



МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭКОНОМИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информатики и прикладной математики
Кафедра прикладной математики и экономико-математических методов

КУРСОВАЯ РАБОТА

по дисциплине:

«Системы компьютерной математики»

Тема: «Моделирование заблокированной полосы на автостраде»

Направление: 01.03.02 Прикладная математика и информатика

Студент: Бронников Егор

Группа: ПМ1901

Подпись: _____

Проверил: Фридман Григорий Морицович

Должность: д.т.н., профессор

Оценка: ОТЛИЧНО

Дата: 10.06.2020

Подпись: _____

Санкт-Петербург

2020

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ	3
1. МОДЕЛЬ НАГЕЛЯ-ШРЕКЕНБЕРГА.....	4
2. ДВУХПОЛОСНАЯ МОДЕЛЬ	5
3. ПРЕПЯТСТВИЕ НА ДОРОГЕ.....	9
4. РЕАЛИЗАЦИЯ МОДЕЛИ.....	10
4.1 Создание автомобилей.....	10
4.2 Реализация модели Нагеля – Шрекенберга	12
4.3 Смена полосы движения автомобилей.....	17
4.4 Обработка итераций модели и визуализация	20
5. ПЛАТФОРМЫ ДЛЯ АНАЛИЗА МОДЕЛИ.....	22
ЗАКЛЮЧЕНИЕ	24
СПИСОК ЛИТЕРАТУРЫ.....	25

ВВЕДЕНИЕ

Для того, чтобы отслеживать ситуацию на дороге удобно создать модель, которая будет описывать движение автомобилей на автостраде. Это может быть полезно, когда на участке дороги собираются производить ремонтные работы или когда происходит авария. Благодаря этому можно представить примерный поток, который будет создаваться в результате перекрытия одной из полос движения.

Задача: создать модель, которая будем имитировать движение автомобилей на двухполосной автостраде с заблокированной полосой движения и создать платформу для анализа модели при различных параметрах.

Существует классическая модель Нагеля-Шрекенберга, которая является дискретной моделью транспортных потоков, в которой улица с одной полосой движения разделена на идентичные ячейки, которые могут быть заняты автомобилем или нет. Так же это простая клеточная модель автомата для потока дорожного движения. Модель была развита в начале 1990-х годов немецкими физиками Каем Нагелем и Михаэлем Шрекенбергом.

В ходе работы эта модель была усовершенствована и была добавлена ещё одна полоса движения и также было добавлено препятствие для имитации происшествия на одной из полос движения.

1. МОДЕЛЬ НАГЕЛЯ-ШРЕКЕНБЕРГА

Скорость v_i каждого i автомобиля может быть представлена только целыми числами до определённой максимальной скорости v_{max} . Автомобили движутся по полосе движения в соответствии с 4 основными правилами:

1) ускорение:

Если автомобиль ещё не движется с максимальной скоростью, то его скорость увеличивается. (см. формулу 1)

$$if\ v_i < v_{max}\ then\ v_i = v_i + 1 \quad (1)$$

2) торможение:

Если существует опасность столкновения с автомобилем впереди, водитель тормозит, чтобы избежать аварии. Пусть x_i – это текущая позиция i автомобиля. Пусть gap будет расстоянием между автомобилем и автомобилем перед ним. (см. формулу 2 и 3)

$$gap = x_{car\ in\ front(i)} - x_i - 1 \quad (2)$$

$$if\ v_i > gap\ then\ v_i = gap \quad (3)$$

3) рандомизация:

С малой вероятностью p_{brake} , водитель дополнительно тормозит (см. формулу 4).

$$if\ rnd < p_{brake} \wedge v_i > 0\ then\ v_i = v_i - 1 \quad (4)$$

4) движение:

$$if\ x_0 > x_i + v_i \vee lane_0 \neq lane_i\ then\ x_i = x_i + v_i \quad (5)$$

Хоть и модель определяется только по этим правилам, она вполне может имитировать реальную динамику потока.

Эти и другие модели были изучены и расширены в последние несколько лет либо для того, чтобы исследовать особые свойства, такие как сложность критической области (области сужения), либо для того, чтобы вводить специальные части улицы, такие как выездная эстакада и изучать их влияние на поведение на дороге. [2]

2. ДВУХПОЛОСНАЯ МОДЕЛЬ

Поскольку большинство улиц имеют две полосы движения, по которым траектория движения либо в том же направлении, либо в противоположном направлении, и возможен обгон, вполне разумно модифицировать классическую модель, чтобы построить систему из двух полос. Была рассмотрена очень распространённую ситуацию, когда автомобили движутся в одном направлении по двухполосной проезжей части [1]. У них есть возможность сменить полосу движения, чтобы они могли обогнать автомобиль впереди, но при этом они должны соблюдать правила, чтобы придерживаться левой полосы движения (если не обгонять) и избегать обгона там. Как следствие, порядок автомобилей не остаётся постоянным, как в модели с одной полосой движения, что приводит к более сложной симуляции. Другая проблема состоит в том, что должен ли автомобиль менять полосу движения или нет, что делает необходимым ввести некоторые дополнительные правила смены полосы движения, применяемые до того, как места автомобилей будут определены их новыми скоростями.

Итак, так как в нашей модели препятствие будет всегда на правой полосе движения, то стоит установить правило, по которому автомобиль на правой полосе движения будет перестраиваться в левую полосу движения, а если автомобиль находится на левой полосе движения, то он уже не может перестроиться на соседнюю полосу. Тогда стоит рассмотреть несколько случаев.

1) Если существует машина, которая находится на левой полосе движения и идёт впереди – $frontLeftLine(i)$ и существует машина, которая находится также на левой полосе, но идёт позади – $behindLeftLine(i)$.

Наглядно изображение продемонстрировано на рисунке (см. рисунок 1).

Так же этот случай можно записать следующим образом (см. формулу 6):

$$if \exists frontLeftLine(i) \wedge \exists behindLeftLine(i) \quad (6)$$

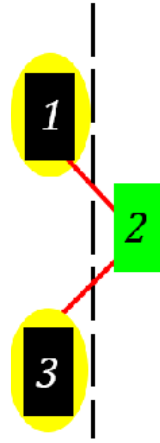


Рисунок 1 – Случай перестройки в левую полосу №1

$$if\ x_1 > x_2 > x_3 \wedge rnd < W_2\ then\ lane_2 = lane_2 - 1 \quad (7)$$

2) Если существует машина, которая находится на левой полосе движения и идёт впереди – $frontLeftLine(i)$, но при это отсутствует автомобиль позади – $behindLeftLine(i)$. Наглядно изображение продемонстрировано на рисунке (см. рисунок 2). Так же этот случай можно записать следующим образом (см. формулу 8):

$$if\ \exists\ frontLeftLine(i) \wedge \nexists\ behindLeftLine(i) \quad (8)$$

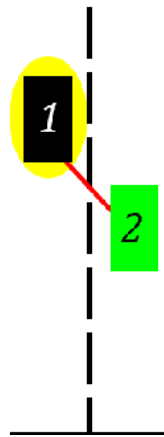


Рисунок 2 – Случай перестройки в левую полосу №2

$$if\ x_1 > x_2\ and\ rnd < W_2\ then\ lane_2 = lane_2 - 1 \quad (9)$$

3) Если отсутствует автомобиль, который находится на левой полосе движения и идёт впереди – $frontLeftLine(i)$, но при это существует автомобиль позади – $behindLeftLine(i)$. Наглядно изображение

продемонстрировано на рисунке (см. рисунок 3). Так же этот случай можно записать следующим образом (см. формулу 10):

$$if \nexists frontLeftLine(i) \wedge \exists behindLeftLine(i) \quad (10)$$

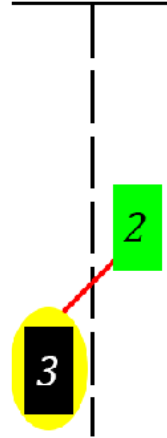


Рисунок 3 – Случай перестройки в левую полосу №3

$$if x_2 > x_3 \wedge rnd < W_2 then lane_2 = lane_2 - 1 \quad (11)$$

Вероятность W вытекает из следующих условий: независимо от скорости автомобиля впереди, водитель начинает обгон с вероятностью максимум W_{max} . Эта вероятность уменьшается, чем быстрее автомобиль на проезжей части, то есть, чем меньше разница между скоростями. При небольшой разнице в скорости по отношению к автомобилю впереди вероятность составляет: $0.1 \times difference$, где $difference$ – это разница между скоростью автомобиля и автомобиля впереди. Например, если автомобиль примерно на 2 единицы быстрее, чем автомобиль впереди, то $W = 0.1 \times 2 = 0.2$. Если разница больше $\frac{v_{max}}{c}$, то W остаётся равной W_{max} (так как это максимум). В результате водитель автомобиля остаётся с вероятностью не менее 0.2 на правой полосе движения.

$\forall \Delta v = v_{car\ in\ front(i)} - v_i then:$

$$W = \begin{cases} 0.1 \cdot \Delta v \cdot \Theta(\Delta v), & \Delta v \leq \frac{v_{max}}{c} \\ W_{max}, & \Delta v > \frac{v_{max}}{c} \end{cases} \quad (12)$$

с функцией Хевисайда:

$$\theta(x) = \begin{cases} 1, & \text{for } x \geq 0 \\ 0, & \text{for } x < 0 \end{cases} \quad (13)$$

или упрощённый вид:

$$W = \max \{0, \min \{W_{max}, 0.1 \cdot \Delta v\}\} \quad (14)$$

3. ПРЕПЯТСТВИЕ НА ДОРОГЕ

Ситуация, когда одна полоса шоссе закрыта из-за аварии или дорожных работ, известна каждому водителю автомобиля: на определённом расстоянии до места, где есть препятствие, дорожный знак предписывает водителям замедлиться и входить в общий поток движения с только одной полосой. При условии, что исследуется ситуацию, когда препятствие прерывает движение на правой полосе системы, таким образом улица должна быть разделена на три секции:

1) зона сужения:

Правило смены полосы движения изменилось на более строгое, чем раньше: теперь водитель на правой полосе вынужден переходить в соседнюю ячейку независимо от того, насколько быстро идут автомобили впереди и сзади.

2) зона с препятствием:

Здесь ситуацию довольно легко объяснить. Правая полоса закрыта, так что движение ограничено левой полосой.

3) двухполосная зона:

В остальной части системы, которая составляет самый длинный маршрут, сохраняются всё те же правила, которые были продемонстрированы выше. [1]

4. РЕАЛИЗАЦИЯ МОДЕЛИ

Модель была реализована на Wolfram Mathematica, версии 12.0. [3]

4.1 Создание автомобилей

Сначала стоит разобрать функцию – *carInitialization*, которая позволяет генерировать автомобили (см. рисунок 4).

```
Clear[carInitialization]
carInitialization[carCount_, maxVelocity_, laneCount_, probabilityOneTime_, W_] := Module[
{
cars = Range@carCount,
speed = RandomInteger[{1, maxVelocity}, carCount],
lane = RandomInteger[{0, laneCount - 1}, carCount],
time,
position = ConstantArray[0, carCount],
probabilityW = ConstantArray[W, carCount],
rndColor = RandomColor[carCount],
prop
},
time = Table[If[RandomReal[] < probabilityOneTime & i ≠ carCount & lane[[i + 1]] != lane[[i], i + 1, i], {i, carCount}];
prop = AssociationThread[{"Velocity", "Lane", "Time", "Position", "W", "Color"} → #] & /@ Transpose[{speed, lane, time, position, probabilityW, rndColor}];
AssociationThread[cars → prop]
]
```

Рисунок 4 – Функция *carInitialization*

1. На вход функция получает количество автомобилей, которые нужно сгенерировать – *carCount*, значение максимальной скорости – *maxVelocity*, количество линий – *laneCount*, вероятность для создания машин в один промежуток времени – *probabilityOneTime*, начальная вероятность *W* для всех машин. Так же были предусмотрены ограничения на головы аргументов и была применена функция *PatternTest*, которая позволит отслеживать их значения. (см. рисунок 5)

```
Clear[carInitialization]
carInitialization[carCount_Integer?Positive,
maxVelocity_Integer?Positive,
laneCount_Integer?Positive,
probabilityOneTime_Real?Positive | probabilityOneTime_Rational?Positive,
W_Real?Positive] :=
```

Рисунок 5 – Функция *carInitialization*, ограничения аргументов

2. Создаются локальные переменные:
 - 2.1 *cars* – номера автомобилей – список чисел от 1 до *carCount*.
 - 2.2 *speed* – скорости автомобилей – список случайных чисел от 1 до *maxVelocity* в количестве *carCount*.
 - 2.3 *lane* – линии появления автомобилей – список чисел от 0 до *laneCount* – 1 в количестве *carCount*.

2.4 *time* – промежутки времени появления автомобилей – список чисел, который будет заполняться по следующему правилу (см. формулу 15):

$$\begin{aligned} & \text{if } rnd < probabilityOneTime \wedge i \neq carCount \wedge lane_{i+1} \neq lane_i \\ & \quad \text{then } i + 1 \text{ else } i \quad , \forall i \in [1, carCount] \end{aligned} \quad (15)$$

2.5 *position* – позиции автомобилей – список нулей в количестве *carCount*.

2.6 *probavilityW* – вероятности *W* автомобилей – список вероятности *W* в количестве *carCount*.

2.7 *rndColor* – цвета автомобилей – список случайных цветов в количестве *carCount*.

2.8 *prop* – объединение всех вышеперечисленных свойств автомобилей в виде ассоциации.

В итоге после результата выполнения функции *carInitialization*, на выходе получается ассоциация, в правых частях которой ассоциация (см. рисунок 6).






```
carInitialization[5, 5, 2, 0.4, 0.8] (* countCars → 5, vmax → 5, laneCount → 2, probabilityOneLane → 0.4, W → 0.8*)
<|1 → <|Velocity → 4, Lane → 1, Time → 1, Position → 0, W → 0.8, Color → |>,
2 → <|Velocity → 5, Lane → 1, Time → 2, Position → 0, W → 0.8, Color → |>,
3 → <|Velocity → 1, Lane → 1, Time → 3, Position → 0, W → 0.8, Color → |>,
4 → <|Velocity → 2, Lane → 0, Time → 4, Position → 0, W → 0.8, Color → |>,
5 → <|Velocity → 2, Lane → 0, Time → 5, Position → 0, W → 0.8, Color → |>
```

Рисунок 6 – Пример работы функции *carInitialization*

Так же стоит рассмотреть функцию – *addObstacle*, которая будет добавлять препятствие на дорогу и тем самым, в дальнейшем, будет специальным образом обрабатываться.

```
Clear[addObstacle]
addObstacle[cars_Association, {posStart_Integer?Positive, posFinish_Integer?Positive}] /; posStart ≤ posFinish :=
<|0 → <| "Velocity" → 0, "Lane" → 1, "Time" → 0, "Position" → {posStart, posFinish}, "W" → 0, "Color" → Red |> |> - Join - cars
```

Рисунок 7 – Функция *addObstacle* для добавления препятствия

1. На вход функция получает ассоциацию – *cars* и список из двух элементов в котором на первом месте находится начальная позиция препятствия - *posStart*, а на второй позиции – конечная позиция – *posFinish*. Так же стоит

Condition на то, чтобы начальная позиция (*posStart*) не превосходила конечную позицию (*posFinish*).

2. С помощью функции *Join* добавляется, к исходной ассоциации *cars*, ещё один элемент с идентификатором – 0, скоростью – 0 («Velocity» → 0), которая находится на правой полосе движения («Lane» → 1), с временем появления на дороге – 0 («Time» → 0), позицией, которая находится на промежутке от *posStart* до *posFinish* («Position» → {*posStart*, *posFinish*}), вероятность *W* – 0 («W» → 0) и красным цветом («Color» → *Red*).

Результат выполнения функции *addObstacle* (см. рисунок 8).







```
addObstacle[cars, {15, 30}] (* cars → cars, positionObstacle → {start → 15, finish → 30} *)
<|0 → <|Velocity → 0, Lane → 1, Time → 0, Position → {15, 30}, W → 0, Color → |>,
1 → <|Velocity → 4, Lane → 1, Time → 1, Position → 0, W → 0.8, Color → |>,
2 → <|Velocity → 5, Lane → 1, Time → 2, Position → 0, W → 0.8, Color → |>,
3 → <|Velocity → 1, Lane → 1, Time → 3, Position → 0, W → 0.8, Color → |>,
4 → <|Velocity → 2, Lane → 0, Time → 4, Position → 0, W → 0.8, Color → |>,
5 → <|Velocity → 2, Lane → 0, Time → 5, Position → 0, W → 0.8, Color → |>|>
```

Рисунок 8 – Пример работы функции *addObstacle*

4.2 Реализация модели Нагеля – Шрекенберга

Теперь перейдём к основным правилам, которые были сформулированы выше, когда мы рассматривали модель Нагеля – Шрекенберга.

1) ускорение:

Была создана функция *acceleration*, которая моделирует правило ускорения (см. формулу 1).

```
Clear[acceleration]
acceleration[carId_Integer?NonNegative, maxVelocity_Integer?Positive][cars_Association] := Module[
{
  cAssociation = cars
},
If[
  cAssociation[carId]["Velocity"] < maxVelocity ∧ carId ≠ 0,
  cAssociation[carId]["Velocity"] += 1
];
cAssociation
]
```

Рисунок 9 – Функция *acceleration*

1. На вход функция получает ассоциацию – *cars*, идентификатор текущей машины – *carId*, значение максимальной скорости – *maxVelocity*.

2. Далее в соответствии с формулой №1 происходит преобразование и также мы обрабатываем случай, когда на вход подаётся объект (препятствие) с идентификатором – 0.

По окончании работы функции мы получаем следующий результат (см. рисунок 10).

```
showExample[carsDataSample,  
acceleration[1, 5]]
```

Входные данные:

	Velocity	Lane	Time	Position	W	Color
0	0	1	0	{15, 30}	0	■
1	3	0	1	13	0.8	■
2	4	1	2	11	0.8	■
3	3	1	3	7	0.2	■

Результат преобразования:

	Velocity	Lane	Time	Position	W	Color
0	0	1	0	{15, 30}	0	■
1	4	0	1	13	0.8	■
2	4	1	2	11	0.8	■
3	3	1	3	7	0.2	■

Рисунок 10 – Пример работы функции *acceleration*

Также такую структуру удобно использовать в связке с функцией *Dataset*, для более наглядного представления. На рисунке №10 я воспользовался функцией *showExample*, которая просто формирует *Dataset* и делает колонку из этих объектов. Как можно видеть, у автомобиля с идентификатором 1 изменилась скорость («Velocity» → 4).

2) торможение:

Создана функция *barking*, которая моделирует правило торможения (см. формулы 2 и 3).

```
Clear[barking]
barking[carId_Integer?NonNegative][cars_Association] := Module[
{
  cAssociation = cars,
  forwardCar = frontCar[cars, carId],
  gap
},
gap = cAssociation[forwardCar]["Position"] - cAssociation[carId]["Position"] - 1;
If[forwardCar != Null & cAssociation[carId]["Velocity"] > gap & carId != 0, |
  cAssociation[carId]["Velocity"] = gap];
cAssociation
]
```





Рисунок 11 – Функция *barking*

1. На вход функция получает ассоциацию – *cars*, идентификатор текущей машины – *carId*.
2. Была создана функция *frontCar*, которая по ассоциации – *cars* и идентификатору автомобиля *carId* определяет идентификатор машины, которая находится на одной линии с текущим автомобилем и едет впереди него.
3. Далее в соответствии с формулами №2 и №3 происходит преобразование и также мы обрабатываем случай, когда на вход подаётся объект (препятствие) с идентификатором – 0.

По окончании работы функции мы получаем следующий результат (см. рисунок 12).

```
showExample[carsDataSample,  
barking[3]]
```

Входные данные:

	Velocity	Lane	Time	Position	W	Color
0	0	1	0	{15, 30}	0	
1	3	0	1	13	0.8	
2	4	1	2	11	0.8	
3	3	1	3	8	0.2	

Результат преобразования:





	Velocity	Lane	Time	Position	W	Color
0	0	1	0	{15, 30}	0	
1	3	0	1	13	0.8	
2	4	1	2	11	0.8	
3	2	1	3	8	0.2	

Рисунок 12 – Пример работы функции *barking*

Как можно видеть на рисунке №12, у автомобиля с идентификатором 3 изменилась скорость («Velocity» → 2).

- 3) рандомизация:

Создана функция *randomization*, которая моделирует правило рандомизации (см. формулу 4).

```

Clear[randomization]
randomization[carId_, pBreak_, rnd_] := Module[
{
    cAssociation=cars
},
If[
rnd<pBreak∧cAssociation[carId]["Velocity"]>0∧cAssociation[carId]["Position"]>0∧carId≠0,
cAssociation[carId]["Velocity"]-=1
];
cAssociation
]

```

Рисунок 13 – функция *randomization*

1. На вход функция получает ассоциацию – *cars*, идентификатор текущей машины – *carId*, вероятность дополнительного торможение – p_{brake} и случайное вещественное число – *rnd*. Так же были предусмотрены ограничения на головы аргументов и была применена функция *PatternTest*, которая позволит отслеживать их значения.

```

Clear[randomization]
randomization[carId_Integer?NonNegative,
pBreak_Rational?Positive|pBreak_Real?Positive,
rnd_Real?Positive][cars_Association]:=

```

Рисунок 14 – Функция *randomization*, ограничения аргументов

2. Далее в соответствии с формулой №4 происходит преобразование и также мы обрабатываем случай, когда на вход подаётся объект (препятствие) с идентификатором – 0.





По окончанию работы функции мы получаем следующий результат:

```

showExample[carsDataSample,
randomization[2, 1/3, 0.2]]

```

Входные данные:

	Velocity	Lane	Time	Position	W	Color
0	0	1	0	{15, 30}	0	
1	3	0	1	13	0.8	
2	4	1	2	11	0.8	
3	3	1	3	8	0.2	

Результат преобразования:





	Velocity	Lane	Time	Position	W	Color
0	0	1	0	{15, 30}	0	
1	3	0	1	13	0.8	
2	3	1	2	11	0.8	
3	3	1	3	8	0.2	

Рисунок 15 – Пример работы функции *randomization*

Как можно видеть на рисунке №15, у автомобиля с идентификатором 2 изменилась скорость («Velocity» → 3).

4) движение:

Создана функция *driving*, которая моделирует правило движения (см. формулу 5).

```
Clear[driving]
driving[carId_Integer?NonNegative][cars_Association]:=Module[
{
  cAssociation=cars,
  obstacle=cars[0],
  currentCar=cars[carId]
},
If[obstacle["Position"][[1]]>currentCar["Position"]+currentCar["Velocity"]Vobstacle["Lane"]#currentCar["Lane"],
cAssociation[carId]["Position"]+=currentCar["Velocity"]
];
cAssociation
]
```

Рисунок 16 – Функция *driving*





1. На вход функция получает ассоциацию – *cars* и идентификатор текущей машины – *carId*.

2. Далее в соответствии с формулой №5 происходит преобразование.

По окончании работы функции мы получаем следующий результат:

```
showExample[carsDataSample,
driving[1]]
```

Входные данные:

	Velocity	Lane	Time	Position	W	Color
0	0	1	0	{15, 30}	0	
1	3	0	1	13	0.8	
2	4	1	2	11	0.8	
3	3	1	3	8	0.2	

Результат преобразования:





	Velocity	Lane	Time	Position	W	Color
0	0	1	0	{15, 30}	0	
1	3	0	1	16	0.8	
2	4	1	2	11	0.8	
3	3	1	3	8	0.2	

Рисунок 17 – Пример работы функции *driving*

Как можно видеть на рисунке №15, у автомобиля с идентификатором 1 изменилась позиция («Position» → 16).

4.3 Смена полосы движения автомобилей

Данную часть реализации стоит начать с обработки вероятности W . Была создана функция – *probabilityW*, которая служит обработчиком вероятности W для автомобилей (см. рисунок 18).

```
Clear[probabilityW]
probabilityW[carId_,maxVelocity_,coefficientW_,maxW_][cars_]:=Module[
{
cAssociation=cars,
forwardCar=frontCar[cars,carId],
obstacle=cars[0],
differenceVelocity
},
If[forwardCar!=Null,
differenceVelocity=cAssociation[forwardCar]["Velocity"]-cAssociation[carId]["Velocity"];(* ΔV = Vcar in front(i)-Vi *)
If[differenceVelocity>maxVelocity/coefficientW,
cAssociation[carId]["W"]=0.1 differenceVelocity UnitStep[differenceVelocity],
cAssociation[carId]["W"]=maxW
]
];
If[obstacle["Position"][[1]]<cAssociation[carId]["Position"]+cAssociation[carId]["Velocity"]*obstacle["Lane"]==cAssociation[carId]["Lane"],
cAssociation[carId]["W"]=maxW
];
cAssociation
]
```

Рисунок 18 – Функция *probabilityW*

1. На вход функция получает ассоциацию – *cars*, идентификатор текущей машины – *carId*, значение максимальной скорости – *maxVelocity*, коэффициент c – *coefficientW* и максимально возможная вероятность W – *maxW*.

2. Тут также была использована функцией *frontCar*, для того, чтобы найти идентификатор впереди идущего автомобиля.





3. Далее в соответствии с формулами № 12-14 происходит преобразование.

4. Так же в этой функции есть обработка случая, когда машина приближается к препятствию и у неё нет другого выхода кроме как перестроится на левую полосу движения. В таком случае данный автомобиль получает максимально возможный параметр W – *maxW*.

По окончании работы функции мы получаем следующий результат (см. рисунок 19).

```
showExample[carsDataSample,  
probabilityW[3, 5, 3, 0.8]]
```

Входные данные:

	Velocity	Lane	Time	Position	W	Color
0	0	1	0	{15, 30}	0	
1	3	0	1	13	0.8	
2	4	1	2	11	0.8	
3	3	1	3	8	0.2	

Результат преобразования:





	Velocity	Lane	Time	Position	W	Color
0	0	1	0	{15, 30}	0	
1	3	0	1	13	0.8	
2	4	1	2	11	0.8	
3	3	1	3	8	0.1	

Рисунок 19 – Пример работы функции *probabilityW*

Как можно видеть на рисунке №19, у автомобиля с идентификатором 3 изменилась вероятность W («W» \rightarrow 0.1).

Теперь можно перейти к функции, которая отвечает за смену полосы движения автомобиля – *changeLaneToLeft* (см. рисунок 20).

```
Clear[changeLaneToLeft]
changeLaneToLeft[carId_Integer?NonNegative, rnd_Real][cars_Association]:=Module[
{
cAssociation=cars,
leftCarBehind=leftLaneCarBehind[cars,carId],
leftCarForward=leftLaneCarFront[cars,carId],
leftCarForwardPosition, currentCarPosition, leftCarBehindPosition
},

leftCarForwardPosition=cAssociation[leftCarForward]["Velocity"]+cAssociation[leftCarForward]["Position"];
currentCarPosition=cAssociation[carId]["Velocity"]+cAssociation[carId]["Position"];
leftCarBehindPosition=cAssociation[leftCarBehind]["Velocity"]+cAssociation[leftCarBehind]["Position"];

Which[
leftCarForward!=Null & leftCarBehind!=Null & leftCarForwardPosition>currentCarPosition>leftCarBehindPosition & rnd<cAssociation[carId]["W"],
cAssociation[carId]["Lane"]-=1,

leftCarForward!=Null&leftCarForwardPosition>currentCarPosition & rnd<cAssociation[carId]["W"],
cAssociation[carId]["Lane"]-=1,

leftCarBehind!=Null&currentCarPosition>leftCarBehindPosition & rnd<cAssociation[carId]["W"],
cAssociation[carId]["Lane"]-=1
];
cAssociation
]
```

Рисунок 20 – Функция *changeLaneToLeft*

1. На вход функция получает ассоциацию – *cars*, идентификатор текущей машины – *carId* и случайное вещественное число – *rnd*.

2. В этой функции были использованы двумя вспомогательными функциями – *leftLaneCarBehind* и *leftLaneCarFront*. По своей сути эти функции похожи, различие их только в том, что *leftLaneCarBehind* ищет идентификатор автомобиля на левой полосе движения, который идёт впереди текущего автомобиля, а *leftLaneCarFront* ищет идентификатор автомобиля впередиидущего. Наглядно это продемонстрировано на рисунке №1, где результатом работы функции *leftLaneCarBehind* является автомобиль с идентификатором 3, а результатом работы функции *leftLaneCarFront* является автомобиль с идентификатором 1.

3. Далее в соответствии с формулами № 6-11 происходит преобразование.

Результатом работы функции *changeLaneToLeft* является ассоциация, в которой автомобиль с идентификатором *carId* в лучшем случае сменит полосу движения. Так же можно представить результат работы функции в следующем виде (см. рисунок 21).

```
showExample(carsDataSample,  
changeLaneToLeft[3, 0.1])
```

Входные данные:

	Velocity	Lane	Time	Position	W	Color
0	0	1	0	{15, 30}	0	■
1	3	0	1	13	0.8	■
2	4	1	2	11	0.8	■
3	3	1	3	8	0.2	■

Результат преобразования:

	Velocity	Lane	Time	Position	W	Color
0	0	1	0	{15, 30}	0	■
1	3	0	1	13	0.8	■
2	4	1	2	11	0.8	■
3	3	0	3	8	0.2	■

Рисунок 21 – Пример работы функции *changeLaneToLeft*

Как можно видеть на рисунке №21, у автомобиля с идентификатором 3 изменилась полоса движения («Lane» → 0).

4.4 Обработка итераций модели и визуализация

Для запуска модели была создана функция *modelIterations*, которая в результат возвращает список итераций модели (список ассоциаций).

```
Clear[modelIterations]
modelIterations[cars_,maxVelocity_,iterationCount_,pBreak_,coefficientW_,maxW_] := Module[
{
  iterCars=cars,
  iterationList={},
  rnd:=RandomReal[]
},
Do[

Do[
  iterCars=Composition[
    changelaneToLeft[car,rnd],
    probabilityW[car,maxVelocity,coefficientW,maxW],
    driving[car],
    randomization[car,pBreak,rnd],
    barking[car],
    acceleration[car,maxVelocity]][iterCars],
    {car,currentCars[cars,time]}}];

AppendTo[iterationList,iterCars],

{time,iterationCount}
];
iterationList
]
```

Рисунок 22 – Функция *modelIterations*

Так же была создана функция *roadPicture*, которая с помощью функции *ArrayPlot* визуализирует текущую ситуацию на проезжей части с текущими введёнными параметрами. Пользователь может строить модель с любыми, приведёнными на рисунке №23 параметрами, так же он может настроить визуализацию под удобные для него настройки (см. рисунок 23).

roadPicture

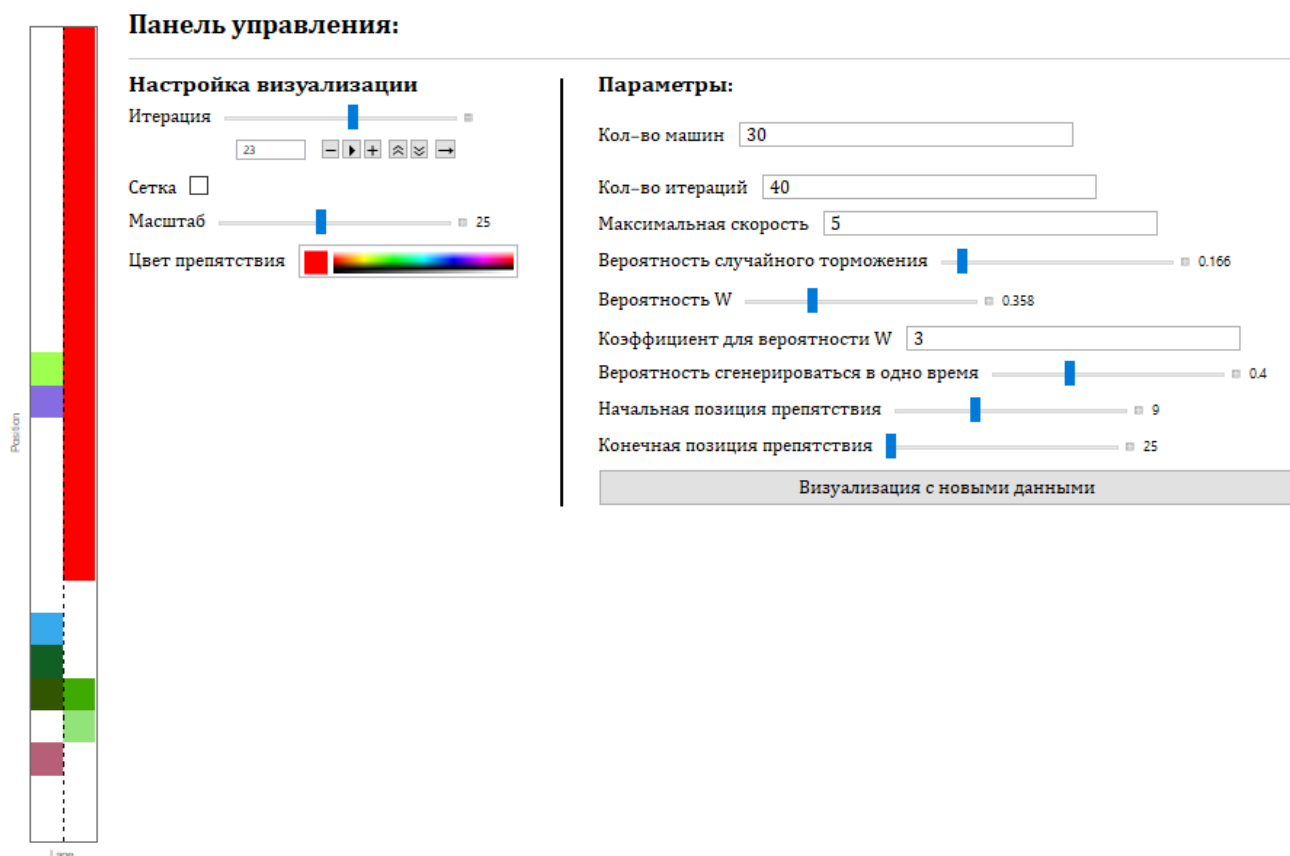


Рисунок 23 – Результат выполнения функции *roadPicture*

Как можно видеть на рисунке №23 пользователю доступны параметры, такие как: регулирование количеством автомобилей – *carCount*, количество итераций *iterationCount*, которые хочет просмотреть пользователь, значение максимальной скорости – v_{max} , вероятность случайного дополнительного торможения – p_{brake} , вероятность W , коэффициент для вероятности W – c , вероятность сгенерироваться автомобилей в одно время – *probabilityOneTime*, начальная позиция препятствия – *posStart*, конечная позиция препятствия – *posFinish*.

5. ПЛАТФОРМЫ ДЛЯ АНАЛИЗА МОДЕЛИ

По окончании создания модели, были созданы платформы, где пользователь может анализировать модель по различным параметрам. Первая платформа – *dataZonesAnalysis* служит для анализа средней скорости и плотности потока автомобилей в трёх основных зонах (см. пункт 3). Вторая платформа – *lanesInteractiveAnalysis* служит для анализа плотности потока автомобилей на полосах автострады до препятствия. После анализа модели становится ясно, что чем меньше вероятность W и чем больше вероятность дополнительного торможения p_{brake} , тем выше плотность автомобилей на всём протяжении рассматриваемого участка дороги. Так же при увеличении вероятности W возрастает плотность автомобилей на правой полосе движения. (см. рисунки 24 и 25)

dataZonesAnalysis

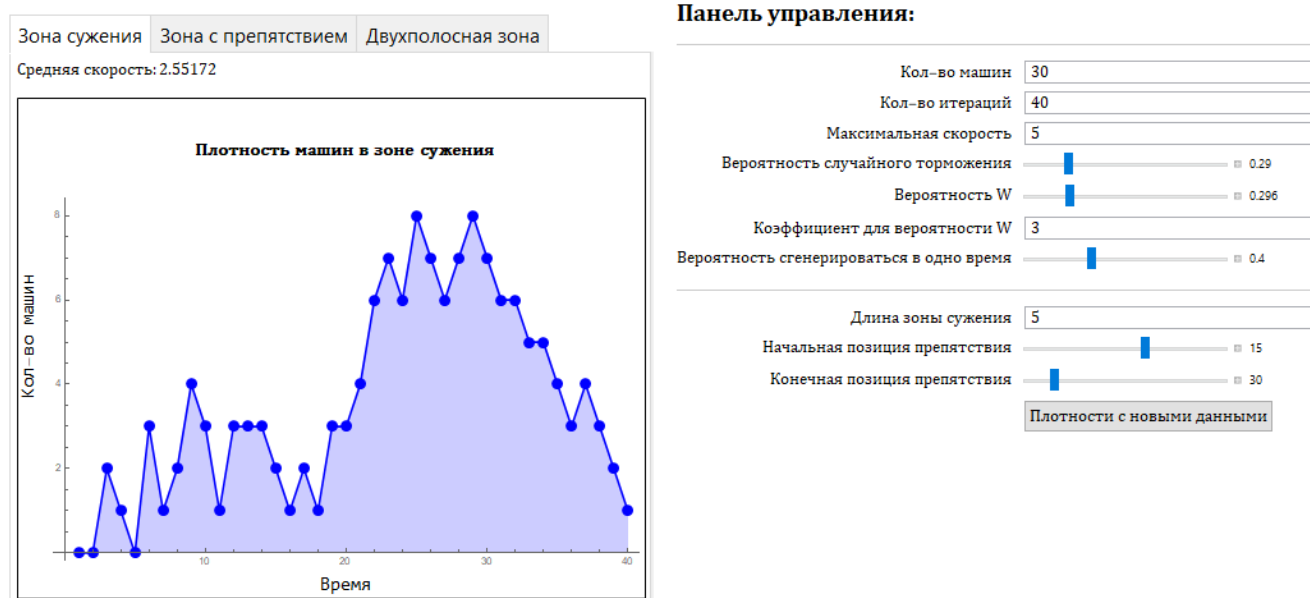


Рисунок 24 – Результат выполнения функции *dataZonesAnalysis*



Рисунок 25 – Результат выполнения функции *lanesInteractiveAnalysis*

Как можно видеть на рисунке № 24 - 25 пользователю доступны параметры, такие как: регулирование количеством автомобилей – *carCount*, количество итераций *iterationCount*, которые хочет просмотреть пользователь, значение максимальной скорости – v_{max} , вероятность случайного дополнительного торможения – p_{brake} , вероятность W , коэффициент для вероятности W – c , вероятность сгенерироваться автомобилями в одно время – *probabilityOneTime*, начальная позиция препятствия – *posStart*, конечная позиция препятствия – *posFinish*.

ЗАКЛЮЧЕНИЕ

В ходе проделанной работы была создана модель и платформа для её анализа. В работе были продемонстрированы места препятствующие движению, так же они могут возникать в реальном сценарии движения, например, из-за дорожных работ. Основываясь на построенной двухполосной модели движения, которая в свою очередь основана на модели Нагеля-Шрекенберга, были введены три основных участка дороги, которые делят проезжую часть: большая часть – двухполосная зона, улица с двумя полосами, на которой действуют правила двух моделей движения. Кроме того, перед началом дорожных работ существует зона сужения, в которой применяется другой набор правил, так что автомобили на правой полосе могут перестроиться на левую полосу, в соседнюю ячейку. Наконец, есть зона с препятствием, в которой модель Нагеля-Шрекенберга применяется для движения по одной полосе. Так же в ходе анализа модели было выяснено, что чем меньше вероятность W и чем больше вероятность дополнительного торможения p_{brake} , тем выше плотность автомобилей на всём протяжении рассматриваемого участка дороги. Так же при увеличении вероятности W возрастает плотность автомобилей на правой полосе движения.

Таким образом, эта модель может использоваться для анализа реальной ситуации на автострате, когда ставится задача исследовать случай с перекрытием одной из полос движения.

СПИСОК ЛИТЕРАТУРЫ

1. Anja Ebersbach Two-Lane Traffic with places of obstruction to traffic / Anja Ebersbach, Johannes J. Schneider // International Journal of Modern Physics C. – 2004. – N 15 (4). – С. 1 – 10.
2. Institute for Theoretical Physics University of Cologne [Электронный ресурс] / Traffic flow modelling; ред. Andreas Schadschneider, 2020. – Режим доступа: <http://www.thp.uni-koeln.de/~as/MyPage/traffic.html>, свободный. – Загл. с экрана. – Яз. англ., нем.
3. Wolfram User Portal – Режим доступа: <https://user.wolfram.com>