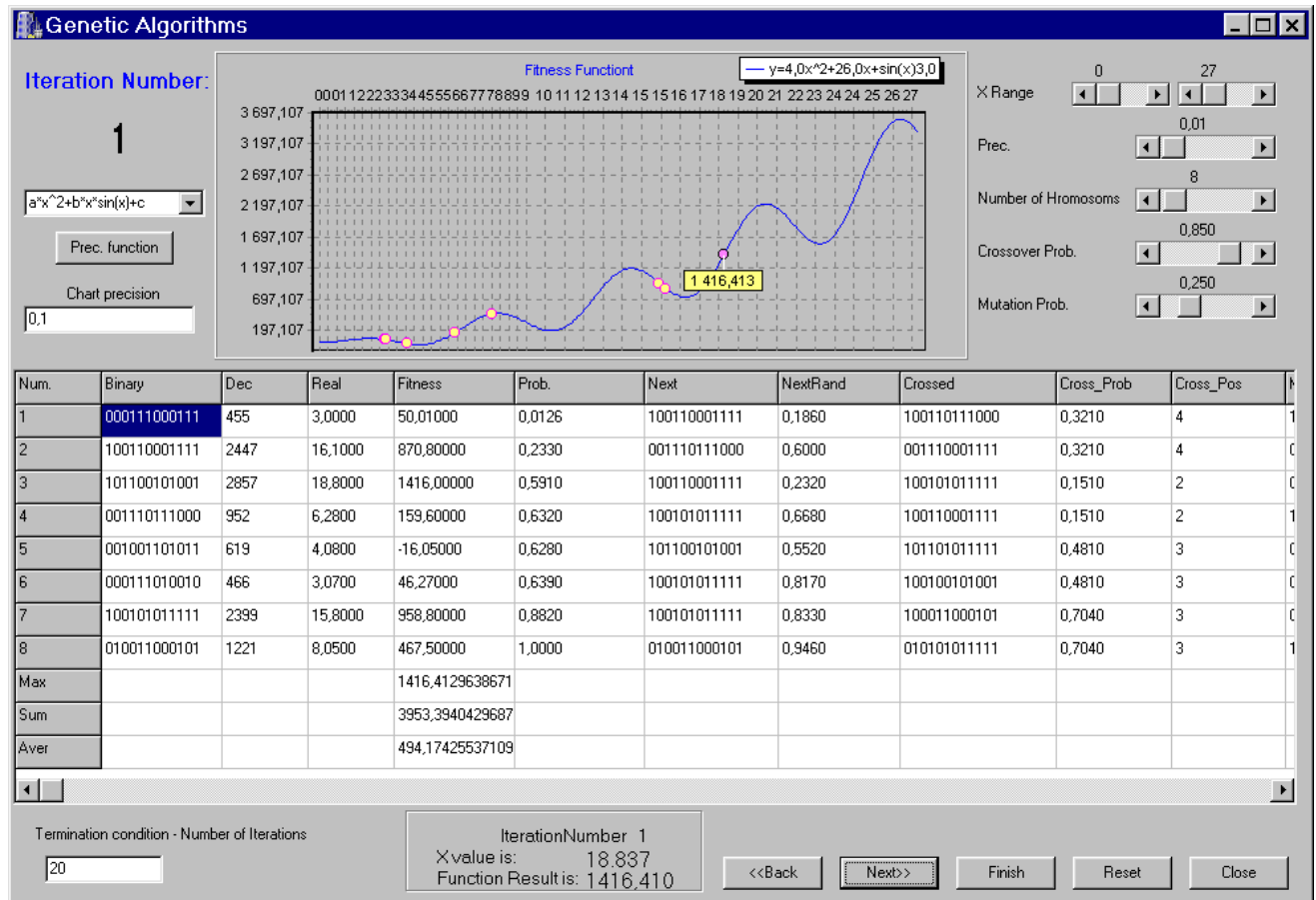
Таким образом, если на некотором множестве задана сложная функция от нескольких переменных, то генетический алгоритм за разумное время находит значение функции достаточно близкое к оптимальному. Задавая время расчета, можно получить одно из лучших решений, которые реально получить за это время.

2. Описание разработанного ПО

2.1. Руководство по эксплуатации

Для демонстрации применения изложенного подхода решено было разработать специальное программное обеспечение. Для реализации была выбрана конкретная задача «Поиск оптимума функции».

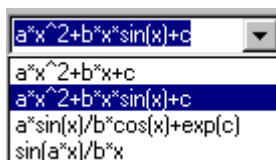
При запуске программы пользователю представляется следующий интерфейс:



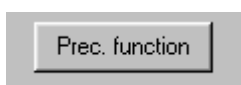
Задание исходных параметров

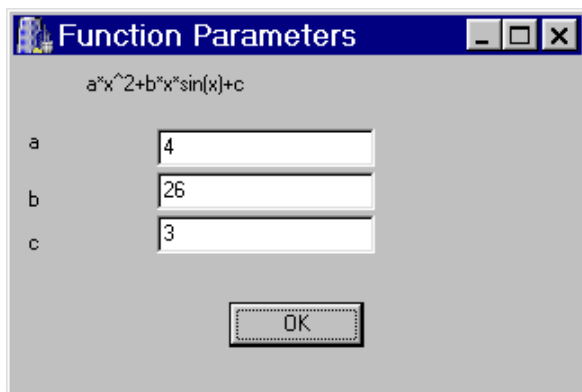
Пользователь должен:

Выбрать функцию для анализа из предложенных в списке.



Далее конкретизировать параметры.





Function Parameters

$a \cdot x^2 + b \cdot x \cdot \sin(x) + c$

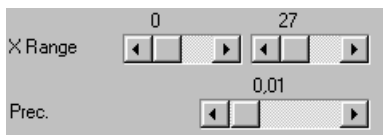
a: 4

b: 26

c: 3

OK

Определить диапазон рассмотрения и размер интервалов.



X Range: 0 27

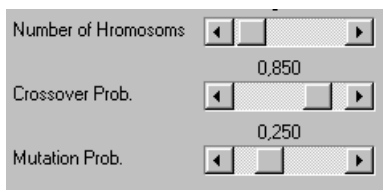
Prec.: 0,01

Определить параметры для работы непосредственно генетического алгоритма.

Задать число хромосом

Вероятность скрещивания

Вероятность мутации



Number of Chromosomes: 1

Crossover Prob.: 0,850

Mutation Prob.: 0,250

Определить точность интервалов для изображения графика(этот параметр носит чисто визуальный характер и нигде в алгоритме не используется).

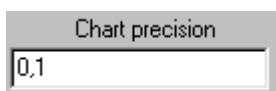
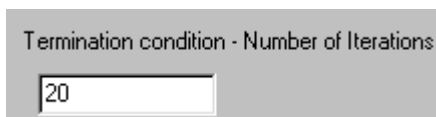


Chart precision

0,1

Определить условие останова — как количество смен поколений, т.е. определение лучшего за ограниченное число шагов.



Termination condition - Number of Iterations

20

Запуск алгоритма

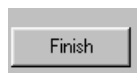
После установки исходных данных можно осуществить запуск алгоритма программы двумя способами:

По шагам



<<Back Next>>

Обозначенное число итераций



Сбросить



Анализ полученных результатов

Результаты текущего шага алгоритма для текущей популяции представляются в таблице:

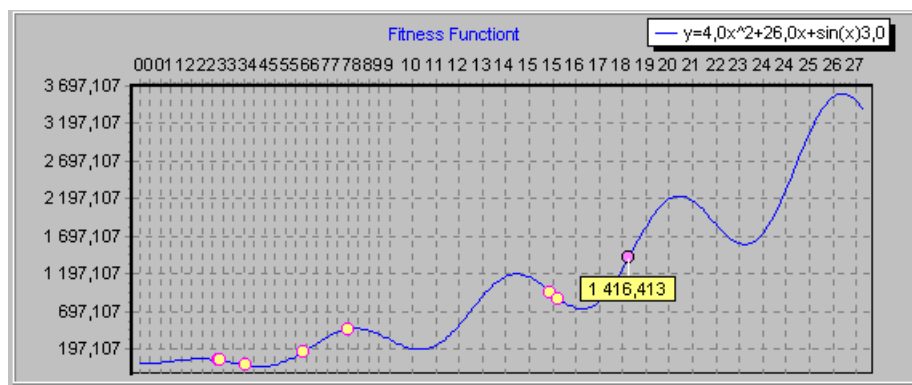
Num.	Binary	Dec	Real	Fitness	Prob.	Next	NextRand	Crossed	Cross_Prob	Cross_Pos
1	000111000111	455	3,0000	50,01000	0,0126	100110001111	0,1860	100110111000	0,3210	4
2	100110001111	2447	16,1000	870,80000	0,2330	001110111000	0,6000	001110001111	0,3210	4
3	101100101001	2857	18,8000	1416,00000	0,5910	100110001111	0,2320	100101011111	0,1510	2
4	001110111000	952	6,2800	159,60000	0,6320	100101011111	0,6680	100110001111	0,1510	2
5	001001101011	619	4,0800	-16,05000	0,6280	101100101001	0,5520	101101011111	0,4810	3
6	000111010010	466	3,0700	46,27000	0,6390	100101011111	0,8170	100100101001	0,4810	3
7	100101011111	2399	15,8000	958,80000	0,8820	100101011111	0,8330	100011000101	0,7040	3
8	010011000101	1221	8,0500	467,50000	1,0000	010011000101	0,9460	010101011111	0,7040	3
Max				1416,4129638671						
Sum				3953,3940429687						
Aver				494,17425537109						

Здесь начальные строки соответствуют каждой особи (хромосоме), представленной бинарным стрингом. Последние 3 строки подводят статистику по текущему поколению.

Столбцы содержат:

- бинарный код хромосомы;
- соответствующее десятичное представление;
- соответствующее значение аргумента исследуемой функции;
- значение Fitness function;
- правая граница интервала для репродукции при помощи «колеса»;
- значение стринга после репродукции до скрещивания;
- сгенерированное случайное число для попадания в интервал для репродукции;
- значение стринга после репродукции и после скрещивания;
- вероятность скрещивания;
- позиция скрещивания;
- значение стринга после мутации (если она произошла).

Также результаты текущего шага алгоритма представлены на графике:



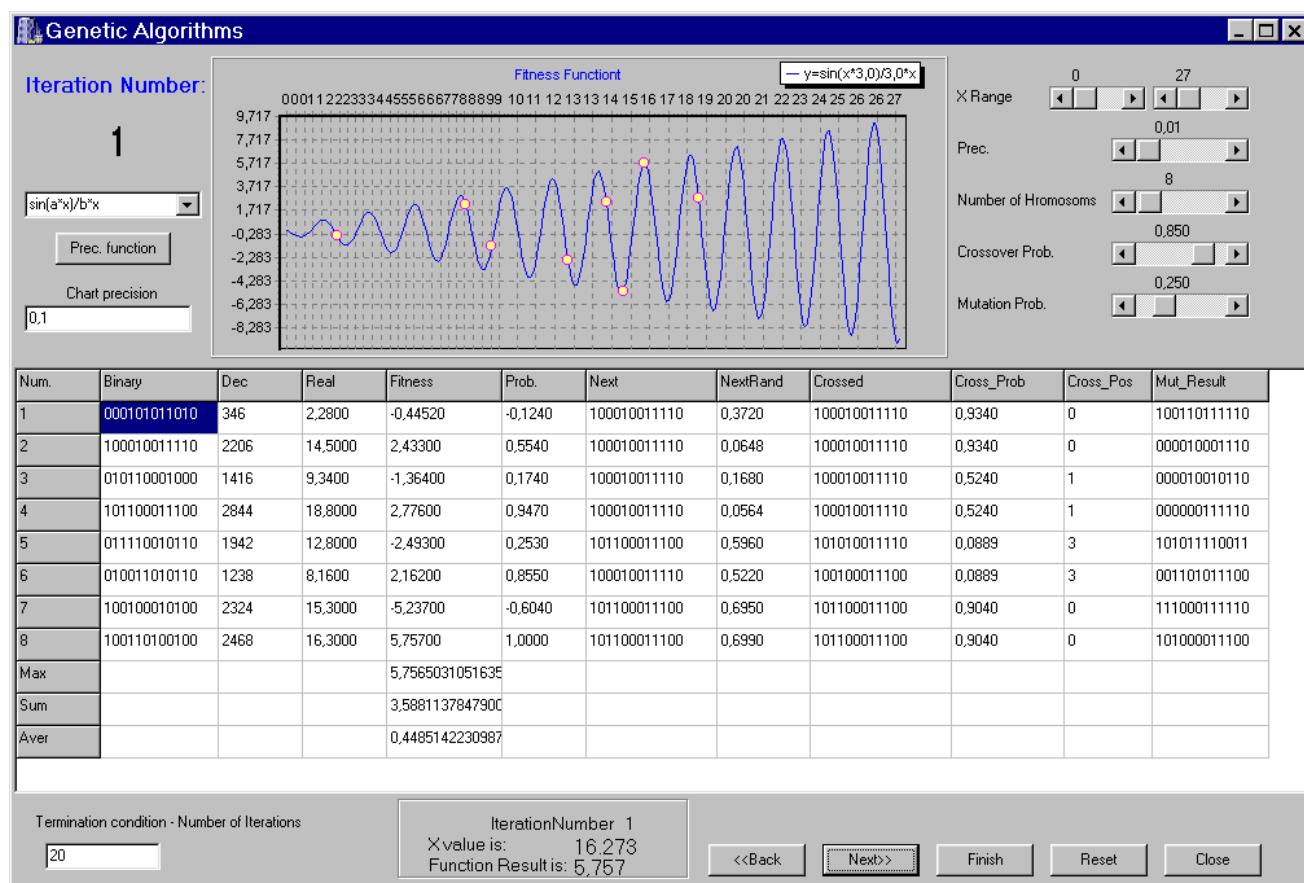
Здесь точками выделены значения текущей популяции.

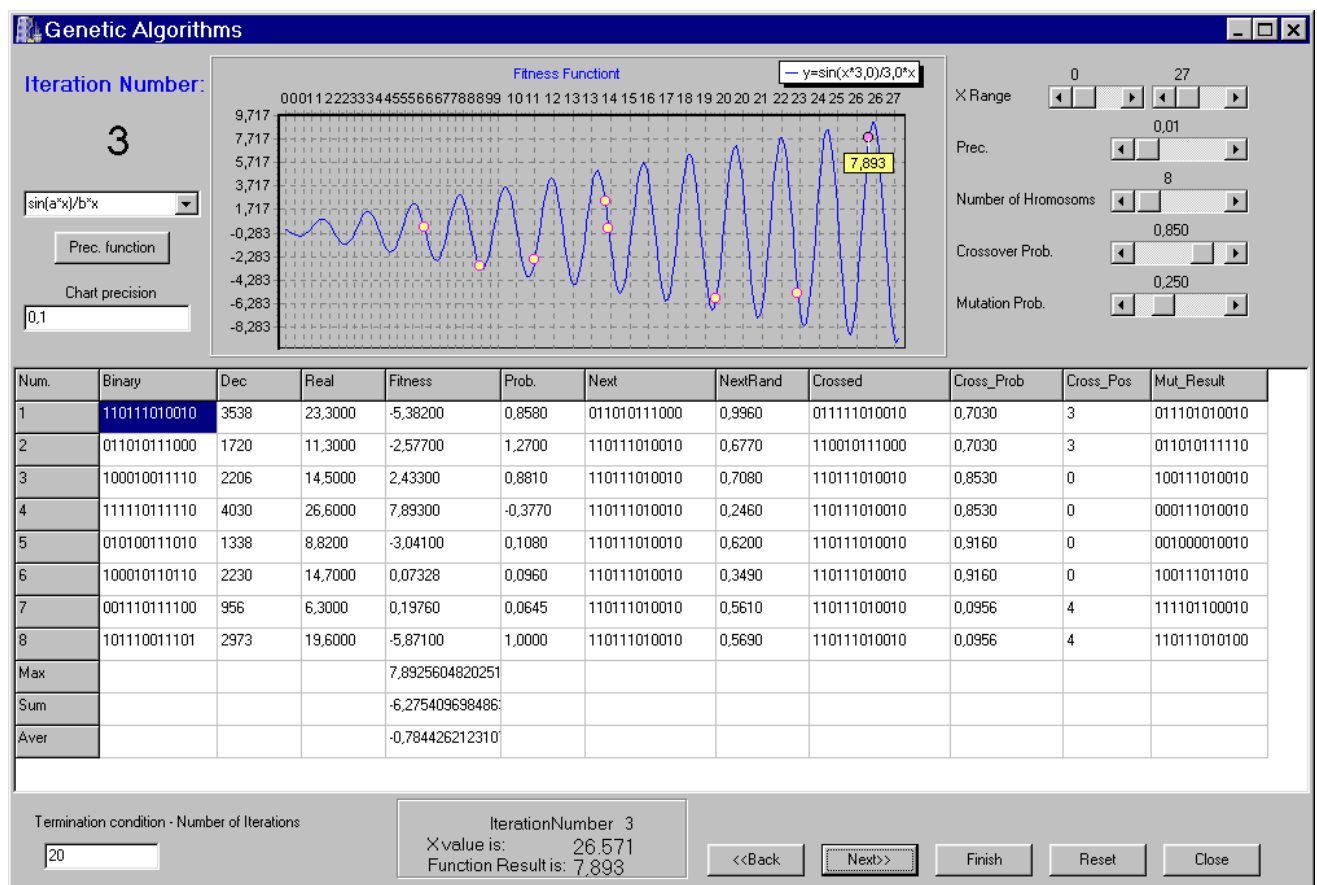
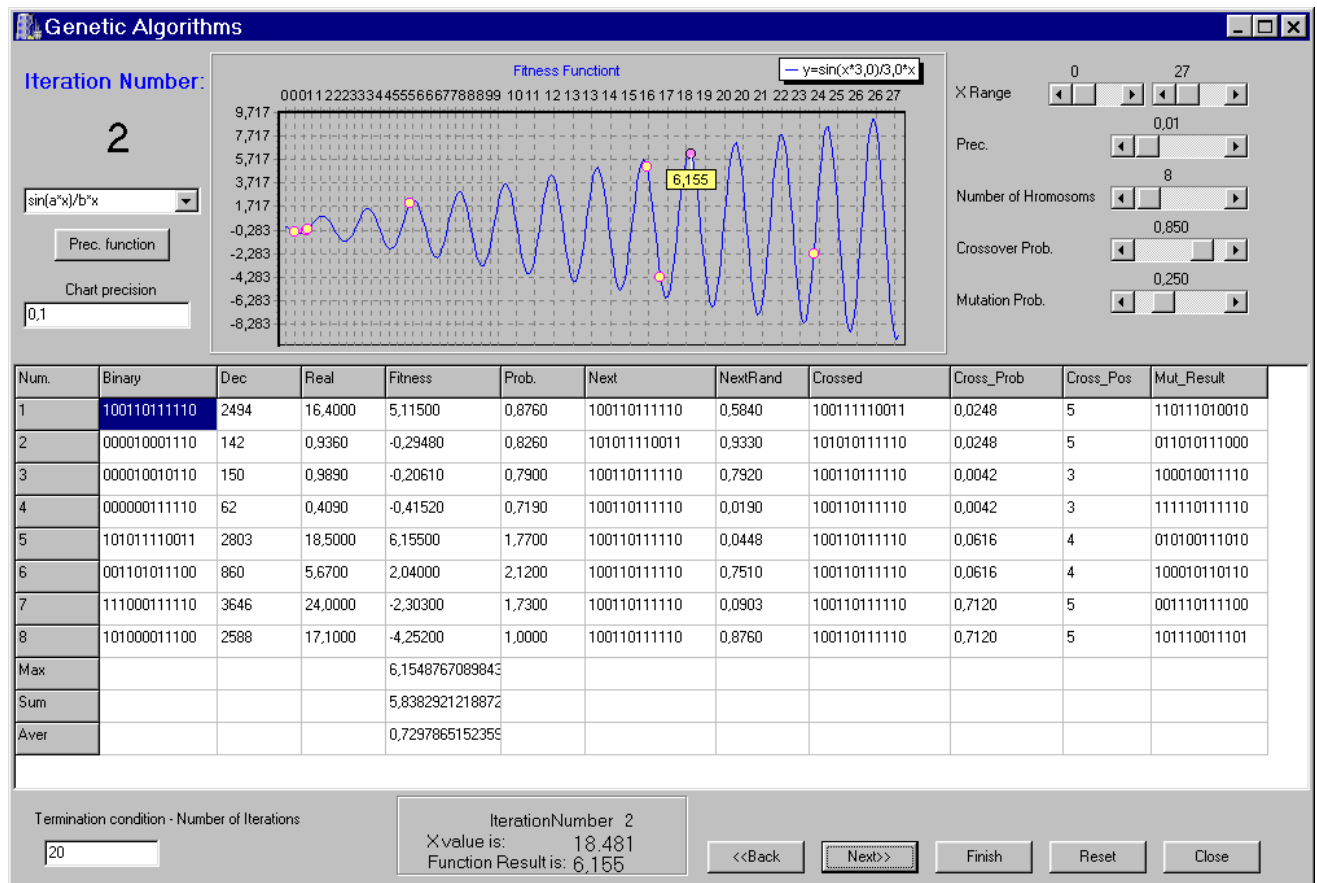
Также можно увидеть наилучшее на данный момент значение аргумента, соответствующее ему значение Fitness function и итерацию, на которой оно было найдено.

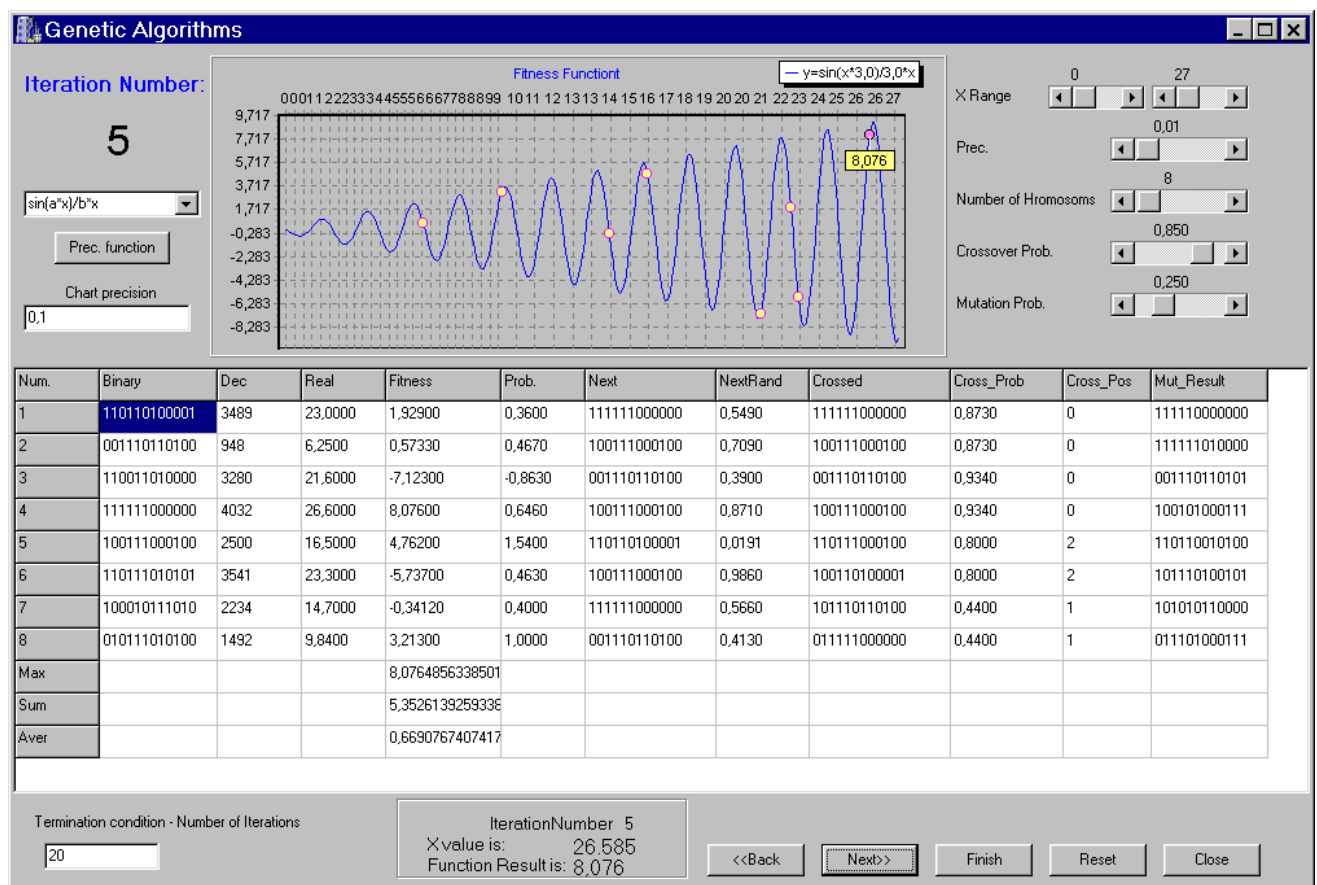
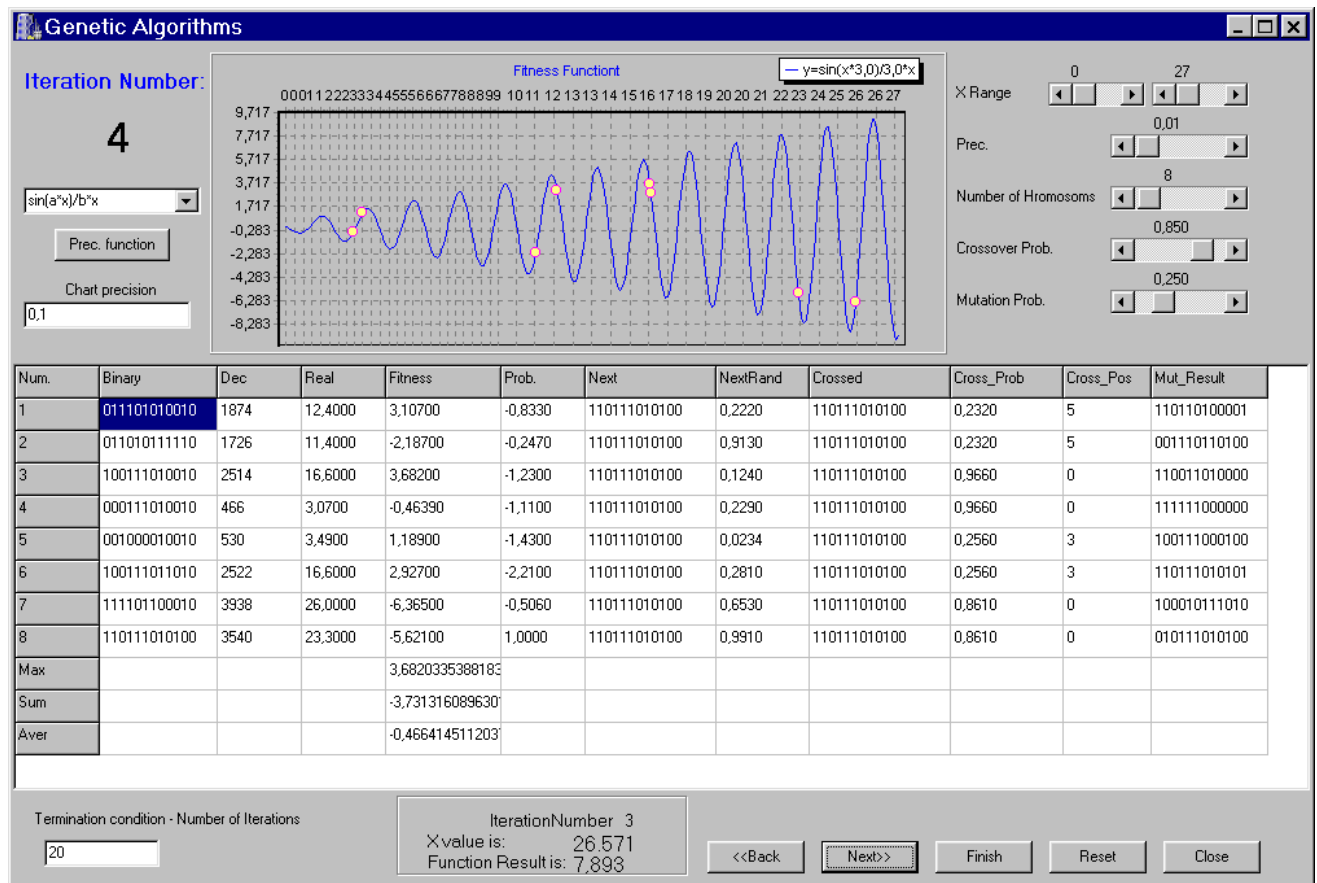
IterationNumber 1
Xvalue is: 18.837
Function Result is: 1416,410

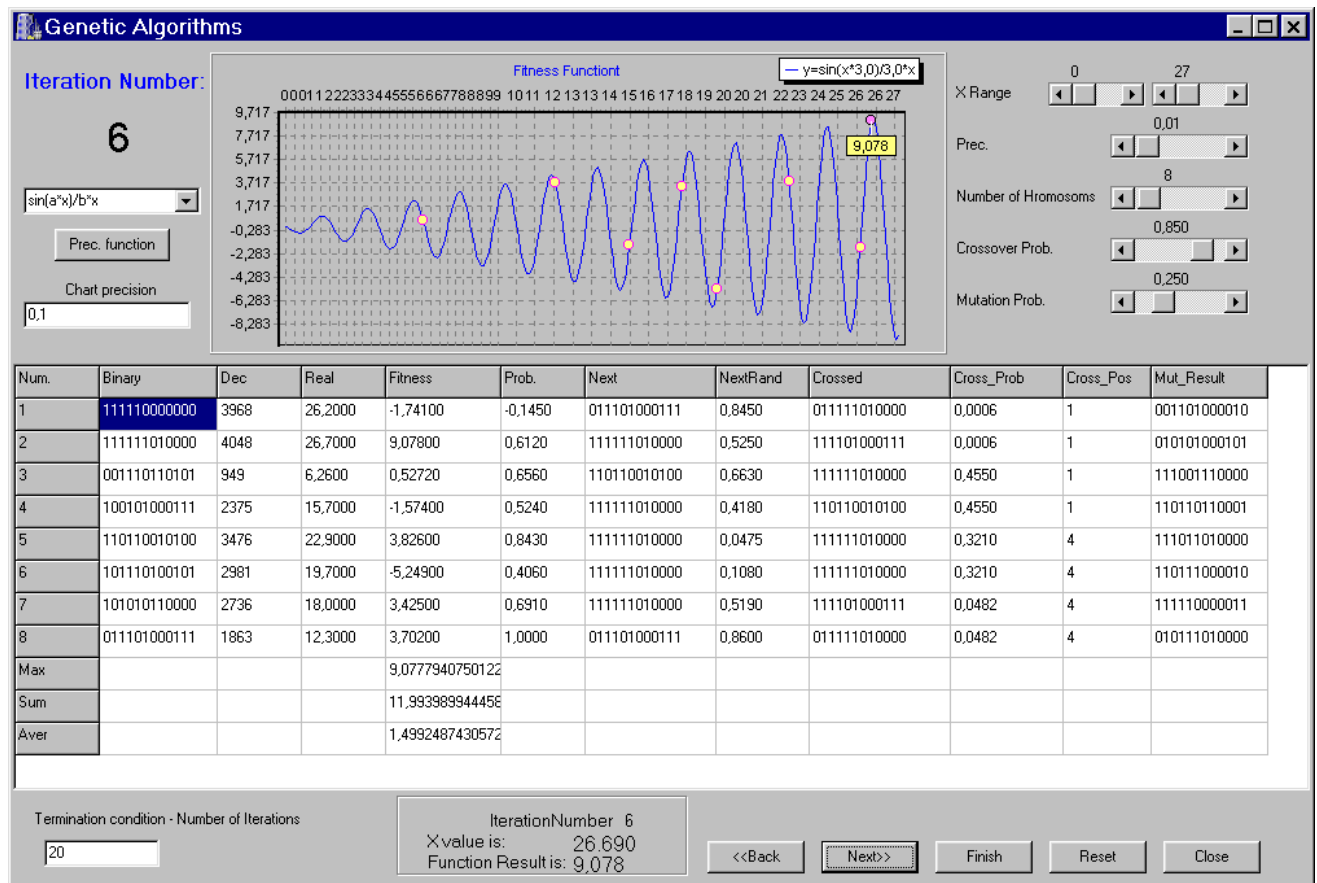
2.2. Пример решения конкретной задачи

Проиллюстрируем работу программы на следующем примере: $y = \sin(3 \cdot x) / x^3$.









В итоге мы получили следующие результаты: $X = 26,690$, $Y = 9,078$.

Эта точка является самой близкой к максимуму функции $y = \sin(3 \cdot x)/x^3$. В течение следующих 14 поколений этот результат так и не изменился. Для получения лучших результатов можно воспользоваться большим количеством хромосом в популяции.

3. Заключение

В результате выполнения лабораторной работы были изучены основы работы генетических алгоритмов. Был разработан программный продукт для решения конкретной задачи «Поиска максимума функции». Применение алгоритма было проиллюстрировано на конкретном примере $y = \sin(3x)/x^3$.

В результате истории 6 поколений были получены следующие результаты: $X = 26,690$, $Y = 9,078$. Эта точка является наиболее близкой к максимуму функции $y = \sin(3 \cdot x)/(x^3)$.

Для достижения лучших результатов нужно было бы взять большую популяцию.

4. Используемая литература

[1] Круглов В.В., Борисов В.В. Искусственные нейронные сети. Теория и практика. –М: Горячая линия – Телеком, 2001. – 382 с.: ил.

[2] Goldberg, D.E., Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Redwood City, 1998