КУРСОВОЙ ПРОЕКТ

Обучение глубокой нейронной сети управления автомобилем с использованием генетических алгоритмов и параллельных вычислений

2020, Цыникин Сергей (C++ developer and scientist)
https://github.com/znseday/Algo_NeuroCar_Coursework

Язык программирования и библиотеки

- Язык C++’17
- Компилятор MinGW 64bit
- Среда Qt Creator
- Фреймворк Qt 5.15
- Библиотеки STL, OpenGL

Цели проекта

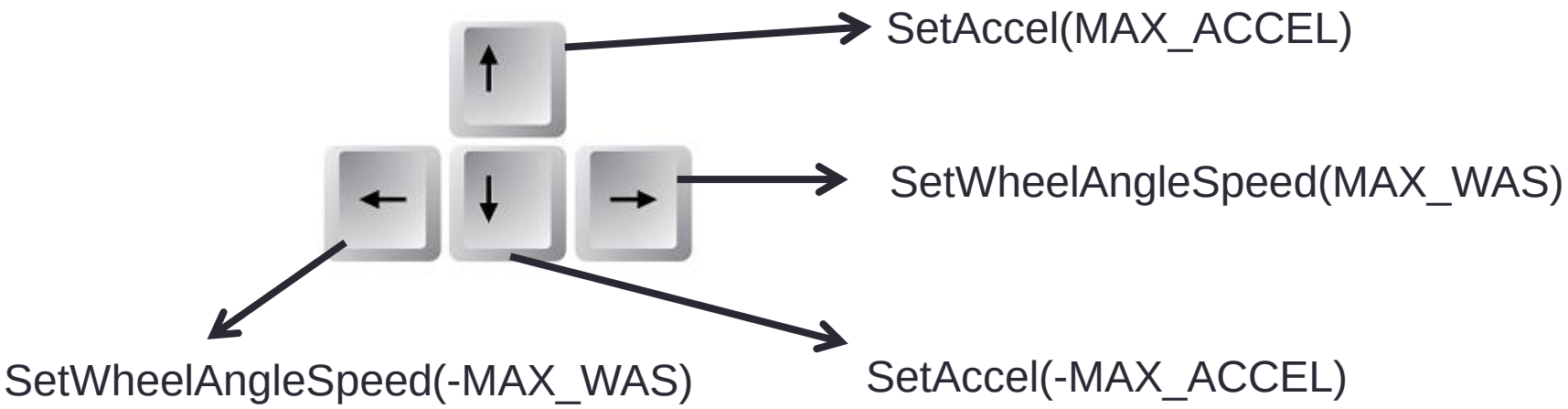
- Получить практический опыт разработки и настройки простой глубокой нейросети
- Получить практический опыт работы с генетическими алгоритмами, изучить их эффективность и применимость для обучения нейронных сетей
- Расширить опыт работы с фреймворком Qt и библиотекой OpenGL.
- Расширить опыт работы с многопоточностью

Постановка задачи

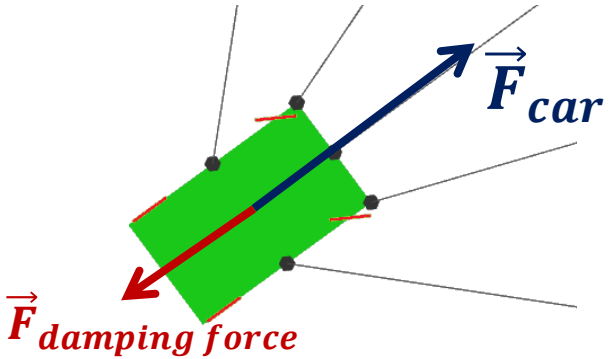
- Редактор карт (трасса и параметры машины)
- Возможность сохранения и загрузки карт (json)
- Режим ручного управления машиной
- Разработка нейронной сети и ее настройка
- Разработка генетического алгоритма и его настройка
- Возможность редактирования основных параметров
- Режим обучения с анимацией процесса
- Многопоточные вычисления (разработка пула потоков)
- Режим демонстрации и режим соревнования с ИИ
- Сбор статистики
- Сохранение и загрузка обученной нейронной сети (json)

Физическая модель

- Двухмерный мир, машины не сталкиваются
- Колеса поворачиваются плавно, поворот колеса требует времени, плавный автовозврат колес при движении, заносы
- Скорость рассчитывается в процессе движения и зависит от ускорения (accel), вязкого трения, и трения при поворотах. Скорость не может быть отрицательной
- Управление:



Физическая модель



- $\sum \vec{F} = m\vec{a}$

Рассмотрим движение вдоль прямой и положим $m = 1$

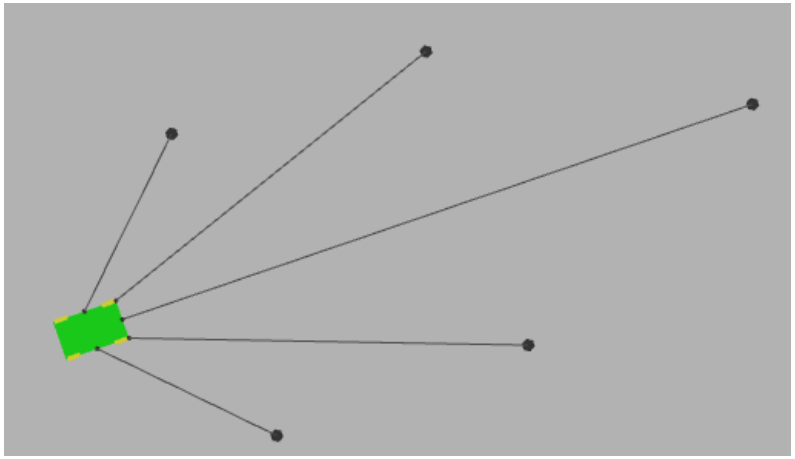
- $a_{eff} = F_{car} - F_{damping\ force}$
- $F_{damping\ force} = k_1 * |V_{car}| + k_2 * V^2{}_{car} + k_3 * |wheelAngle|$
- $V'_{car} = V_{car} + a_{eff} * dt$ (для краткости опущен учет поворотов)

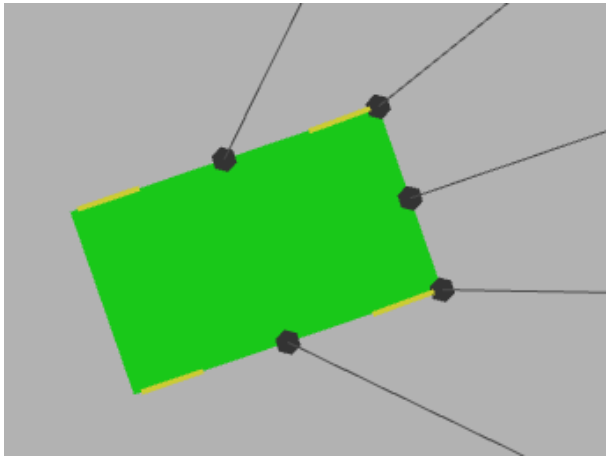
Поворот колес с автовозвратом при движении (эмпирическая ф-ла):

$$wheelAngle' = \frac{(wheelAngle + wheelAngleSpeed * k_3 * dt)}{(1 + |V_{car}| * k_4 * dt)^{\frac{3}{2}}}$$

Детекторы

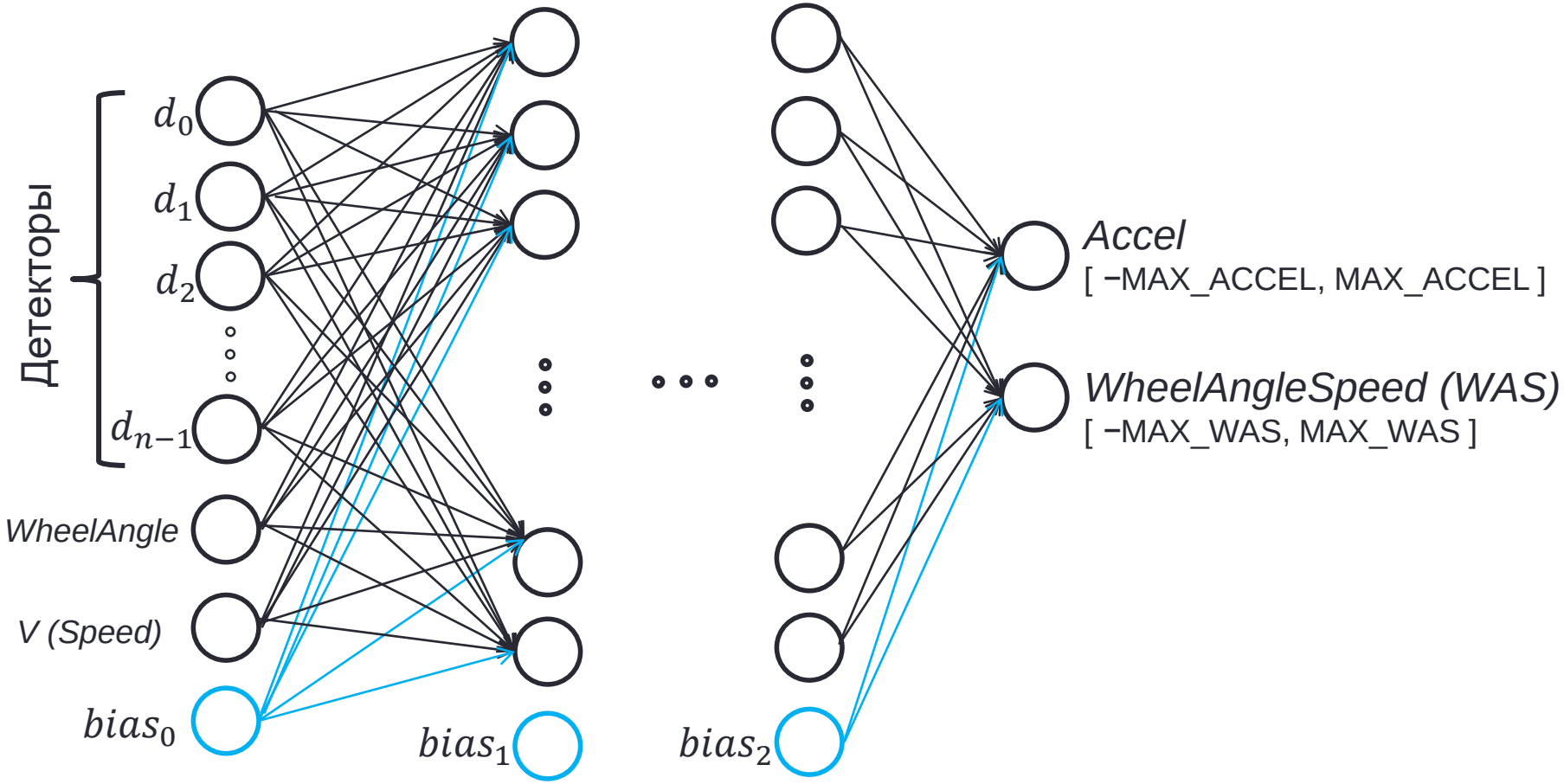
- Кол-во детекторов, длины лучей и углы ориентации задаются в настройках
- Расположение детекторов на автомобиле автоматически рассчитывается оптимальным образом исходя из их количества
- Значение на детекторе рассчитывается как корень из относительного расстояния до препятствия





Архитектура нейронной сети

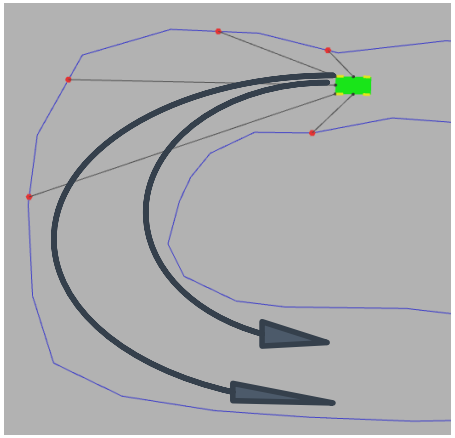
- Топология нейросети, функции активации нейронов и способ начальной инициализации весов задаются в настройках программы.



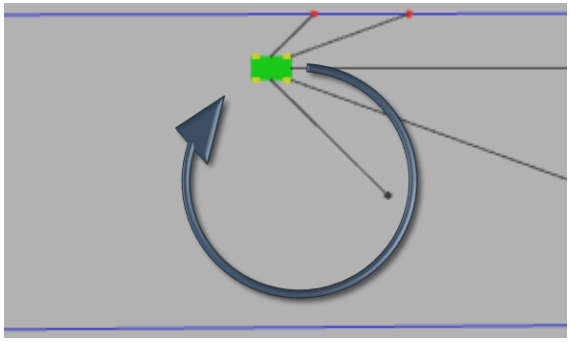
Что считать целевой функцией?

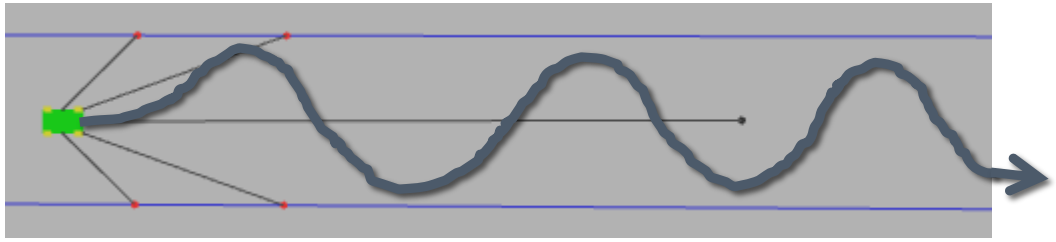
- Пройденный путь

  (Хотим, чтобы была пройдена вся трасса)



- Проблемы:
- 1. Зацикливание по кругу
- 2. Езда зигзагами
- 3. Возможно неоптимальная траектория (например, прохождение поворотов по большим радиусам вместо малых)





Что считать целевой функцией?

- Время жизни

  (Хотим, чтобы автомобиль жил как можно дольше без соударений о бордюры)

- Проблемы:
- 1. Машина может стоять на месте
- 2. Аккуратная, но очень медленная езда

  (“тише едешь – дальше будешь”)

Что считать целевой функцией?

- Средняя скорость

  (Хотим, чтобы ИИ проезжал трассу быстрее человека)

- Проблемы:

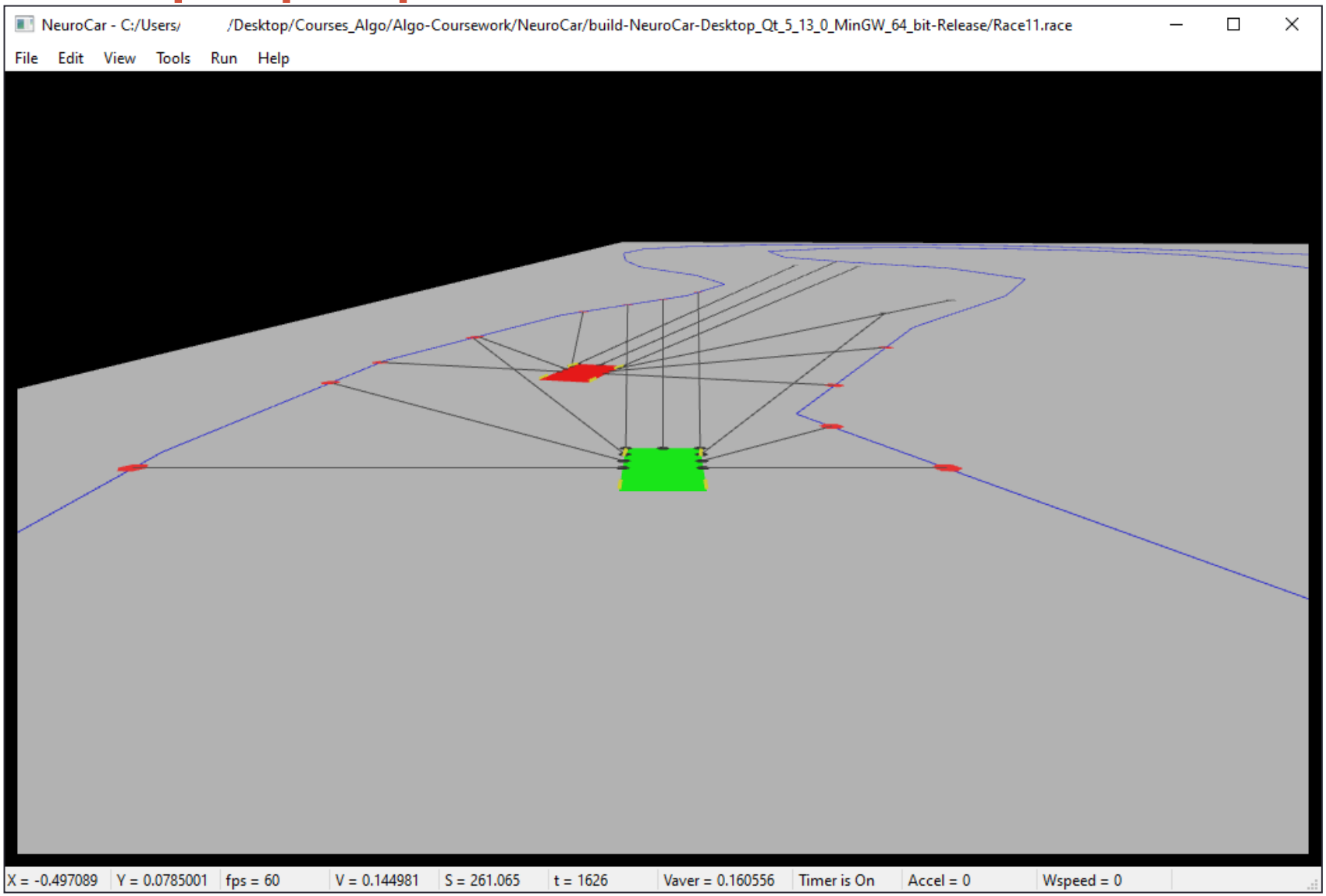
  1. В начале обучения машине выгоднее разогнаться и врезаться в ближайший поворот, чем притормозить и повернуть

Решение проблемы выбора

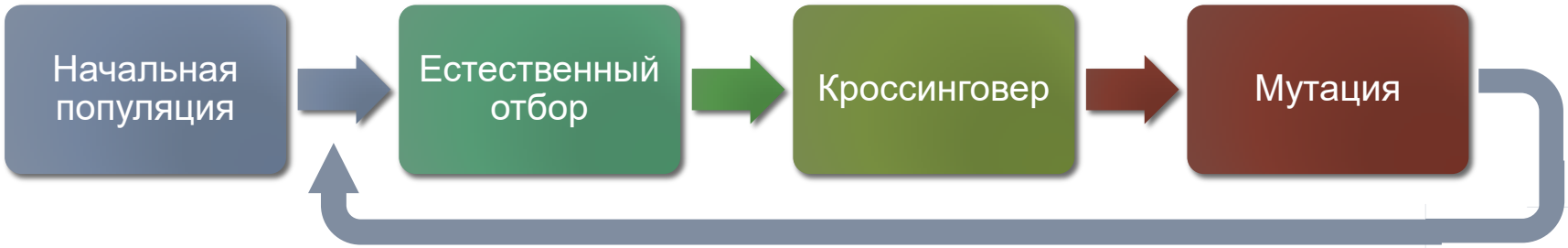
- Целевая функция – пройденный путь
- В начале трасса достаточно узкая, чтобы машина не развернулась
- Ограничение по минимальной скорости (скользящее среднее) – медленные машины отмирают
- Круговая трасса и ограничение на время жизни – как только сеть научится аккуратно проезжать круг, единственным критерием остановки станет максимальное время жизни. Тогда для всех успешных машин средняя скорость численно будет равна пройденному пути, и фактически начнется рост средней скорости (то, что мы и хотели)

- Другие возможные решения и дополнения (НЕ используем):
- 1. Обучение с учителем
- 2. Обозначение финиша и чекпоинты, средняя скорость по чекпоинтам (кол-во/мин)
- 3. Траектория, вдоль которой следует двигаться
- 4. Коридор внутри трассы, в котором нужно находиться
- 5. Прочее

Обзор программы

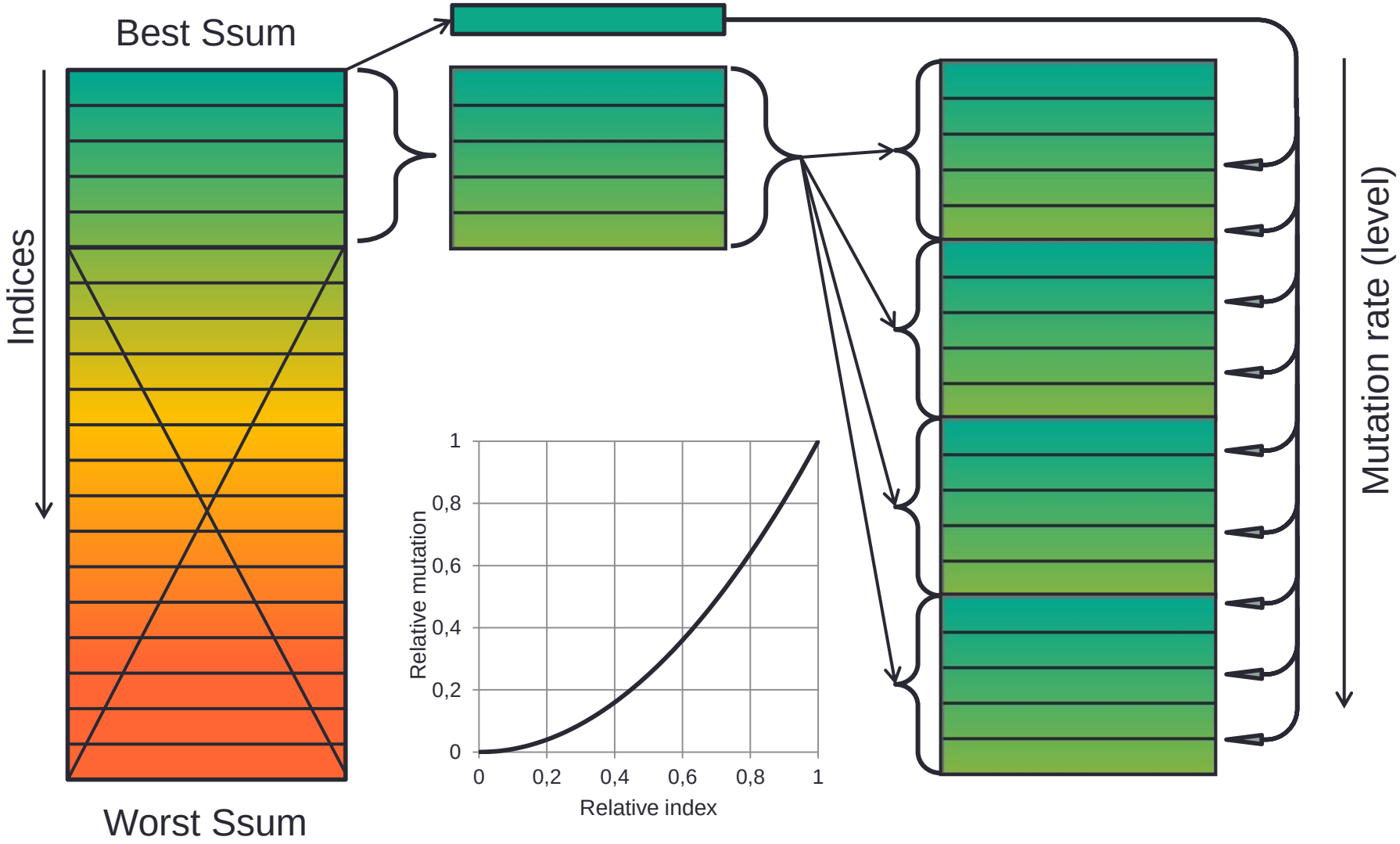


Генетический алгоритм

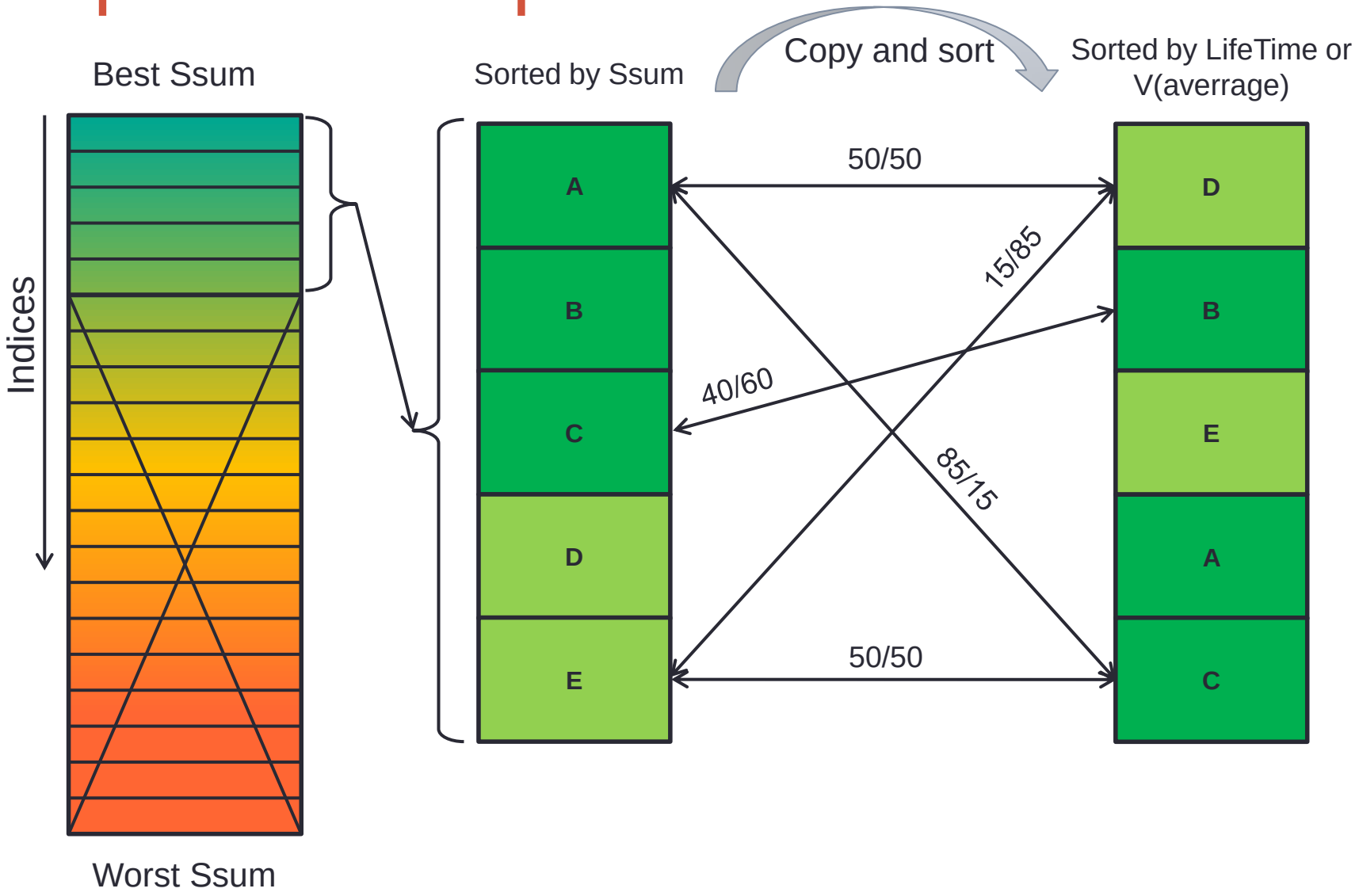


- Сортируем по пройденному пути
- Лидер никогда не умирает и не мутирует – “backup” лучшей хромосомы
- Кроссинговер (можно вкл/выкл)
- Разный уровень мутаций для разных сетей (настраиваемый параметр)

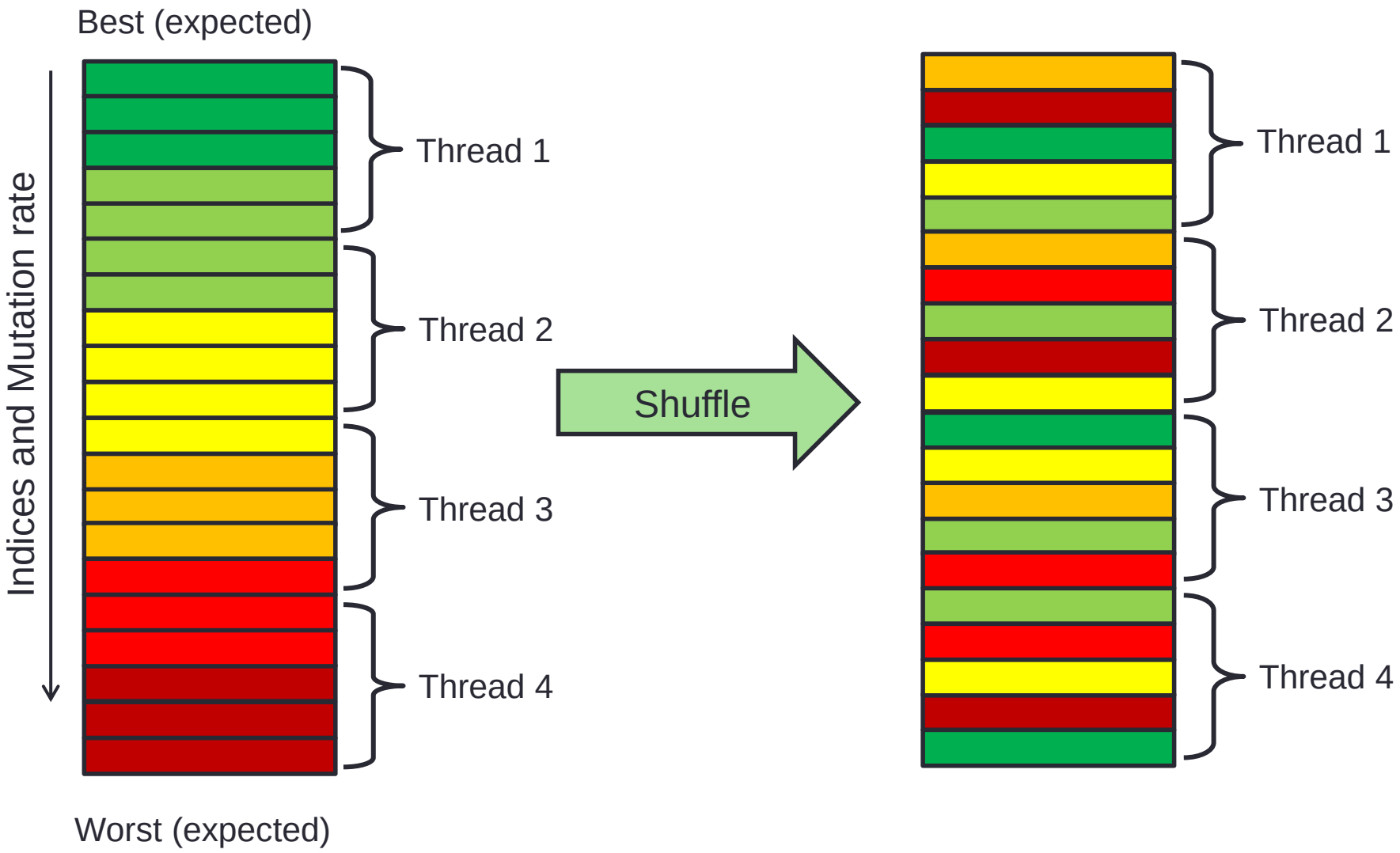
Естественный отбор



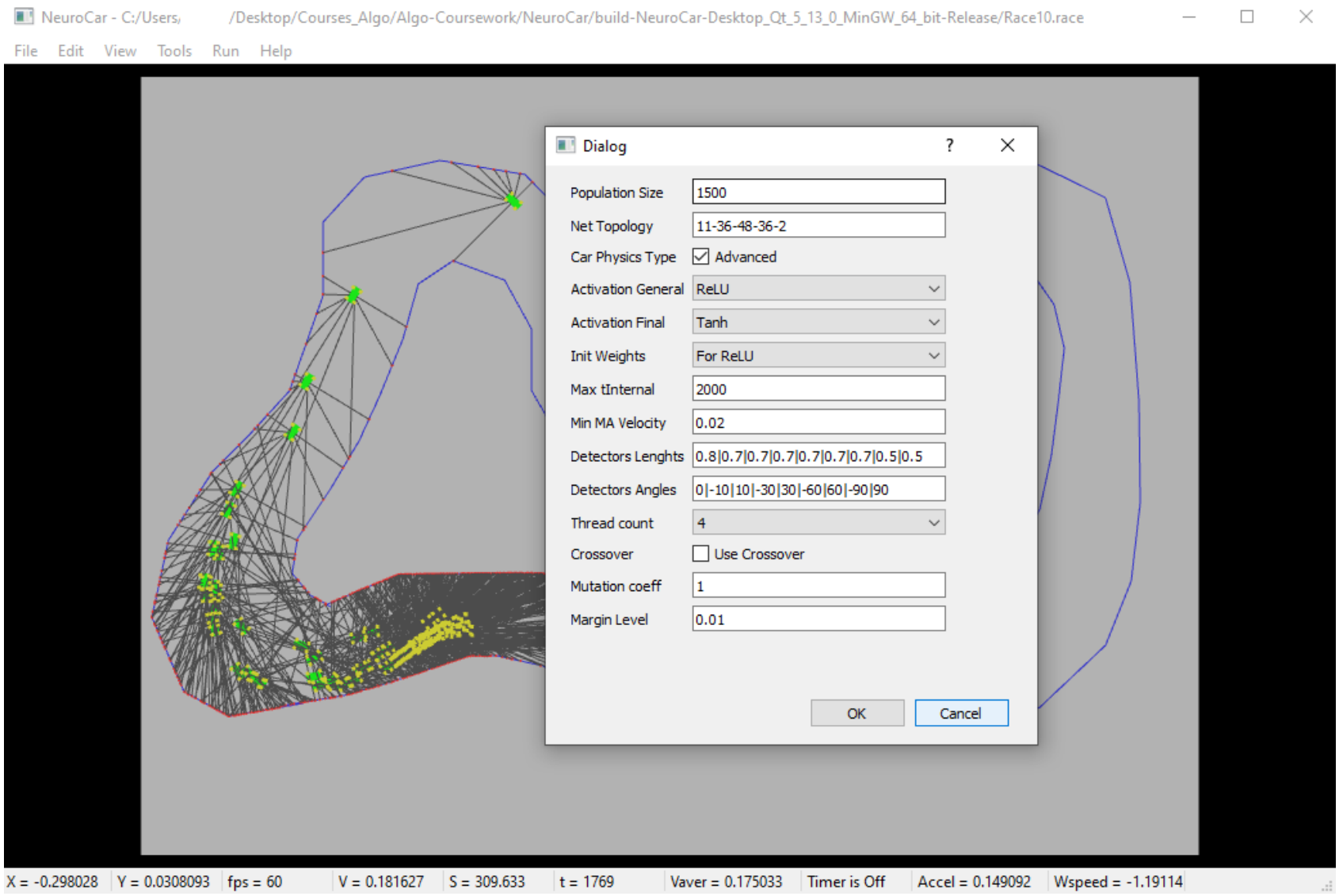
Кроссинговер



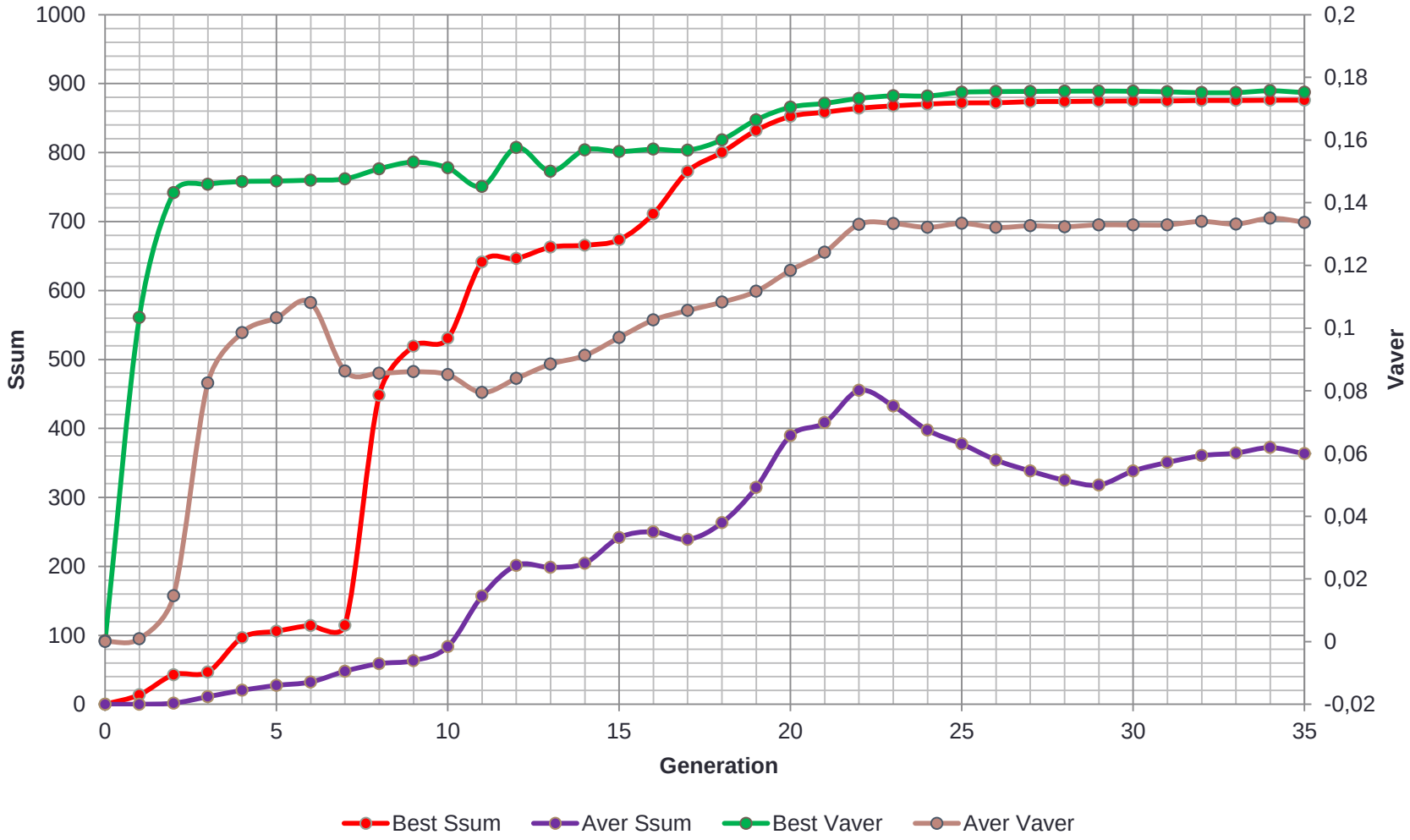
Оптимизация для многопоточности



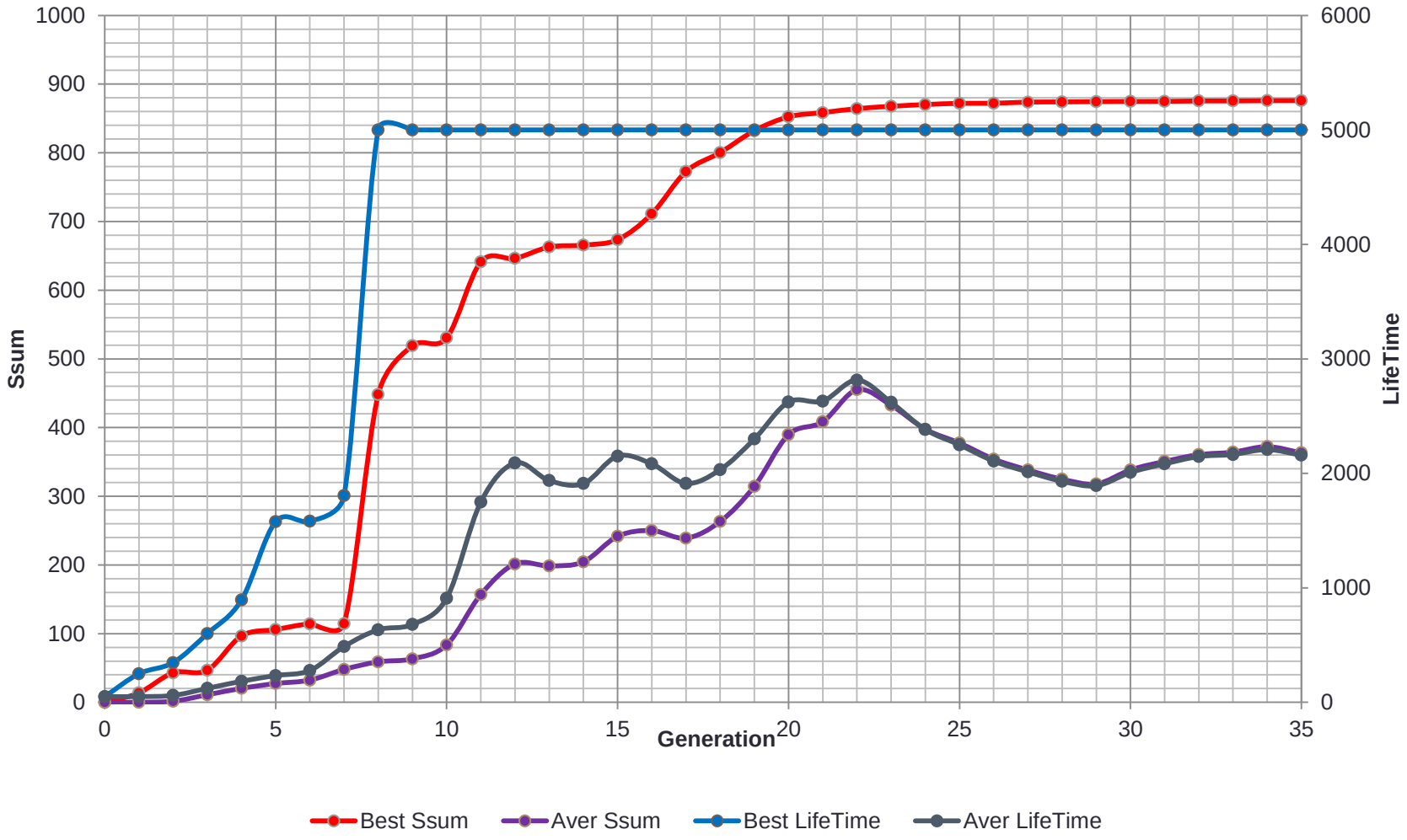
Настройки


NeuroCar - C:/Users/ /Desktop/Courses_Algo/Algo-Coursework/NeuroCar/build-NeuroCar-Desktop_Qt_5_13_0_MinGW_64_bit-Release/Race10.race
File Edit View Tools Run Help
Dialog
Population Size 1500
Net Topology 11-36-48-36-2
Car Physics Type Advanced
Activation General ReLU
Activation Final Tanh
Init Weights For ReLU
Max tInternal 2000
Min MA Velocity 0.02
Detectors Lenghts 0.8|0.7|0.7|0.7|0.7|0.7|0.7|0.5|0.5
Detectors Angles 0|-10|10|-30|30|-60|60|-90|90
Thread count 4
Crossover Use Crossover
Mutation coeff 1
Margin Level 0.01
OK
Cancel
X = -0.298028 Y = 0.0308093 fps = 60 V = 0.181627 S = 309.633 t = 1769 Vaver = 0.175033 Timer is Off Accel = 0.149092 Wspeed = -1.19114

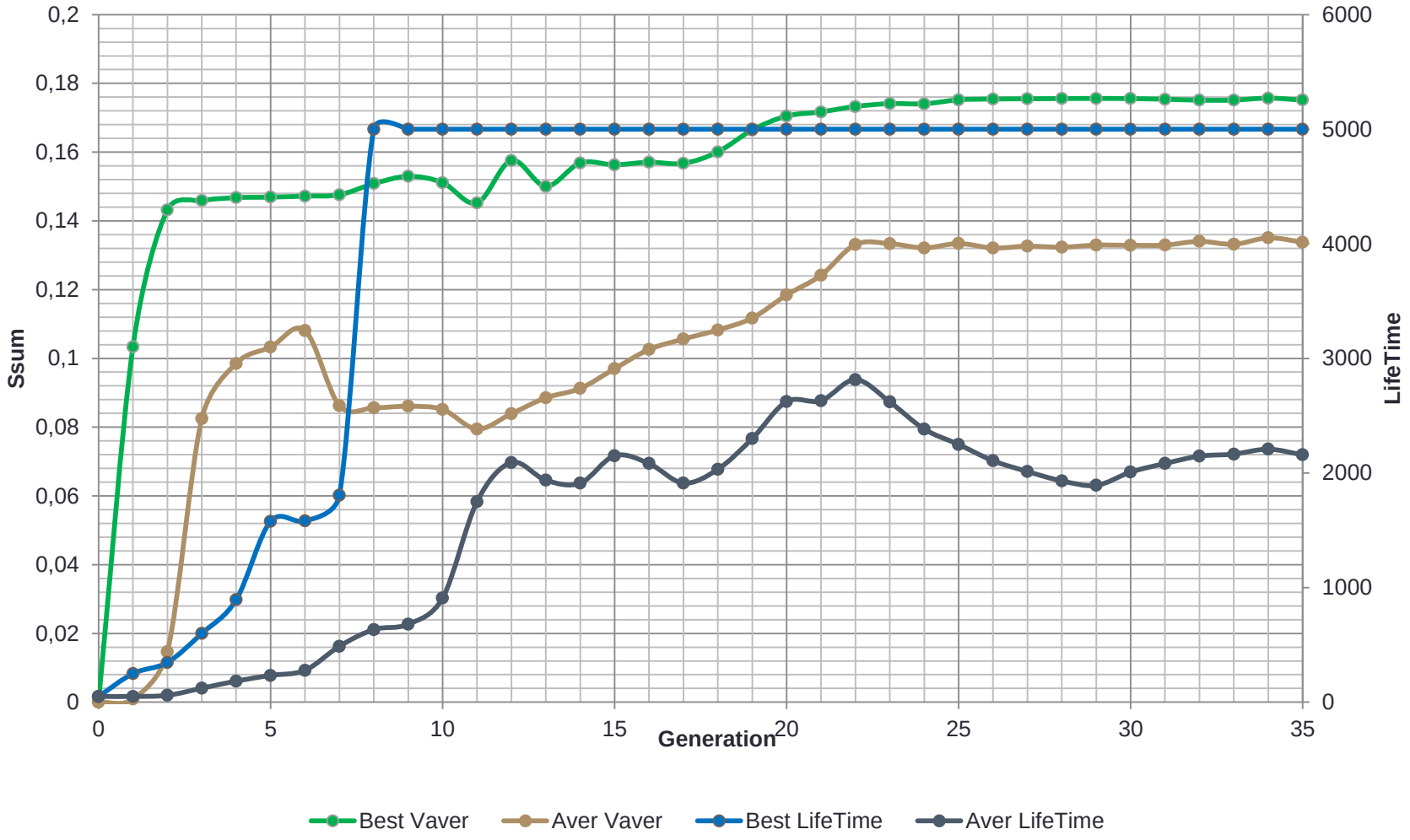
Статистика обучения



Статистика обучения



Статистика обучения



Борьба с переобучением

- Замечено, что на сложных трассах имеет место переобучение (оверфиттинг)
- Теоритически может помочь (еще не реализовано):
- Разброс стартовых точек с последующим кроссинговером
- Обучение на нескольких трассах с последующим кроссинговером
- Более удачные схемы мутация и кроссинговера
- “Интеллектуальные” схемы мутаций

Выводы

- Проект можно считать успешным
- Получены навыки работы с нейросетями, генетическими алгоритмами и пулом потоков
- Примененные модели и схемы генетических алгоритмов приводят к переобучению на сложных трассах, но дают хороший результат на простых трассах
- Пока не было выявлено значимой корреляции между топологией нейросети и результатами обучения, однако сети с 2-3 скрытыми слоями показывали более уверенный результат
- Проект открывает просторы для дальнейших экспериментов в данной области

Возможное развитие проекта

- Рефакторинг кода
- Улучшение UI
- Введение элементов игры (игровые очки, многопользовательская игра, поддержка игрового руля/педалей и т.п.)
- Учет столкновений машин между собой
- 3d мир, подъем в горку и спуск
- Применение дополнительных целевых функций (учет чекпоинтов и т.п.)
- Внедрение более продуманных и хитрых схем кроссинговера и мутаций
- Более подробное изучение влияния настраиваемых параметров на качество обучения нейросети

Спасибо за внимание!