

Проект. Планирование движения материальной точки

В качестве темы для проекта по предмету "Методы оптимизации" был выбран 15-ый вариант:
Планирование движения материальной точки с использованием эволюционного программирования (генетический алгоритм с мутациями).

Постановка задачи

Теоретическая формулировка

Материальная точка (робот) массы $m = 1$ покоится в точке $A(0,0)$ плоскости Oxy . Нужно привести робота в точку $B(1,1)$ и остановиться в ней за минимальное время. Роботу запрещается покидать квадрат $[0,1] \times [0,1]$, также роботу запрещается заезжать в круги вида $(x - X_j)^2 + (y - Y_j)^2 \leq R_j^2$. На робот действует единственная сила F , величину и направление которой мы можем произвольно изменять $|F| < F_{\max}$.

Данные

Входные данные алгоритма:

- максимальная сила F_{\max}
- шаг времени d_t
- центры помех $P_j(X_j, Y_j)$ и радиусы R_j .

Выход алгоритма:

- последовательность значений сил в каждый момент времени: F_{xk}, F_{yk} .

Полигон для тестирования

Перед тем как начать разработку алгоритмов, нужно реализовать возможность визуализации результатов выполнения. Во-первых, это упростит работу с восприятием результатов. Во-вторых, графическое представление дает возможность пользователю проанализировать ответ на наличие дефектов, выявить их источники, что сделать анализатором на уровне программы намного сложнее.

По условию задачи, используется закрытая область 1×1 . Это значит, что все координаты будут вещественными и для отображения нужно вводить ограничения точности.

Отображение результата выполнения выполняется следующим образом:

1. Область пользовательского интерфейса закрашивается черным квадратом (далее - Поле).
Допускается, что сторона является единицей измерения.
2. Для каждой зоны генерируется множество точек, которые описывают круг вокруг этой зоны. Эти точки задаются координатой $x, y \in [0;1]$ и в масштабе отображаются на поле.

3. Выбирается интересующий путь, который представлен набором точек и по одно отображается на поле.

Решение применением только генетического алгоритма

В первом приближении, предполагаем, что можно взять случайный набор команд и оптимизировать его генетическим алгоритмом по средствам мутации. Данное решение будет является достаточно абстрактным и не требовательным к дополнительной логике генерации решения.

Перед выбором способа формализации алгоритма под генетические изменения, нужно определить как алгоритм влияет на решение, какая информация при этом предоставляется. Рассмотрим простой случай: управляемый объект находится в точке (x_i, y_i) и ждет команды алгоритма. Зная все решения, которые были приняты на прошлых шагах, можно посчитать также направление и скорость объекта.

Рассмотрим два подхода к представлению алгоритма в данной задачи:

1. Сгенерируем цепочку команд, которая будет применяться на объект до тех пор, пока алгоритм не закончит выполнение.
2. Создадим зависимость принятого решения от данных, которые предоставляются на каждом шаге $f_i(p_i, v_i)$.

Анализируя первый способ, можно прийти к заключению, что его обучение не является достаточно эффективным: если есть набор принимаемых решений на каждом шаге (в нашем случае f_i), то небольшое изменение первого решения может сильно исказить эффективность всех следующих.

Второй способ позволит не зависеть от принятых решений в прошлом, а смотреть только на текущее состояние. Это позволит сделать итоговое решение задачи более устойчивым при изменении одного элемента.

Основные недостатки такого подхода заключаются в том, что для всех пар (p_i, v_i) нужно хранить решение. Т.е. количество генов очень большое и, с большой вероятностью, избыточно для отдельно взятого решения. Помимо расходов памяти для алгоритма, это чревато очень медленным процессом мутации так, как вероятность изменения каждого гена очень мала.

Формализация задачи к теории генетических алгоритмов

Введем основные понятия [1, с. 126]:

- Ген - атомарный элемент генотипа и/или хромосомы
- Хромосома - упорядоченный набор генов
- Генотип - набор хромосом данной особи
- Аллель - значение конкретного гена, значение его свойства
- Особи - хромосомы с закодированными в них параметрами задачи
- Популяция - множество особей

Стоит отметить, что в классических гены представлены значением $\{0, 1\}$, то есть хромосомы - двоичные вектора. Но это не строгое ограничение, в тех же эволюционных алгоритмах используются действительные числа в качестве значений генов [1, с. 207-208].

Для выбранного алгоритма, хромосомой будет набор генов, каждый из которых сопоставляется какому-то набору (p_i, v_i) . Для представления данных в программе, значения p_i округляются до какого-то ϵ . Таким образом, хромосома - это закодированный двумерный массив генов. Каждый ген представляет два значения: (f_x, f_y) , т.е. вектор силы, который нужно приложить.

В качестве начальной хромосомы берется случайный набор генов.

Одной из ключевых проблем данной задачи является выбор фитнес функции. Ввиду того, что Адамовой хромосомой является набор случайных генов, который почти всегда не является решением, на первом шаге стоит взять такую функцию, чтобы она выбирала в качестве лучших такие хромосомы, что являются максимально близкими к решению. Например, такой функцией является $f(p) = |(1, 1) - p|$, т.е. расстояние последней точки до целевой позиции.

Определимся с генетическими операциями, которые выполняются над данной популяцией. Так как фитнес функция уже определена, на ее основе можно выделить методы селекции. В данном случае была применена Элитная селекция (*Truncation selection*). Данный вид селекции подразумевает, что все хромосомы будут упорядочены по значению фитнес-функции и из них будут выбраны N-лучших.

Для всех полученных хромосом применяется генетический оператор. В нашем случае это простой оператор мутации - их хромосомы выбирается случайный ген и его значение заменяется на случайное подходящее значение. То есть, помимо исходной хромосомы появляются еще ее несколько мутированных потомков. Таким образом из набора хромосом, которые остались после селекции создается новая популяция.

Процесс вычисления фитнес функции

Как было указано ранее, в качестве фитнес функции было выбрана функция расстояния до целевой позиции. Рассмотрим процесс выполнения симуляции.

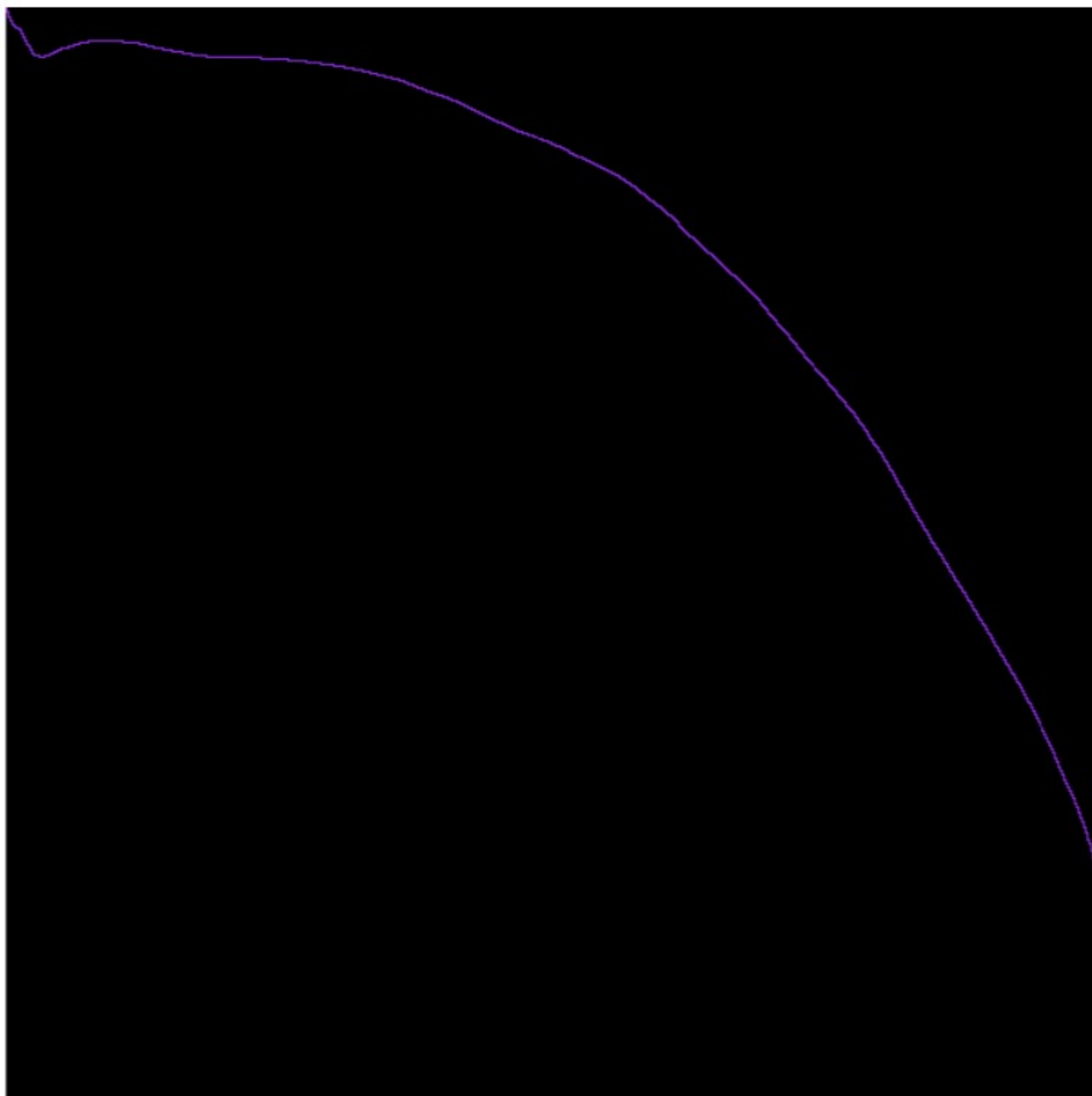
В начале выполнения объект ставится на начальную позицию $(0, 0)$. На каждом шаге делается проверка - находится ли объект внутри зоны, за пределами поля или в точке $(1, 1)$. Если хотя бы одно условие выполняется - выполнение симуляции заканчивается, возвращается информация о последней посещенной точке. Если же проверка не проходит, то выполняется один шаг. Под шагом подразумевается взятие из хромосомы нужного гена, получение с него информации и применение. Так как хромосома представляет собой зависимость $f(p_x, p_y)$, то по положению объекта в данный момент можем получить вектор силы, который нужно применить. Изменение состояния объекта:

$$F = \text{chromosome}[p_x, p_y] \setminus v = v + a \cdot F \cdot \Delta t \setminus p = p + v \cdot \Delta t$$

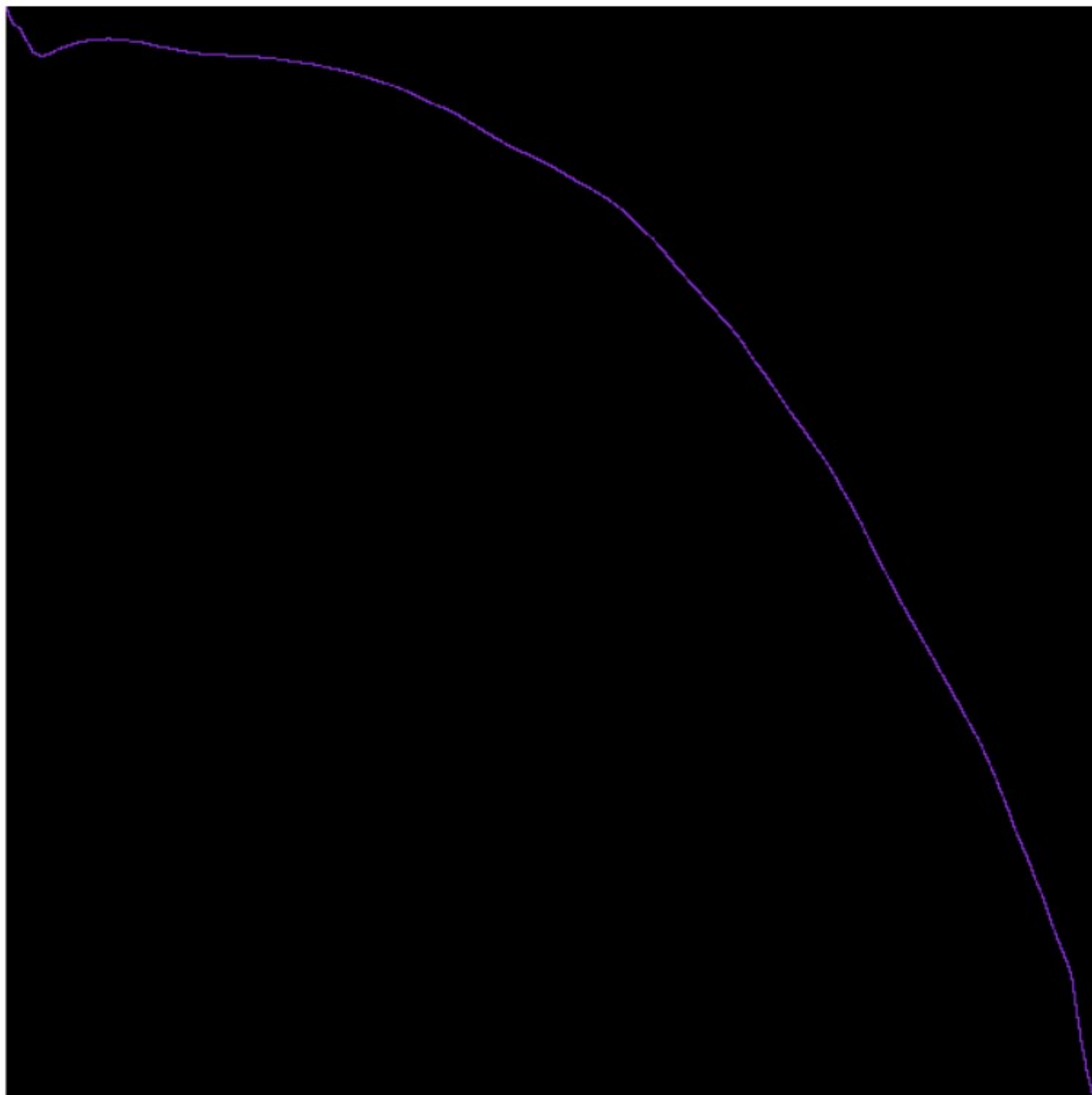
Таким образом получаем новое положение объекта и для него продолжается выполнение алгоритма. Также стоит отметить наличие верхней границы длительности выполнения т.к. случайная хромосома может иметь такой набор генов, что построение маршрута по нему будет выполняться бесконечно долго.

Примеры и результат

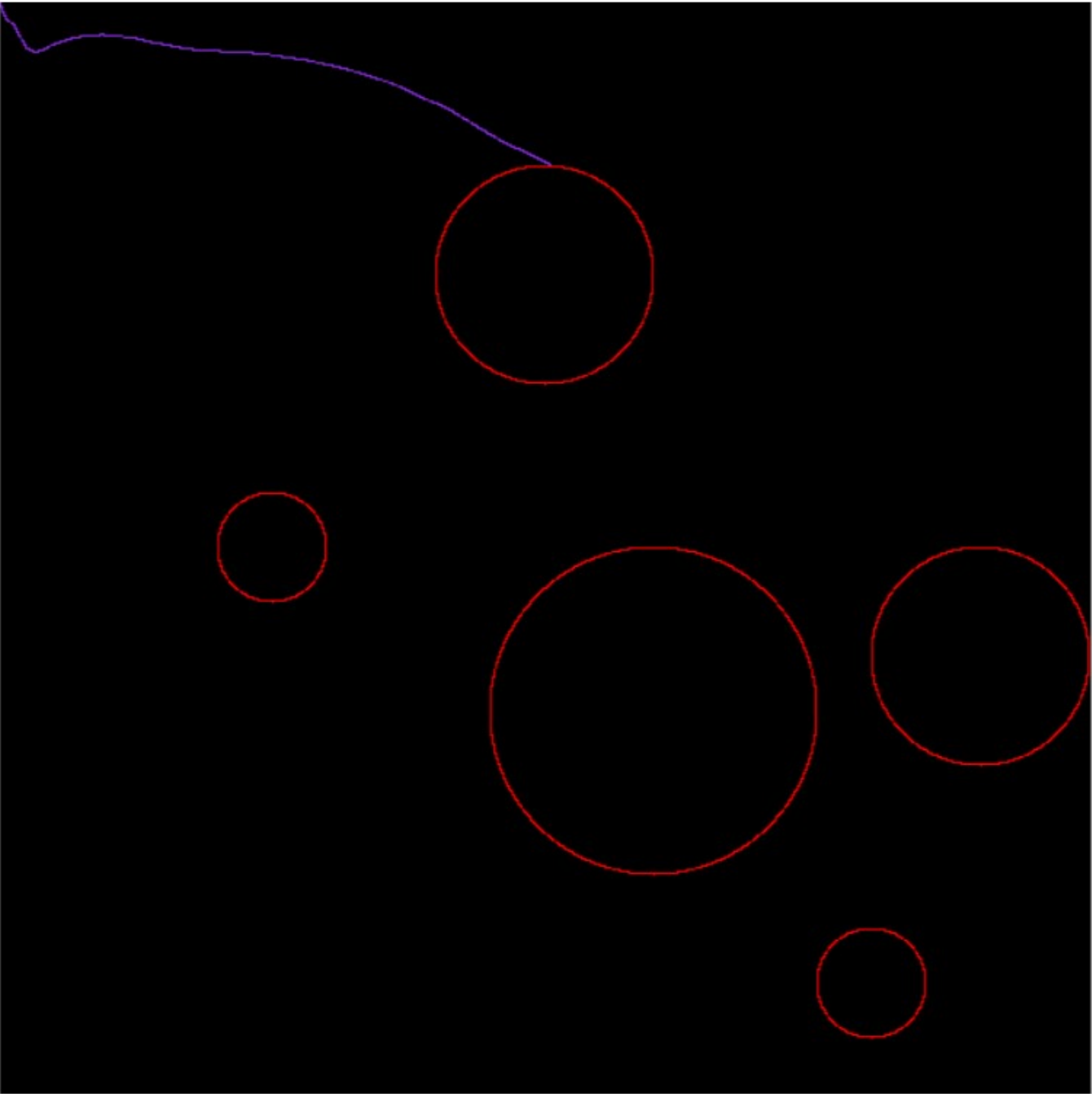
При отображении результаты выполнения случайно сгенерированной хромосомы, скорее всего будет полученная странный маршрут который к тому же может закончиться в самом начале. Но выполнение нескольких итерация генетического алгоритма можно получить что-то очень похожее на решение:

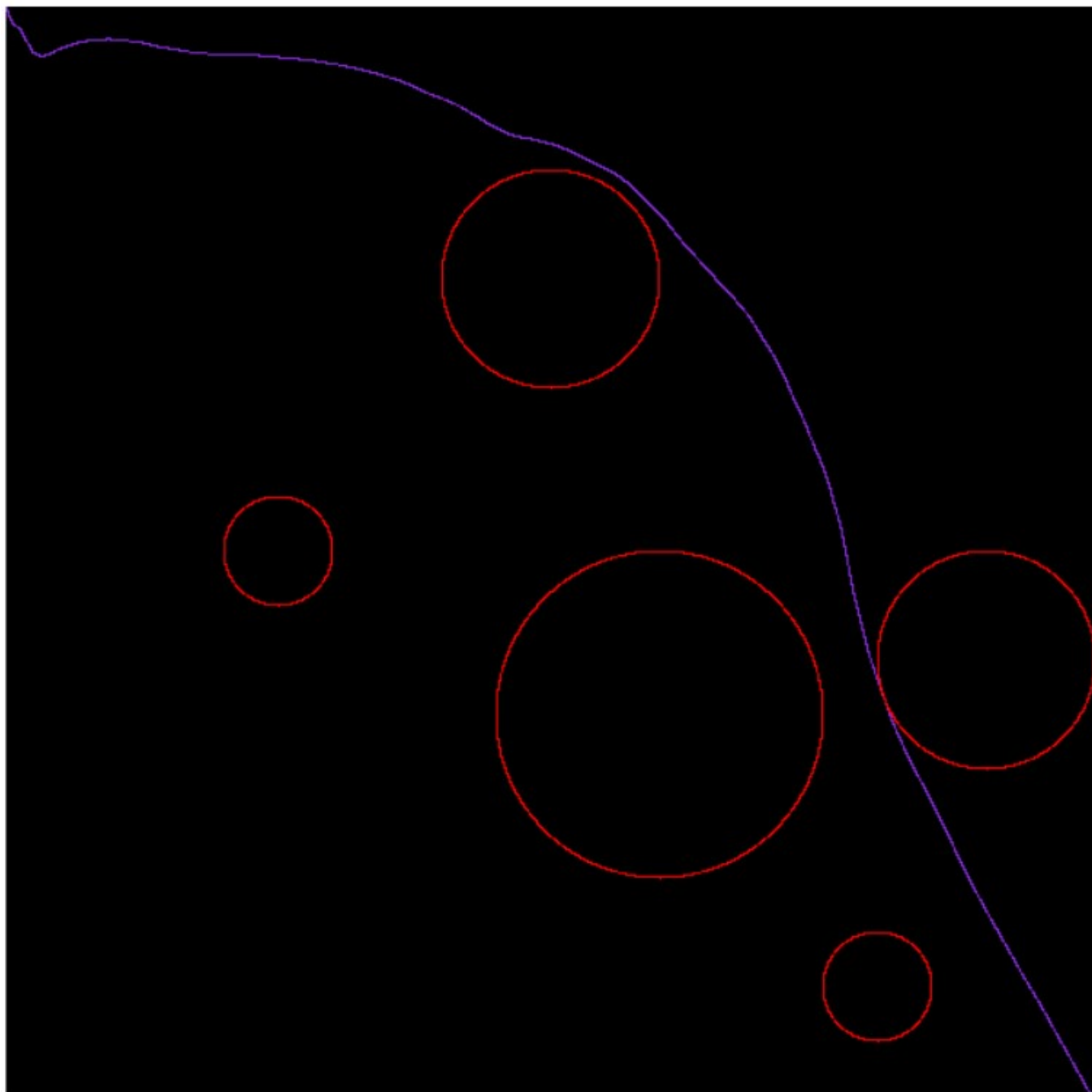


Дальнейший запуск алгоритма можем сгенерировать путь к конечной точке:



Рассмотрим поведение алгоритма после добавление зон. Если хоть одна зона будет находится на пути объекта до ранее успешный алгоритм начнет показывать очень плохие показатели фитнес функции. Но при существовании не сложных маршрутов к конечной точек, данный алгоритм все же сможет воссоздать такую хромосому, которая сможет обходить зоны:





Использованная литература

1. Рутковская. - Нейронные сети, генетические алгоритмы и нечеткие системы