Traffic light simulator and optimization

Д. Байдин

Симулятор

`https://github.com/volkhin/RoadTrafficSimulator`

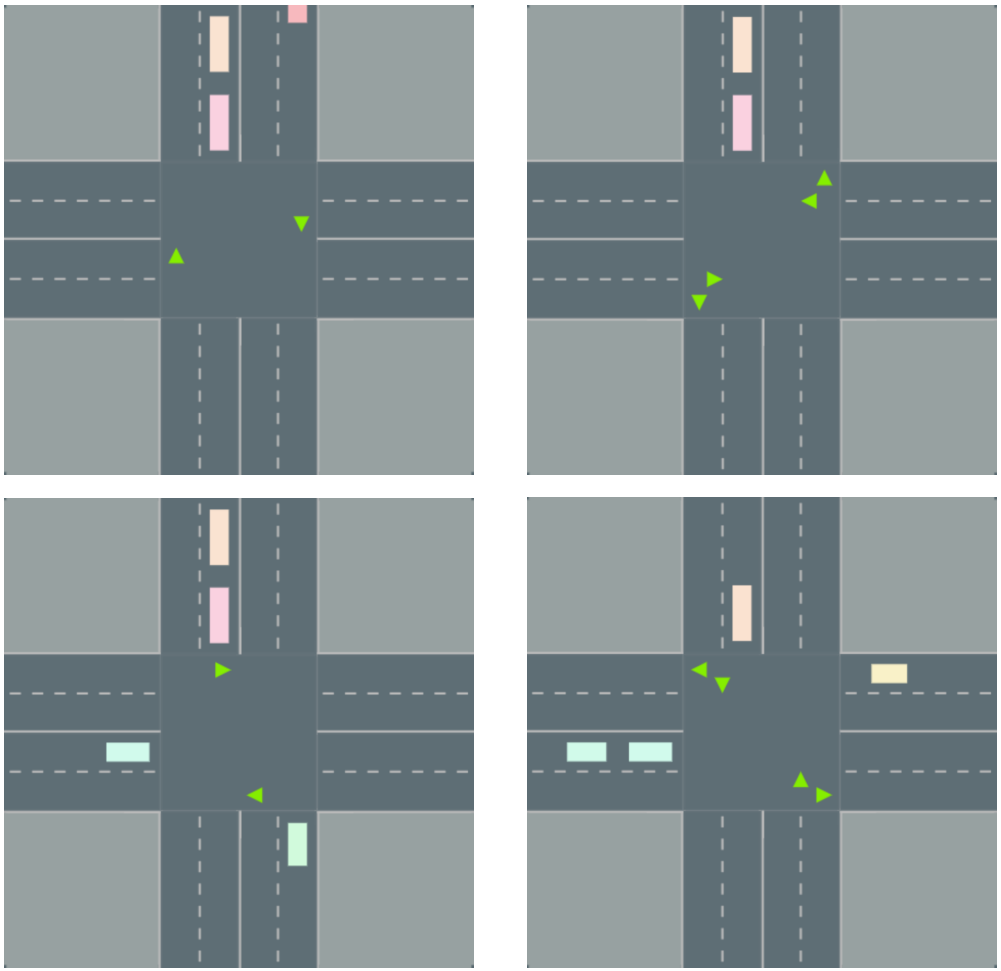
Основные проблемы:

- Автомобиль может появится на произвольном перекрестке
- Слишком простая модель перекрестка
- Отсутствует проверка столкновений автомобилей
- Карта (транспортная сеть) хранится в local storage браузера. Нет возможности сохранить карту в файл/загрузить карту из файла.

Симулятор

https://github.com/volkhin/RoadTrafficSimulator

Состояния перекрестка:



Симулятор (Изменения)

Модель:
Транспортную сеть можно представить в виде орграфа $G = \langle V, E \rangle$, где V - перекрестки, E - множество дорог.

$$S = \{v \in V \mid \deg_{out}(v) = 1\}$$

- множество истоков. $C = \{c_i\}$ - множество автомобилей.

Изменения:

- Автомобили могут появляться только на перекрестках - истоках.

Симулятор (Изменения)

- Интенсивность (λ) появления автомобилей (для каждого истока). Если $S = \{s_i\}_{i=1}^{k}$, тогда $\Lambda = \lambda_1 + \lambda_2 + \ldots + \lambda_k$ - общее количество автомобилей на карте.
  Можем определить $p_i = \lambda_i / \Lambda$ - вероятность появления автомобиля в i-ом истоке.
- Для того чтобы определить то на каком перекрестке появится новый автомобиль выполняем Sampling from discrete distributions

Симулятор (Изменения)

- Более сложная модель перекрестка.
  Перекресток $v_i \in V$ можно описать четверкой

$$\tau_i = (t_{i,1}, t_{i,2}, t_{i,3}, t_{i,4}),$$

  где $t_{i,j}$ - время работы перекрестка в состоянии j.

Симулятор (Изменения)

- Сервер на Node.js позволяет
  - сохранять карту в json файл
  - загружать карту из json файла
- Модификация модели автомобиля (выбор траектории движения)

Симулятор (Изменения)

Для каждого автомобиля c_i из множества $c_1, c_2, \ldots, c_\Lambda$ известно t_i - номер перекрестка, к которому в данный момент направляется c_i, а также v_i - скорость c_i.

Сбор данных:

- Общее количество автомобилей, направляющихся к i-ому перекрестку
- Количество автомобилей, ожидающих на перекрестке i
- Время ожидания автомобиля
- avg_i — среднее время ожидания автомобилей на i-ом перекрестке

Оптимизация

Задача:
Дана система из k *активных* перекрестков $I_1, I_2, \ldots, I_k$. Перекресток i будем называть активным если $avg_i > 0$.
Найти

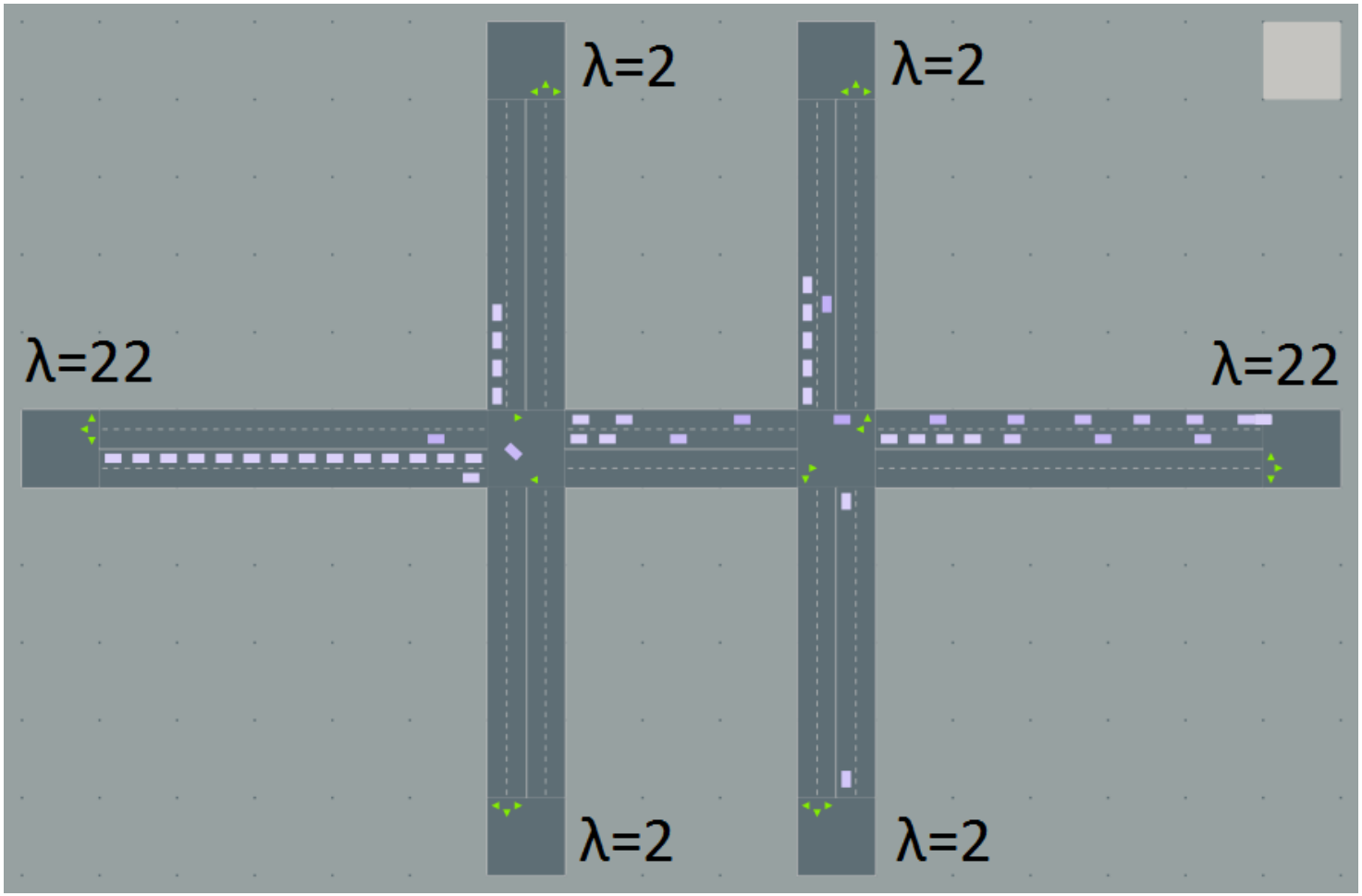
$$(t_{1,1}, t_{1,2}, t_{1,3}, t_{1,4}, \ldots t_{k,1}, t_{k,2}, t_{k,3}, t_{k,4}) \colon A = \frac{1}{k} \sum_{i=1}^{k} avg_i \to \min$$

$$\forall i = \overline{1,k}\ \forall j = \overline{1,4} \quad L \leqslant t_{i,j} \leqslant U,$$

где L, U - заранее известные ограничения.

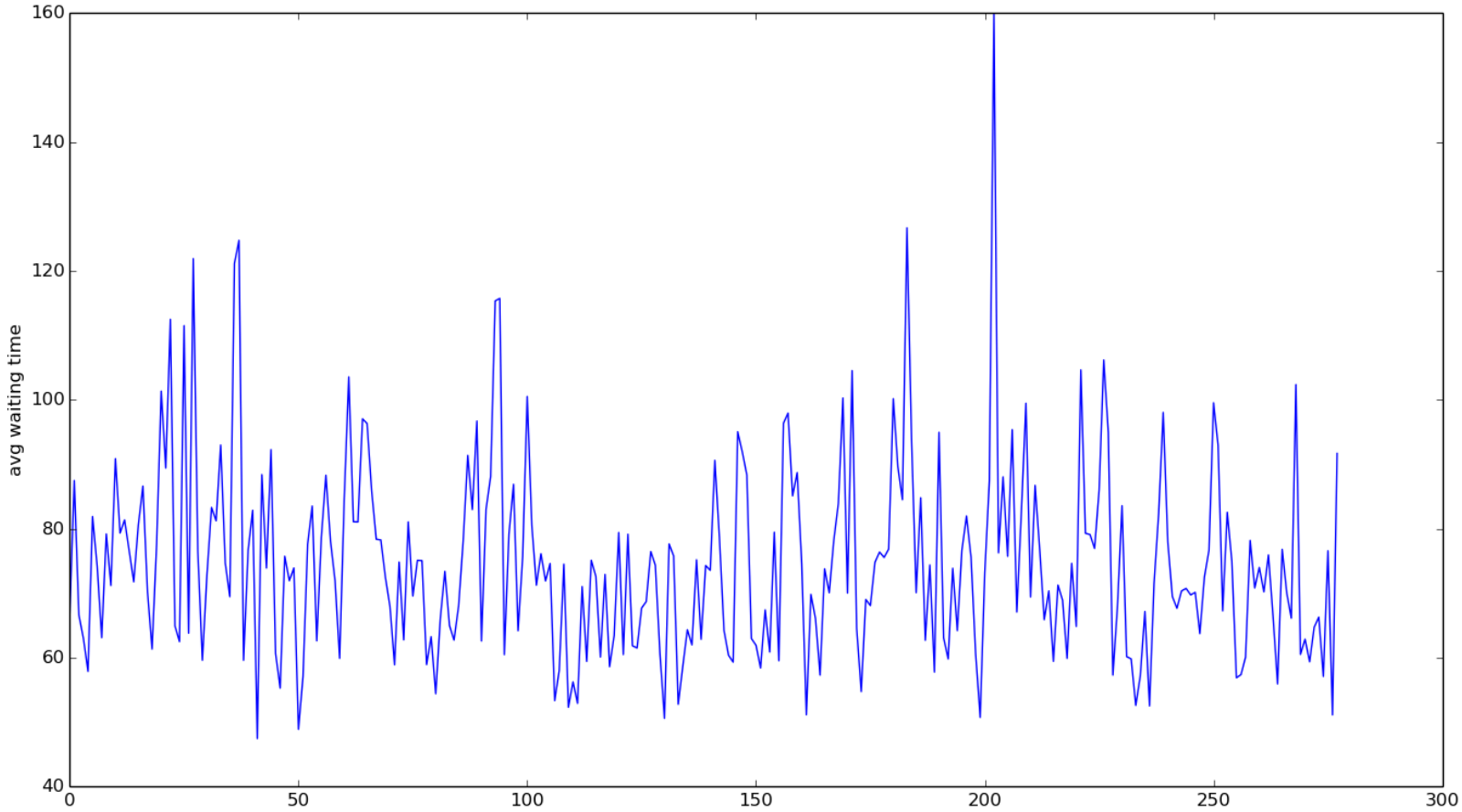
Пример 1

Без оптимизации: $A = 80.46$



Пример 1 (Оптимизация)

Simulated annealing: $A = 43.88$



Пример 1 (Оптимизация)

Genetic Algorithm:

Chromosome Encoding

Каждая хромосома это вектор

$$(t_{1,1}, t_{1,2}, t_{1,3}, t_{1,4}, \ldots t_{k,1}, t_{k,2}, t_{k,3}, t_{k,4})$$

Fitness Function

Fitness функция для определенной хромосомы зависит от величины A: чем меньше A при заданных параметрах, тем больше значение fitness.

Population size and Number of generations

Количество хромосом в популяции - 10. Количество поколений - 50.

Crossover operator Single Point Crossover. Rate - 0.8

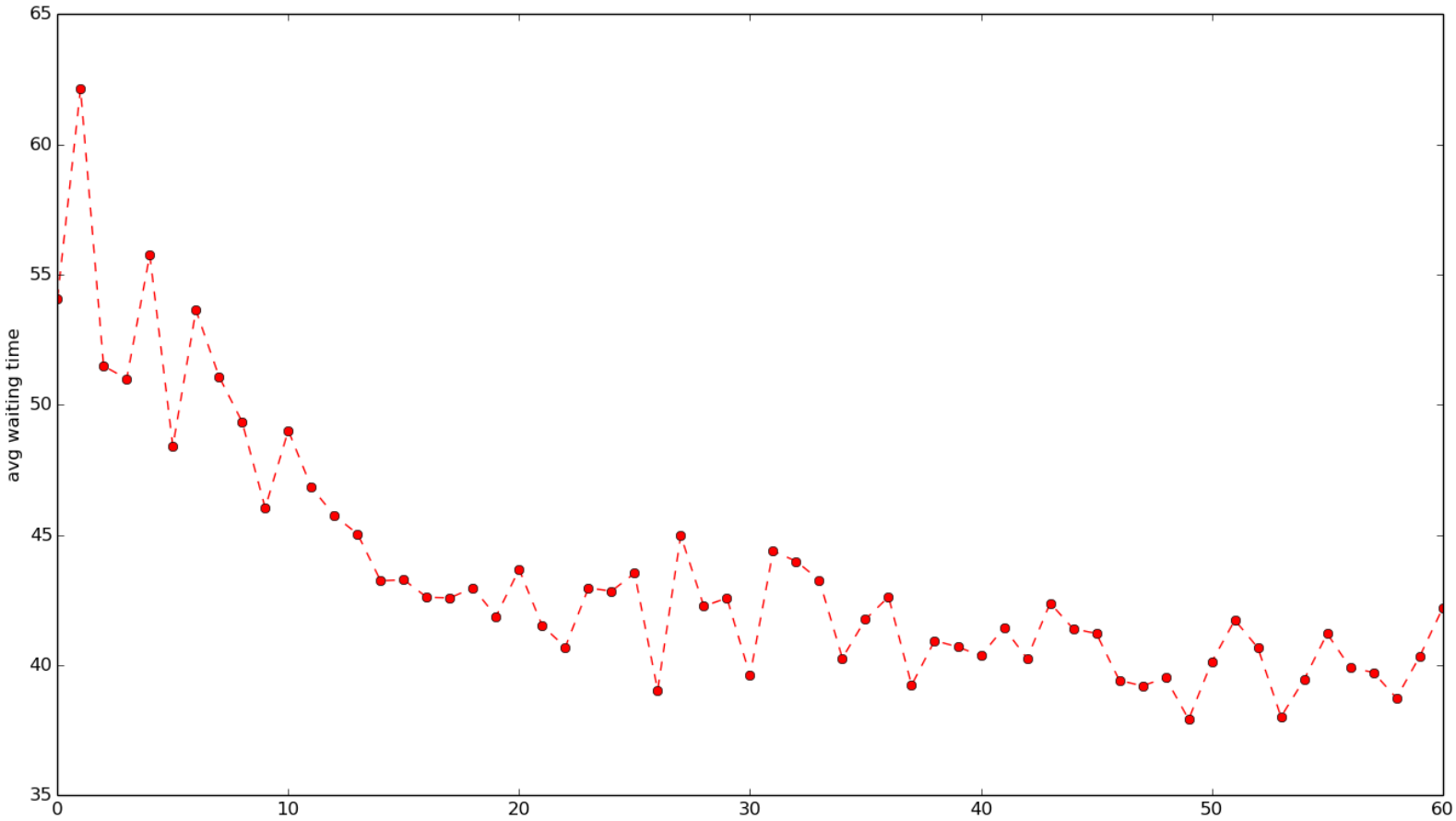
Mutation operator Gaussian. Rate - 0.02

Selection operator Roulette Wheel

Пример 1 (Оптимизация)

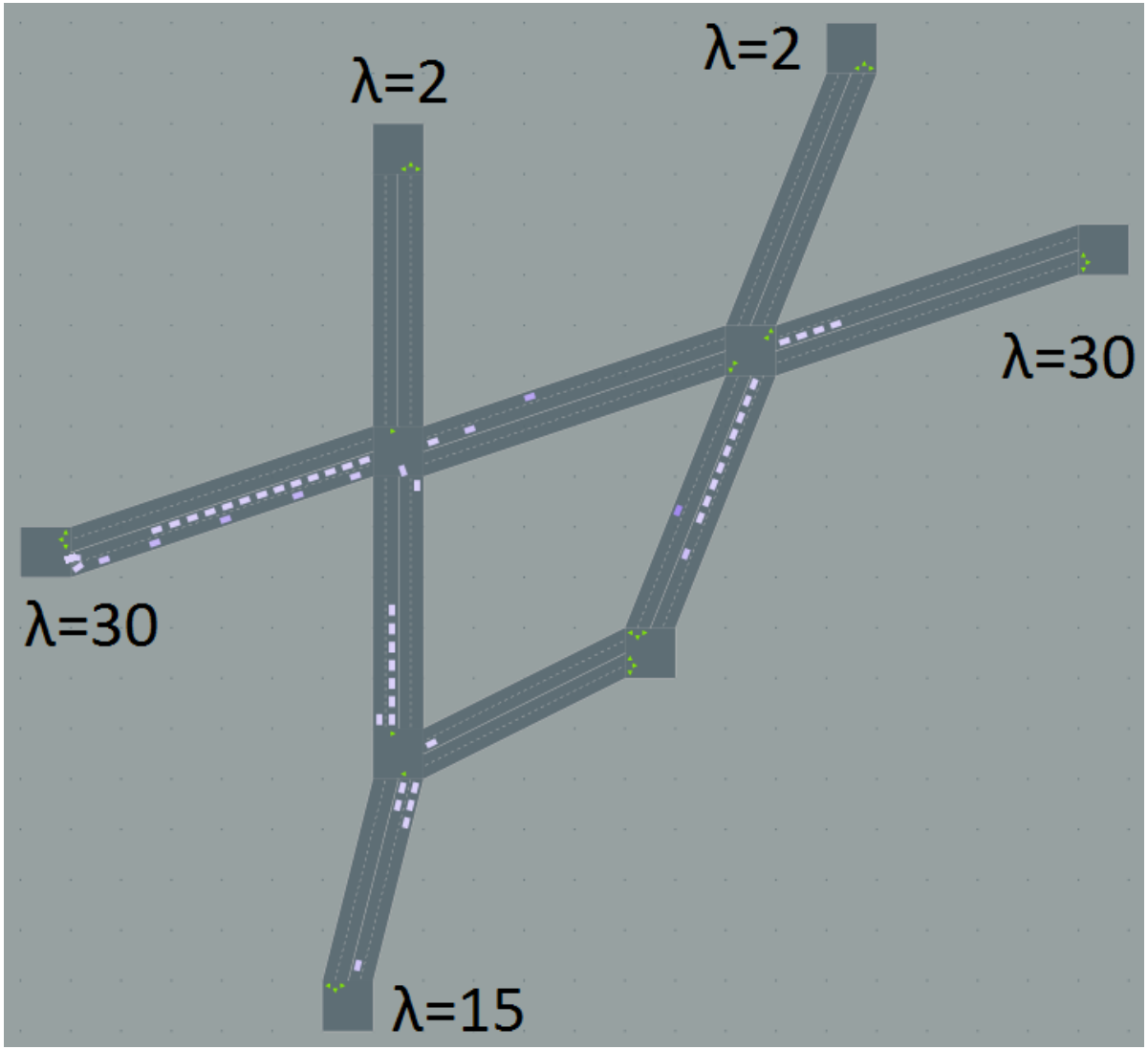
Genetic Algorithm: $A = 43.63$

Зависимость A от номера поколения



Пример 2

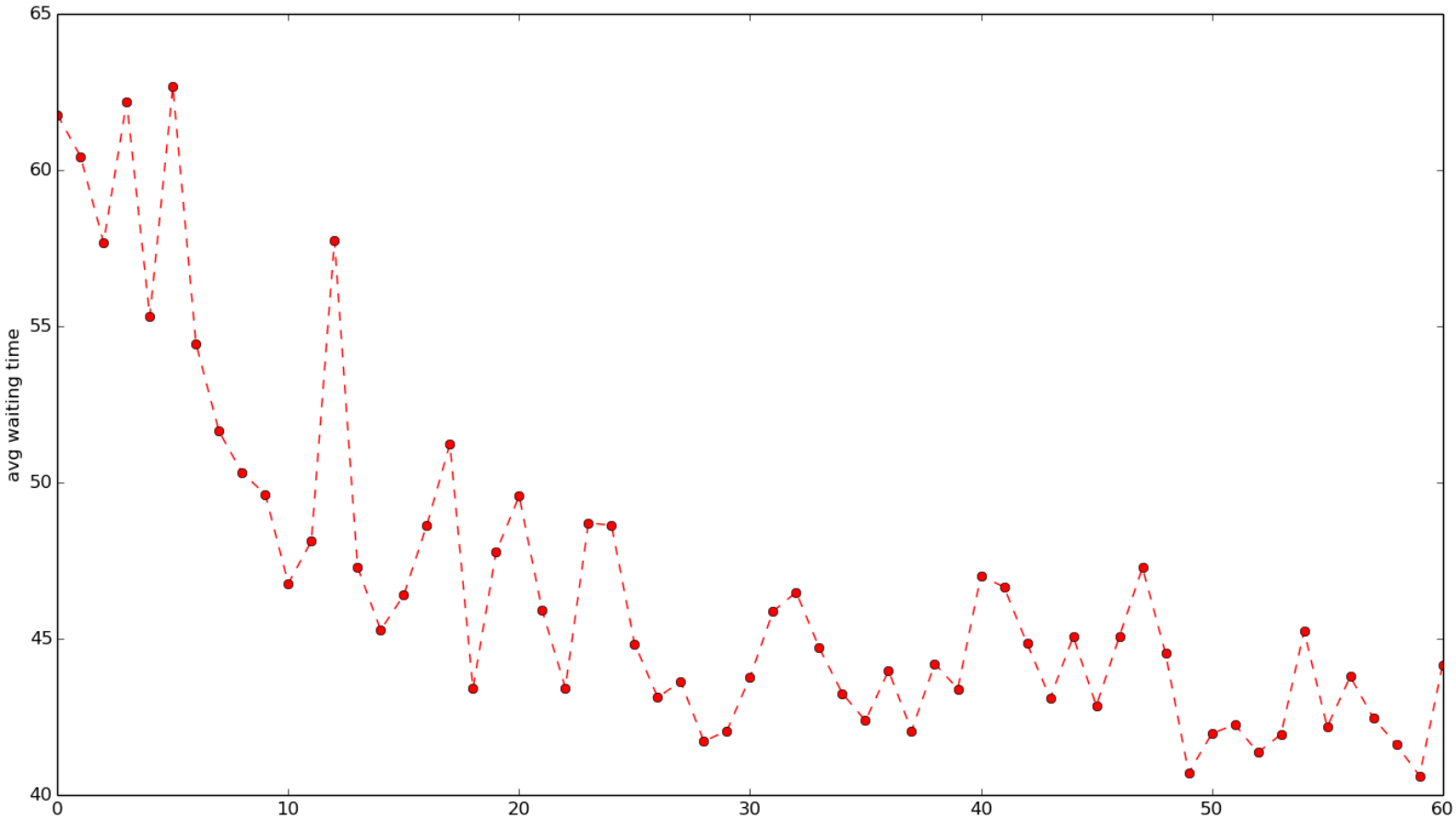
Без оптимизации: $A = 75.46$



Пример 2 (Оптимизация)

Genetic Algorithm: $A = 40.21$

Зависимость A от номера поколения



Результаты

- Для выполнения оптимизации были использованы библиотеки Pyevolve и Scipy.
- В качестве интерфейса взаимодействия между Python и CoffeeScript был выбран файловый ввод/вывод.

Технологии: JavaScript, CoffeeScript, Node.js, Python