Генетические алгоритмы.

Ссылка на видеокурс от МИАН

Второй рассматриваемый метод коллективной оптимизации — **генетические алгоритмы**. Это крайне эффективный метод, который применяется для решения широкого класса прикладных задач, в частности для плохо формализуемых или некорректных.

Постановка задачи и идея метода

Еще в 1950 в своей большой работе "Может ли машина мыслить?" **Алан Тьюринг** (1912 – 1954) задавался вопросом искусственного интеллекта машин. В 1975 году **Джон Холланд** (1929 – 2015) публикует свою известную монографию "Adaptation in Natural and Artificial System", в которой популяризирует идею эволюционных вычислений. Однако в то время вычислительные ресурсы не давали возможность осуществлять генетические алгоритмы и эволюционные вычисления, но в нынешнее время с развитием техники эти методы начинают играть огромную роль. У них есть и свои недостатки, но в ряде случаев они показывают лучшие результаты, чем **нейронные сети**.

Идею для алгоритма, опять же, подсказала сама биология. Попытаемся объяснить ее простым языком. Каким-то образом в процессе эволюции из простых организмов рождались все более и более сложные. Выживали те виды живых существ, которые более приспособлены к окружающей среде и которые способны производить потомство. Эволюционная стратегия позволяла находить **оптимальный** ответ среде: рождался вид, который взял лучшее от предков и, несмотря на резкие изменения климата, мог спокойно выживать, продолжать свой род. Эволюция использует три механизма: "плохие" (с точки зрения приспособленности) виды отмирают, некоторые виды подвергаются случайным мутациям, а также, конечно же, при размножении происходит скрещевание генотипов. Эти механизмы использованы и в генетических алгоритмах для решения задач математики.

Имеется непустое множество U (от англ. universe) — всевозможные ответы на внешнюю среду, особи. Задана функция $F : U \to \mathbb{R}$, которую (без ограничения общности) будем считать неотрицательной $F \geq 0$. Для нашей задачи в случае отрицательности функции можно просто произвести смещение на сколь угодно большую константу, тогда F будет удовлетворять условию неотрицательности (бессмысленно искать минимум у той функции, которая неограничена снизу). Необходимо найти $x^* \in U$ такую, что $F(x^*) = \min\limits_{x \in U} F(x)$. Нужно также отметить, что $U \subset \mathbb{R}^n$, то есть каждый элемент представляет собой n-мерный вектор, который может быть разбит на составные части. Стандартная постановка задачи многомерной оптимизации.

Некоторое приближение к решению $x = (x_1, \ldots, x_n) \in U$ можно назвать **генотипом**, а компоненты вектора $x_i, i = 1, \ldots, n$ — **генами** (или хромосомами, что в нашем случае синонимы). Число $F(x) \in \mathbb{R}$ можно назвать **фенотипом**, то есть признаком, определяемым функцией F — насколько та или иная особь приближена к глобальному минимуму. Набор особей с генотипами $(x^1, x^2, \ldots, x^M) \subset U$ можно назвать геномом, а множество U — генофондом.

Генетические операции и функция приспособленности

Для того, чтобы определить оптимальное решение, алгорим пользуется теми механизмами, которые подсказаны эволюцией.

1. Добавление в популяцию некоторой случайной особи $\tilde{x} \in U$.

   Фиксируем некоторое вероятностное пространство $\langle \Omega, \mathcal{A}, \mathbb{P} \rangle$. Чаще всего множество U является счетным или конечным, поэтому его можно представить как $U = \bigcup\limits_{s \in S} x_s$, где $S \subset \mathbb{R}$. Рассмотрим некоторую случайную величину ξ, измеримую относительно $\mathcal{A}$, которая реализуется (по некоторому определенному закону) для некоторого $\omega \in \Omega$ и принимает значение $\xi(\omega) \in S$. То есть порождаемый $\tilde{x} = x_\xi$.

Вообще говоря, эта операция является искусственной, она в природе не встречается, но для генетических алгоритмов она весьма применима.

2. Операция скрещивания (**кроссинговер**, crossover).

   Под скрещиванием мы будем понимать операцию создания новой особи в пуле, у которой часть хромосом от одной родительской особи, а часть хромосом от другой.

   Вводим оператор $H : U \times U \to U$, такой, что $x, y \in U \to H(x, y) \in U$. Это оператор кроссинговера.

   Операция скрещивания является основной в генетических алгоритмах. В дальнейшем мы увидим, что те гены, которые являются лучшими относительно оптимизируемой функции, чаще участвуют в скрещивании.

3. Операция **мутации**.

   Под мутацией будем понимать операцию, когда у особи случайным образом изменяются одна или несколько хромосом.

   Снова фиксируем некоторое вероятностное пространство $\langle \Omega, \mathcal{A}, \mathbb{P} \rangle$. Вводим случайный оператор мутации $G : U \times \Omega \to U$, который в зависимости от реализации некоторой случайной величины меняет некоторую особь.

   Довольно часто используется жесткий механизм мутации, который состоит в том, что выбранная особь полностью заменяется на случайную другую. Мутация способна как улучшить особь и, соответственно, популяцию, так и ухудшить.

Мы будем одновременно оптимизировать не один генотип, а целый геном, причем пул из особей будет эволюционировать по приведенным выше правилам.

В генетическом алгоритме нам также нужно будет ранжировать особи по их качеству, то есть близости к оптимальному решению. Для этого введем **функцию приспособленности** (Fitness function):

$$Fit(x) = \frac{1}{1 + F(x)},$$

которая оценивает особь по ее близости к оптимуму (по приспособленности). Она хороша тем, что она безразмерна, причем она нормированна, так как

$$0 < Fit(x) \leq 1,$$

поскольку $F \geq 0$. Нормализация данных довольно частый и полезный прием в машинном обучении. Будем говорить, что особь x лучше, чем особь y, если $Fit(x) > Fit(y)$. То есть требуется максимизировать функцию Fit.

Приведенный выше вариант функции приспособленности не является единственным верным, можно подобрать и другую функцию, главное, чтобы ее можно было довольно легко и быстро вычислить, поскольку это придется делать много раз за время работы алгоритма.

Алгоритм

Пусть M — количество особей в пуле.

1. Создаем пул (популяцию) из случайных особей $(x^1, x^2, \ldots, x^M) \in U^M$. Особи в пуле могут повторяться.

   Популяция создается по следующему рецепту:

   $$x^m = (\xi_1, \ldots, \xi_n),\ \xi_i \sim R(a_i, b_i),\ i = 1, \ldots, n,\ m = 1, \ldots, M.$$

   Здесь ξ_i — это случайная величина, равномерно распределенная на отрезке $[a_i, b_i]$. Естественно, границы для U задаются в зависимости от решаемой задачи.

2. Для каждой особи рассчитываем ее функцию приспособленности.

3. Ранжируем (упорядочиваем) популяцию по значениям их функции приспособленности таким образом, чтобы
$$Fit(x^1) \geq Fit(x^2) \geq \cdots \geq Fit(x^M).$$
На первом месте располагается лучшая (или одна из лучших) с точки зрения приспособленности особь.

4. "Убиваем" часть худших особей с номерами $m = K, K+1, \ldots, M$, причем $K < M$ (обычно отсекается примерно 20% популяции, не больше, иначе алгоритм превратится в простой случайный поиск). Заменяем их на новые случайные особи $x^m \in U, m = K, K+1, \ldots, M$.

4*. Либо же производим мутацию некоторых случайных особей. Применяем оператор G, например, для особей с номерами $m = K, K+1, \ldots, M$.

Мутацией особи $\tilde{x} \in U$ является особь $G(\tilde{x}, \omega) \in D$, которая строится по следующим правилам:

- В зависимости от ω выбирается номер $1 \leq j \leq n$.
- Вычисляется случайная величина $\eta_j = \eta_j(\tilde{x}_j)$ такая, что $\tilde{x}_j + \eta_j \in [a_j, b_j]$.

Особь $G(\tilde{x}, \omega) = \tilde{y}$ состоит из компонент
$$\tilde{y}_i = \begin{cases} \tilde{x}_i, & i \neq j \\ \tilde{x}_i + \eta_i, & i = j \end{cases}$$

5. Для каждой новой рассчитываем ее функцию приспособленности.

6. Для особей с номерами $m = 2, \ldots, M$ подбираем пару для скрещивания. Мы оставляем нетронутой наилучшую в текущем пуле особь — благодаря этому на каждой итерации решение не будет ухудшаться.

Для особи x^m находим пару x^{δ_m}, где δ_m — случайная величина, $\delta_m \in \{1, \ldots, M\}$, причем
$$\mathbb{P}\{\delta_m = l\} = p_l = \frac{Fit(x^l)}{Fit(x^1) + Fit(x^2) + \cdots + Fit(x^M)}.$$

Ясно, что $\sum\limits_{l=1}^{M} p_l = 1$. Это так называемый метод колеса фортуны: вероятность p_l тем больше, чем больше $Fit(x^l)$ — более приспособленная особь чаще будет оказываться в парах и участвовать в скрещивании. Поэтому лучшая на данный момент особь чаще других будет второй в паре. Заметим также, что может быть так, что $m = \delta_m$, тогда особь при скрещивании не меняется.

Оператор скрещивания $H(x^m, x^{\delta_m}) = \tilde{y}$ можно задать двумя способами:

- Допустим, что приспособленность x^m в два раза больше, чем приспособленность x^{δ_m} — $Fit(x^m) > Fit(x^{\delta_m})$. Тогда $\tilde{y}$ возьмет 2/3 генов от x^m и 1/3 генов от x^{δ_m}. В реальности, обычно, берется половина генов от одной родительской особи, а другая половина от другой, но мы будем отдавать предпочтение более сильной особи, так как мы моделируем эволюционный процесс, который должен развиваться, а не стагнировать.
- Второй путь дает большее смешивание генов. Фиксируем вероятностное пространство $\langle \Omega, \mathcal{A}, \mathbb{P} \rangle$ и расширяем наш оператор скрещивания $H : U \times U \times \Omega \to U$.
Потомок $\tilde{y} = (\tilde{y}_1, \ldots, \tilde{y}_n)$, у которого
$$\tilde{y}_i = \begin{cases} x_i^m, \text{ с вероятностью } \dfrac{Fit(x^m)}{Fit(x^m) + Fit(x^{\delta_m})} \\ x_i^{\delta_m}, \text{ с вероятностью } \dfrac{Fit(x^{\delta_m})}{Fit(x^m) + Fit(x^{\delta_m})} \end{cases}, \qquad i = 1, \ldots, n,$$
почти наверняка возьмет больше генов у более приспособленной особи.

7. Переходим к шагу 2.

Алгоритм позволяет делать множество модификаций, причем все они рабочие и эффективные. Это дает свободу для аналитика, в зависимости от задачи он может крутить этот метод так, как ему захочется и выбирать оптимальную схему реализации.

Отметим также, что каждый из генов особи должен вносить равносильный вклад в минимизацию, то есть независимо друг от друга. Генетический алгоритм не будет работать или будет работать плохо, если минимизация одной компоненты играет большую роль, чем минимизация других. Поэтому нежелательно, чтобы гены были жестко связаны таким образом.

Как и в прошлые разы мы не указываем критерия остановки. Они могут иметь различную природу, например, на количество поколений (итераций или эпох) или на качество найденного приближения.

Генетические алгоритмы и поток σ-алгебр

На начальном этапе мы формируем некоторую популяцию $(x^1, \dots, x^M)$, и на основе полученной с помощью функции Fit информации можем построить некоторую начальную σ-алгебру, рассчитать мат. ожидание. В дальнейшем мы меняем нашу популяцию с помощью описанных операций скрещивания, мутации и нарождения, в то же время σ-алгебра также меняется. Так как нам становится известно больше информации о приближенности к оптимальной точки, σ-алгебра расширяется. Таким образом мы строим поток σ-алгебр (фильтрацию) стохастического анализа.

Пример 1. Диофантово уравнение

Нахождение целого решения уравнения

$$x^2 - 2x + y^2 - 2yz + 2z^2 - 4z + u^2 - 6u + 14 = 0$$

можно свести к задаче минимизации.

Преобразуем левую часть:

$$\begin{aligned}
&(x^2 - 2x + 1) - 1 && + \\
&(y^2 - 2yz + z^2) - z^2 && + \\
&z^2 + (z^2 - 4z + 4) - 4 && + \\
&(u^2 - 6u + 9) - 9 && + \\
&14 && = \\
&(x-1)^2 + (y-z)^2 + (z-2)^2 + (u-3)^2 &&
\end{aligned}$$

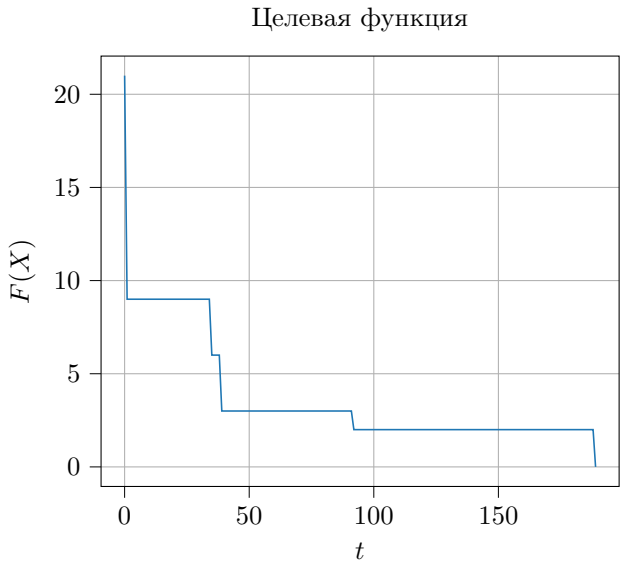
Как видим, левая часть неотрицательная, значит в нуле — минимум этой функции. Минимум достигается тогда, когда все квадраты, участвующие в сумме, равны нулю, поэтому ответ — $(1, 2, 2, 3)$.

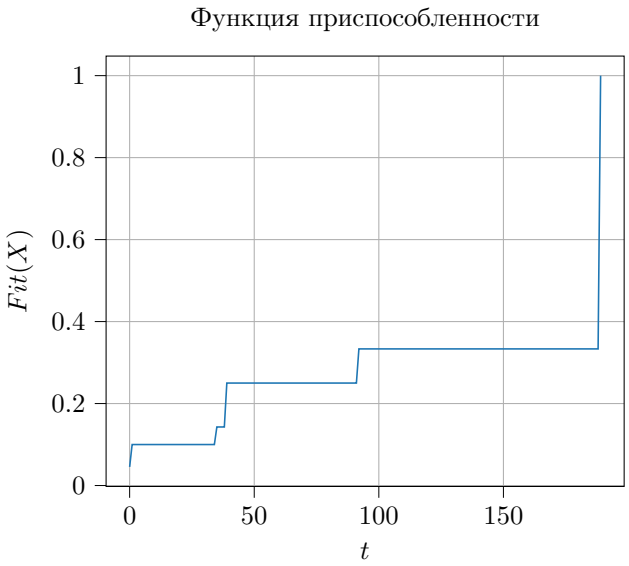
Посмотрим, как генетический алгоритм справится с уравнением. Будем использовать следующие параметры:

1. Количество особей в популяции: $M = 20$.
2. Количество отсекаемых худших особей: $K = 4$ (20%).
3. Критерий окончания: $F(X) = 0$.
4. Границы выбора случайных генов: $[-10, 10]$.

Протокол вычислений представлен на графике значений целевой функции и значений функции приспособленности. Для нахождения решения потребовалось 190 эпох.

В течение большого количества эпох лучшая особь не меняется, но внутри пула происходят постоянные процессы по выращиванию более приспособленной особи. В результате появляется новый лидер и значение целевой функции достигает искомого минимума. Заметим также, что значение функции не ухудшается, по причине того, что лучшая особь остается нетронутой при скрещивании. И в этом и в последующих примерах используется механизм жесткой мутации, когда слабые особи полностью заменяются на новые.





Пример 2. Минимизация многомерной функции

Рассмотрим функцию

$$F(\mathbf{x}) = \sum_{i=1}^{n} x_i^2 \cdot (1 + |\sin(100 x_i)|),$$

заданную на $\mathbb{R}^n$. Положим $n = 5$.

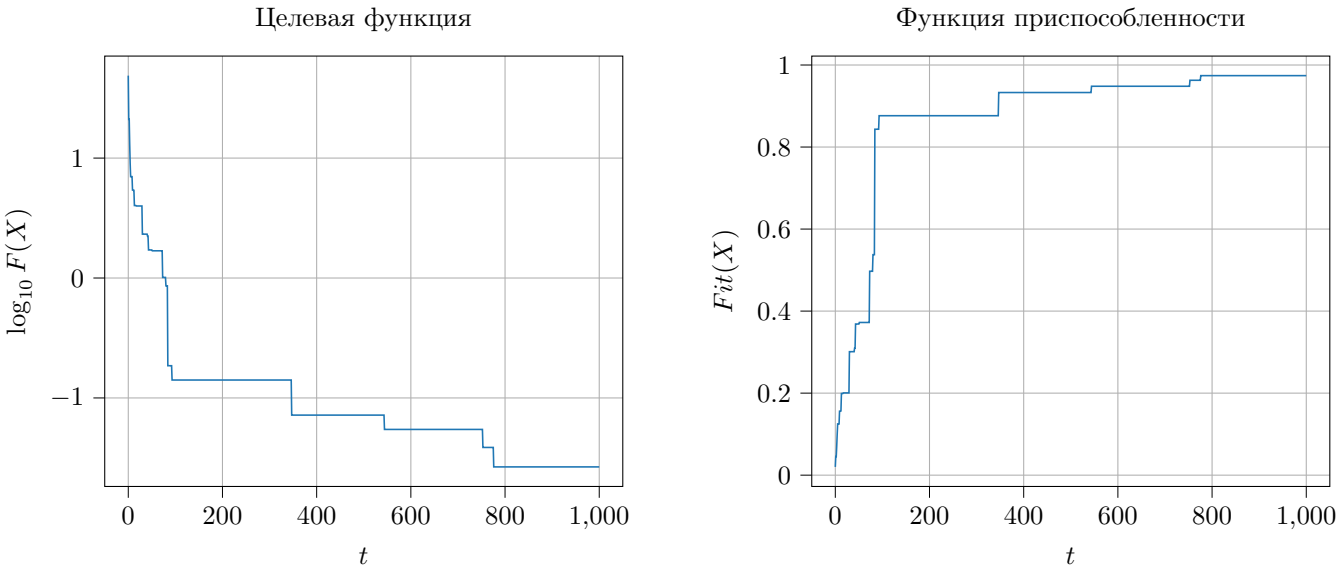
Функция имеет большое количество локальных минимумов, ее глобальный минимумом достигается в точке $\mathbf{x} = \mathbf{0}$.

Будем использовать следующие параметры:

1. Количество особей в популяции: $M = 50$.
2. Количество отсекаемых худших особей: $K = 10$ (20%).
3. Количество эпох: $L = 1000$.
4. Границы выбора случайных генов: $[-10, 10]$.

Протокол вычислений представлен ниже, значения целевой функции представлены в логарифмическом масштабе для того, чтобы лучше разглядеть динамику при близких к нулю приближениях. Найденное приближение имеет точность до $3 \cdot 10^{-2}$.

И в этом и в прошлом примерах мы наблюдаем ситуацию, когда лидер долгие поколения не покидает свое место, что приводит к ситуации стагнации. В реальных проектах эти простои не очень желательны, поэтому иногда бывает полезно “встряхнуть” популяцию и скрестить также и монополиста. Для этого можно добавить доп. условие, контролирующее эту ситуацию.



Пример 3. Задача о разбиении

У нас есть конечное упорядоченное множество натуральных чисел A (например, монеты разного достоинства). Числа могут повторяться. Необходимо разбить данное множество на два подмножества

$$A = K_1 \cup K_2, \quad K_1 \cap K_2 = \varnothing$$

таким образом, чтобы

$$F(K_1, K_2) = \left| \sum_{a_{k_1} \in K_1} a_{k_1} - \sum_{a_{k_2} \in K_2} a_{k_2} \right|$$

достигала минимального значения. В идеале это минимальное значение — 0, но это не всегда возможно.

Хорошо известно, что эта задача является NP-полной. Введем кодировку к данной задаче. Пусть мощность $|A| = N$. В качестве генотипа введем N-мерный вектор $x = (x_1, \ldots, x_N)$, каждая компонента которого принимает значение либо 0, либо 1. При этом если $x_k = 1$, то будем считать, что элемент $a_k \in A$ принадлежит множеству K_1, в противном случае — множеству K_2.

Рассмотрим применение генетического алгоритма в случае, когда $N = 1000$ и

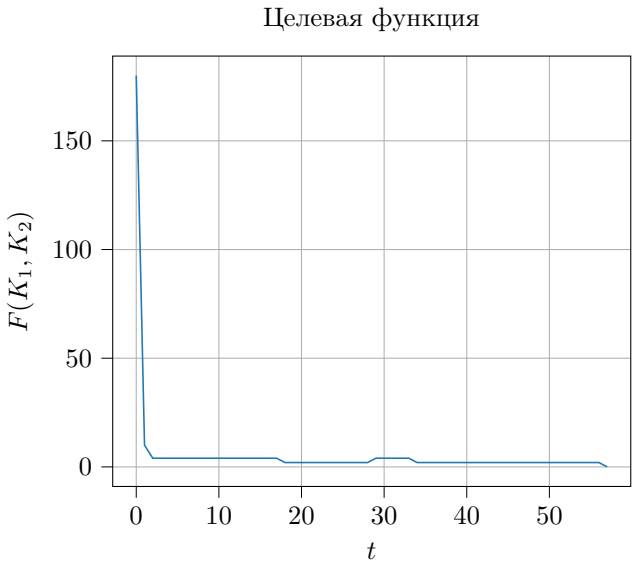
$$A = (1, 2, 3, \ldots, 1000).$$

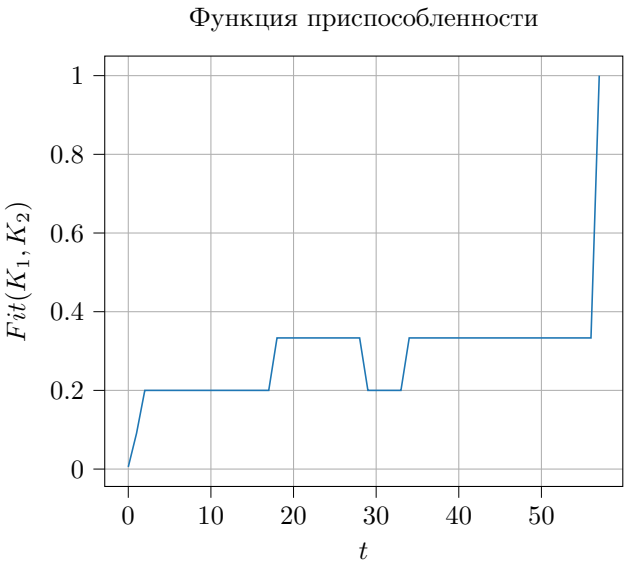
Будем использовать следующие параметры:

1. Количество особей в популяции: $M = 500$.
2. Количество отсекаемых худших особей: $K = 100$ (20%).
3. Критерий окончания: $F(K_1, K_2) = 0$.

Результаты вычислений приведены ниже на графиках. Решение найдено за 58 итераций алгоритма.

В отличие от предыдущих примеров, в этом мы контролируем ситуацию с неизменным лидером. Если главарь держится на своем месте более десяти итераций — проводим кроссинговер и для него. Это временно ухудшает ситуацию, но затем приближение быстро сходится к искомому оптимуму.





Пример 4. Имитация задач менеджмента

В кадровом менеджменте принципиален вопрос кого и на какую должность назначать, ведь от этого назначения зависит будущее предприятия: неоптимальное распределение приведет к плачевным последствиям, а оптимальное — к успеху. В этом деле важную роль играет функция приспособленности. С ее помощью мы оцениваем кандидата на некоторую важную или побочную должность. Оценка кандидата априори — только по его резюме или по итогам интервью — может оказаться ложной, если речь идет о должности высокой важности. Необходимо посмотреть на него в работе, особенно важна здесь именно работа в команде. Тогда на помощь приходит генетическое моделирование. Менеджер формирует команду из кандидатов, каждому назначает какую-то должность и дает задание. Следит за процессом и оценивает результат. После этого он отбраковывает самых неэффективных, и каждому снова выдает по новой должности. Затем снова происходит оценка. То есть в этой деловой игре моделируется своеобразный генетический алгоритм, на выходе которого после нескольких итераций получается оптимальное распределение кадров для последующей эффективной работы.

Другой похожий пример. Допустим, что мы имеем крупное наукоемкое предприятие, у которого бОльшая часть средств из баланса уходит на научно-исследовательские и опытно-конструкторские работы (НИОКР). У этого предприятия появляется проект, который нужно реализовать эффективно, но быстро. Обычно, для таких проектов не требуется огромное число сотрудников — места становятся вакантными, на них устраивается конкурс. Временно претенденты из числа сотрудников начинают работать над этим проектом за свою з/п, впоследствии самые эффективные продолжают работу, завершают ее и получают отдельное вознаграждение. Ресурсы предприятия позволяют менеджементу проводить деловую игру для определения подходящих сотрудников. В игре также будут применены принципы генетического моделирования: разделение на несколько команд, оценка приспособленности, отбраковка худших, добавление новых, перемешивание членов разных команд — поиск оптимальной команды. После 2-3 месяцев становится понятно, кто должен продолжить работу над инновационным проектом.

В нашем моделировании будет 6 кадидатов/6 команд-кандидатов $x = (x_0, x_1, x_2, x_3, x_4, x_5)$ и оптимальной будет конфигурация $x = (0, 1, 2, 3, 4, 5)$. В качестве функции $Fit(x)$ будет имитация некоторой оценки эффективности: $+\frac{100}{1+x_i}$ в случае правильного определения позиции и $+\frac{1}{1+x_i}$ — в случае неправильного.

1. Количество особей в популяции: $M = 20$.
2. Количество отсекаемых худших особей: $K = 4$ (20%).
3. Критерий окончания: нашли оптимум.

За 8 эпох найдено оптимальное распределение. В реальности итераций должно быть примерно столько же, а лучше еще меньше. Как мы убедились, большая часть успеха состоит в правильном подборе функции приспособленности.

Вместо заключения

Генетический алгоритм — мощный инструмент для решения огромного числа задач. В его эффективности можно убедиться на примерах. Все представленные примеры запрограммированы, код и комментарии к нему можно посмотреть в приложении.