Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

“Национальный исследовательский университет ИТМО”

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

РАЗРАБОТКА МОДЕЛИ ОБНАРУЖЕНИЯ НАРУШЕНИЙ СОДЕРЖАТЕЛЬНОЙ ЦЕЛОСТНОСТИ В КИБЕР-ФИЗИЧЕСКИХ СИСТЕМАХ С ИСПОЛЬЗОВАНИЕМ ТЕОРИИ ИГР

Автор Мариненков Егор Денисович _____
(Фамилия, Имя, Отчество) (Подпись)

Направление подготовки (специальность) 10.03.01
(код, наименование)
Информационная безопасность

Квалификация бакалавр
(бакалавр, магистр, инженер)*

Руководитель ВКР Викснин И. И., к.т.н. _____
(Фамилия, И., О., ученое звание, степень) (Подпись)

Санкт-Петербург, 2020 г.

Обучающийся____Мариненков Егор Денисович_____
(ФИО полностью)
Группа___N3452_____Факультет/институт/кластер__БИТ_____

Направленность (профиль), специализация Организация и технология защиты информации_
_____

Консультант (ы):
а) ____Чупров Сергей Сергеевич_____ _____
(Фамилия, И., О., ученое звание, степень) (Подпись)

б) _____ _____
(Фамилия, И., О., ученое звание, степень) (Подпись)

ВКР принята "_ __ "_____ ___20 ___ г.

Оригинальность ВКР _____%

ВКР выполнена с оценкой _____

Дата защиты "_17_ "_____июня_____20 _20_ г.

Секретарь ГЭК ____ Коваль Елена Николаевна_____ _____
(ФИО) (подпись)

Листов хранения _____85_____

Демонстрационных материалов/Чертежей хранения _____

Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

"НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО"

УТВЕРЖДАЮ

Руководитель ОП

Канжелев Ю.А. _____

(Фамилия, И.О.) (подпись)

«____» «_____» 20____ г.

З А Д А Н И Е
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

Обучающемуся Мариненкову Егору Денисовичу **Группа** N3452 **Факультет** БИТ

Руководитель ВКР Викснин Илья Игоревич, к.т.н., Университет ИТМО, научный сотрудник

(ФИО, ученое звание, степень, место работы, должность)

1 Наименование темы: Разработка модели обнаружения нарушений содержательной целостности в кибер-физических системах с использованием теории игр

Направление подготовки (специальность) 10.03.01 «Информационная безопасность»

Направленность (профиль) «Организация и технология защиты информации»

Квалификация бакалавр

2 Срок сдачи студентом законченной работы «31» «мая» 20 20 г.

3 Техническое задание и исходные данные к работе

1. Анализ существующих методов, подходов и моделей в области выявления нарушений содержательной целостности в кибер-физических системах.
2. Формализация модели информационного взаимодействия элементов кибер-физической системы.
3. Разработка модели нарушений содержательной целостности информационного сообщения в информационном взаимодействии элементов кибер-физической системы.
4. Выявление уязвимостей в информационном взаимодействии элементов кибер-физической системы.
5. Разработка модели обнаружения нарушений содержательной целостности в кибер-физической системе.
6. Оценка эффективности разработанной модели.

4 Содержание выпускной квалификационной работы (перечень подлежащих разработке вопросов)

1. Анализ предметной области.

2. Разработка модели нарушений содержательной целостности информационного сообщения в информационном взаимодействии элементов кибер-физической системы.

3. Разработка модели обнаружения нарушений содержательной целостности в кибер-физической системе с использованием теории игр.

5 Перечень графического материала (с указанием обязательного материала)

1. Платежная матрица игры (таблица).

2. Результаты оценки эффективности разработанной модели (таблица).

6 Исходные материалы и пособия

1. Викснин, И.И. Модели и методы обнаружения нарушений целостности информации в группах беспилотных транспортных средств: диссертация ... канд. тех. наук: 05.13.19 / Викснин Илья Игоревич. – СПб., 2018. – 207 с.

2. Петросян, Л.А. Теория игр: учебник / Л.А. Петросян, Н.А. Зенкевич, Е.В. Шевкопляс. – 2-е изд., перераб. и доп. – СПб.: БХВ-Петербург, 2017. – 432 с.

3. Кузин, Л.Т. Основы кибернетики: В 2-х т. Т. 2. Основы кибернетических моделей: учебное пособие для вузов / Л.Т. Кузин. – М.: Энергия, 1979. – 584 с.

7 Дата выдачи задания «_10_» «___февраля___» 20_20_г.

Руководитель ВКР_Викснин Илья Игоревич__
(подпись)

Задание принял к исполнению_Мариненков Егор Денисович_ «_10_» «_февраля_» 20_20_г.
(подпись)

Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

"НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО"

АННОТАЦИЯ

ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Обучающийся Мариненков Егор Денисович
(ФИО)

Наименование темы ВКР: Разработка модели обнаружения нарушений содержательной целостности в кибер-физических системах с использованием теории игр

Наименование организации, где выполнена ВКР Университет ИТМО

ХАРАКТЕРИСТИКА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

1 Цель исследования повышение точности метода обнаружения нарушений содержательной целостности, основанного на репутационных механизмах

2 Задачи, решаемые в ВКР 1) адаптировать модель КФС в рамках поставленной цели; 2) проанализировать метод обнаружения нарушений содержательной целостности, основанный на репутационных механизмах; 3) разработать модель нарушения содержательной целостности информации в рамках КФС; 4) адаптировать метод обнаружения нарушений содержательной целостности, основанный на репутационных механизмах, в рамках модели КФС; 5) разработать подход к формированию субъективной оценки информации элементами КФС в случаях неполноты данных; 6) провести моделирование метода обнаружения нарушений содержательной целостности с использованием и без использования разработанного подхода.

3 Число источников, использованных при составлении обзора 6

4 Полное число источников, использованных в работе 23

5 В том числе источников по годам

Отечественных			**Иностранных**		
Последние 5 лет	От 5 до 10 лет	Более 10 лет	Последние 5 лет	От 5 до 10 лет	Более 10 лет
5	3	5	4	4	2

6 Использование информационных ресурсов Internet да, 4
(Да, нет, число ссылок в списке литературы)

7 Использование современных пакетов компьютерных программ и технологий (Указать, какие именно, и в каком разделе работы)

Пакеты компьютерных программ и технологий	**Раздел работы**
Python 3.7	3
Microsoft Word 2016	Вся работа
Microsoft Excel 2016	3
Microsoft PowerPoint 2016	Презентация

8 Краткая характеристика полученных результатов разработана модель обнаружения нарушений содержательной целостности в кибер-физических системах с использованием теории

игр, которая позволяет повысить точность выявления некорректного поведения среди диверсантов, присутствующих в системе, на 15%.

9 Полученные гранты, при выполнении работы нет
(Название гранта)

10 Наличие публикаций и выступлений на конференциях по теме выпускной работы да
(Да, нет)

а) 1 Marinenkov E., Chuprov S., Viksnin I., Kim I. Empirical study on trust, reputation, and game theory approach to secure communication in a group of unmanned vehicles // CEUR Workshop Proceedings – 2020. Vol. 2590. P. 1 – 12.
(Библиографическое описание публикаций)

б) 1 XI Международная научно-практическая конференция «Программная инженерия и компьютерная техника» (Майоровские чтения) The Majorov International Conference on Software Engineering and Computer Systems (MICSECS-2019), 2019.
(Библиографическое описание выступлений на конференциях)

2 XLIX научная и учебно-методическая конференция Университета ИТМО, 2020.

3 IX Всероссийский конгресс молодых ученых, 2020.

Обучающийся Мариненков Егор Денисович _____
(ФИО) (подпись)

Руководитель ВКР Викснин Илья Игоревич _____
(ФИО) (подпись)

“ 18 ” мая 20 20 г.

ОГЛАВЛЕНИЕ

Список сокращений и условных обозначений

ИБ – информационная безопасность

ИВ – информационное взаимодействие

КФС – кибер-физическая система

МАС – мультиагентная система

СДИВ – скрытое деструктивное информационное воздействие

ВВЕДЕНИЕ

В связи с наступлением Четвертой промышленной революции в настоящее время в сферы человеческой жизнедеятельности активно внедряются кибер-физические системы (КФС). Согласно [1] под КФС следует понимать умные системы, включающие спроектированные взаимодействующие сети физических и вычислительных компонентов. Иными словами в классические автоматизированные системы внедряются вычислительные ресурсы, способствующие взаимодействию систем между собой и повышающие тем самым автономность подобных систем.

Увеличение возможностей подобных систем неизбежно ведет к появлению новых угроз информационной безопасности (ИБ). Несмотря на то, что для обеспечения целостности, доступности и конфиденциальности информации существует множество подходов, методов и средств, выделяют нарушения, которые невозможно выявить данными классическими подходами по обеспечению ИБ и прочим [2; 3]. В литературе атаки, вызывающие подобные нарушения, именуются «мягкими». В контексте данной работы под «мягкими» атаками будет пониматься скрытое деструктивное информационное воздействие (СДИВ) на информационное взаимодействие (ИВ) элементов КФС, нарушающее содержательную целостность информации. В данной работе под целостностью понимается точность и полнота информации [4]. Согласно работе [5] семантические (содержательные) проблемы информационного взаимодействия элементов связаны с идентичностью или достаточным приближением значения информации для получателя и предполагаемым значением информации для отправителя. Иными словами, в случае, когда принимающий информацию элемент рассчитывает на достоверную информацию, а получает дезинформацию, появляется семантическая проблема. Поэтому в данной работе под содержательной целостностью будет пониматься точность значения информации, нарушение которой может произойти в случае нарушения функционирования физических компонентов КФС или умышленной передачи недостоверных данных диверсантом.

Для обнаружения подобного рода нарушений используются методы, основанные на репутационных механизмах, которые позволяют выявлять СДИВ благодаря ретроспективе формируемых оценок информации.

Существующие методы обнаружения нарушений содержательной целостности на базе репутационных механизмов в случаях невозможности формирования оценок используют за оценку среднее значение из диапазона допустимых значений для данной оценки. Одним из подобных методов является метод, разработанный И. И. Викснинym в рамках его диссертации [6]. Данная работа направлена на разработку алгоритма формирования оценок в подобных случаях, основанного на равновесии в существующем множестве вариантов. Для этого используется базис теории игр, который позволяет определить равновесие при представлении механизма формирования оценки, как игры.

Разработка данного алгоритма позволит повысить эффективность метода обнаружения нарушений содержательной целостности, предложенного И. И. Викснинym, за счет разработки правил по формированию оценки в случаях неполноты данных.

Таким образом объектом исследования является ИБ КФС, а предметом – метод обнаружения нарушений содержательной целостности. В качестве гипотезы выдвигается следующее предположение: формирование субъективной оценки информации в случае неполноты данных, основанное на влиянии оцениваемой информации на КФС, позволит увеличить точность метода, разработанного И. И. Викснинym.

Отсюда целью работы является повышение точности метода обнаружения нарушений содержательной целостности, основанного на репутационных механизмах.

Для достижения цели в рамках работы ставятся следующие задачи:

– адаптировать модель КФС в рамках поставленной цели;

– проанализировать метод обнаружения нарушений содержательной целостности, основанный на репутационных механизмах;

– разработать модель нарушения содержательной целостности информации в рамках КФС;

– адаптировать метод обнаружения нарушений содержательной целостности, основанный на репутационных механизмах, в рамках модели КФС;

– разработать подход к формированию субъективной оценки информации элементами КФС в случаях неполноты данных;

– провести моделирование метода обнаружения нарушений содержательной целостности с использованием и без использования разработанного подхода.

Выполнение поставленных задач позволит разработать модель обнаружения нарушений содержательной целостности, применимую в рамках КФС, с использованием теории игр и достичь поставленной цели работы.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Обзор концепции кибер-физических систем

Существующий термин «Индустрия 4.0» появился в 2011 году для обозначения проекта по повышению конкурентоспособности обрабатывающей промышленности [7]. Для осуществления цели проекта предполагалось массовое внедрение вычислительных ресурсов, имеющих доступ в Интернет, в автоматизированные системы для минимизации человеческого фактора. Данная концепция получила название «кибер-физические системы».

Согласно [8] Индустрия 4.0 имеет четыре основания:

– совместимость,

– виртуализация,

– децентрализация,

– работа в режиме реального времени.

Под совместимостью понимается возможность взаимодействия людей, как потребителей, и систем. Подобное взаимодействие не ограничивается областью функционирования одной системы за счет использования различных протоколов связи, что позволяет взаимодействовать различного вида системам.

Для мониторинга состояния системы, выполняющей технологический процесс, используются облачные вычисления, большие данные и Интернет. Данные информационные технологии позволяют создать виртуальную модель системы для дальнейшего осуществления контроля ее функционирования в режиме реального времени.

Под децентрализацией понимается организация структуры системы без использования центральных элементов. В работе [9] выделяются два вида группового управления: централизованная и децентрализованная. Помимо высокой скорости принятия решений по сравнению с централизованным подходом, системы, основанные на децентрализованном подходе, обладают высокой отказоустойчивостью, что, несомненно, является преимуществом при решении проблем ИБ.

Массовое внедрение децентрализованных систем в различные сферы жизнедеятельности человека увеличивает число вычислительных устройств, взаимодействующих между собой и принимающих за счет этого коллаборативно-выработанные решения. Большое количество подобных устройств осложняет реализацию систем, поскольку возникают проблемы, характерные для управления сложными системами [10]. Для решения данных проблем используются подходы, разработанные в рамках теории систем и теории управления [11]. Одним из разделов теории систем, для которого характерно управление распределенными «умными» устройствами, являются мультиагентные системы (МАС).

Парадигма МАС обладает следующими характеристиками [12]:

– автономность элементов – элементы способны самостоятельно управлять своим поведением без вмешательства извне для достижения своих локальных целей;

– взаимодействие с окружающим миром – элементы способны получать информацию из окружающего мира, а также воздействовать на него, с помощью технических устройств;

– наличие программно-аппаратной среды для распределенного взаимодействия;

– наличие средств координации – элементы способны взаимодействовать друг с другом для достижения «глобальной» цели – цели системы.

Поскольку данные характеристики присущи системам в рамках Индустрии 4.0, методы и подходы, лежащие в основе МАС, применимы в рамках управления КФС.

Как было выделено ранее, КФС объединяет физические и вычислительные компоненты, следовательно внутри КФС можно выделить два уровня логики: информационный и физический.

К информационному уровню относятся компоненты, обрабатывающие информацию, формирующие решения и передающие данные решения другим компонентам данного уровня. К физическому можно отнести компоненты

выполняющие команды, поступающие от компонентов информационного уровня, и взаимодействующие с окружающим миром. Таким образом к КФС можно отнести любые взаимодействующие с окружающей средой системы, в которых присутствует контроль и координация действий со стороны коммуникационного центра.

1.2 Адаптация модели кибер-физической системы

1.2.1 Информационное взаимодействие элементов кибер-физической системы

В данной работе КФС представляет собой совокупность гомогенных элементов, основанную на децентрализованном подходе. Гомогенность элементов обеспечит возможность проверки информации, получаемой от других элементов, посредством использования одинакового функционала и технических устройств, что позволит применить в данной системе методы ИБ, основанные на репутационных механизмах.

Подобная КФС, удовлетворяющая критериям гомогенности и децентрализации, описана в работе [13], в рамках организации «умной» сети электроснабжения. Поскольку цель исследования не ограничивает область применения КФС, эта система будет адаптирована для формирования обобщенной модели КФС.

Пусть $CPS = \{e_i, i = \overline{1,n}\}$ – множество элементов КФС. Поскольку каждому элементу присуще свойство гомогенности, все элементы оперируют одной и той же информацией и способны выполнять одни и те же функции.

В качестве информации, циркулирующей внутри КФС, для исследования выделены следующие виды информации об элементе e_i:

– информация о техническом состоянии элемента TS_{ei}, где под техническим состоянием подразумевается совокупность информаций об исправности, местоположении, скорости (в случае динамических элементов) и тому подобные, характеризующие свойства элемента;

– информация о статусе элемента S_{e_i}, где под статусом понимается состояние выполнения задачи или бездействия;

– информация об окружающей среде E_{e_i}, которая была получена элементом с использованием его технических средств;

– иная информация O_{ei}, способствующая достижению цели системы, например, информация о выполненных агентом задачах.

Поскольку в данном исследовании предметом является механизм обнаружения нарушений содержательной целостности, построенный на методе [6], в системе также циркулирует информация I_{eiej} о других элементах e_j, где $j \neq i$, которой владеет элемент e_i:

– информация о технических состояниях элементов– TS_{eiej};

– информация о статусах элементов S_{eiej};

– информация об окружающей среде E_{eiej}, полученной элементами;

– иная информация O_{eiej}.

Всю информацию, имеющуюся у e_i о других элементах, обозначим как I_{ei}. Тогда в КФС циркулируют следующие множества видов информаций обо всех элементах:

– множество информаций о технических состояниях элементов – $TS_{CPS} \ni TS_{ei}$;

– множество информаций о статусах элементов – $S_{CPS} \ni S_{ei}$;

– множество информаций об окружающей среде – $E_{CPS} \ni E_{ei}$;

– множество информаций, имеющихся у элементов КФС о других ее элементах – $I_{CPS} \ni I_{ei}$;

– множество иной необходимой информации об элементах – $O_{CPS} \ni O_{ei}$.

Необходимо отметить, что выделенные виды информаций не всегда могут циркулировать внутри КФС. Наличие того или иного вида необходимо определять в зависимости от цели и особенностей КФС.

1.2.2 Цель кибер-физической системы

В рамках данного исследования целью КФС ставится выполнение максимального количества задач при минимальных затратах. Под ресурсами понимаются любые энергетические, временные и иные ресурсы, затрачиваемые

на выполнение задач. Поскольку элементы КФС используют информацию, циркулирующую в КФС, при функционировании, то количество затрачиваемых ресурсов при выполнении элементами задач напрямую зависит от циркулирующей в КФС информации. Следовательно, необходимо определить порядок расчета количества затраченных ресурсов КФС за все время. Для этого определим функцию расчета затраченных ресурсов элементом e_i в момент времени t.

Пусть существует функция $act(TS_{ei}^t, S_{ei}^t, E_{ei}^t, I_{ei}^t, O_{ei}^t)$, определяющая оптимальное для элемента e_i действие в момент времени t. В данном случае под оптимальным действием понимается действие, при котором элемент затратит наименьшее количество ресурсов для достижения выполняемой задачи. Тогда существует функция расчета затрат на данное действие:

$$c_{ei}^t = costs\left(act(TS_{ei}^t, S_{ei}^t, E_{ei}^t, I_{ei}^t, O_{ei}^t)\right), \quad (1)$$

где c_{ei}^t – количество затраченных ресурсов в момент времени t, $costs()$ – сама функция расчета количества затрат.

Тогда, основываясь на (1), количество затраченных ресурсов КФС за все моменты времени до момента t включительно можно рассчитать, как:

$$c_{CPS} = \sum_{k=0}^{t}\sum_{i=1}^{n} c_{ei}^t = \sum_{k=0}^{t}\sum_{i=1}^{n} costs\left(act(TS_{ei}^t, S_{ei}^t, E_{ei}^t, I_{ei}^t, O_{ei}^t)\right), \quad (2)$$

где c_{CPS} – количество всех ресурсов, затраченных КФС.

Поскольку для выполнения части цели, связанной с минимальными затратами, оптимальным поведением КФС будет полное бездействие, необходимо сформулировать условие, которое не позволит системе бездействовать.

Пусть $\exists T^d \in T: T = \{t_i | i = \overline{0,m}\}$ – множество выполненных задач элементами КФС, являющееся подмножеством всех доступных задач КФС.

Тогда условие, обязующие КФС выполнять задачи, выглядит следующим образом:

$$\left|T^d\right| \to |T|. \tag{3}$$

Исходя из (2) и (3) цель системы формируется следующим образом:

$$\begin{cases} c_{CPS} \to 0 \\ \left|T^d\right| \to |T| \end{cases}$$

или

$$\begin{cases} \sum_{k=0}^{t} \sum_{i=1}^{n} costs\left(act(TS_{ei}^t, S_{ei}^t, E_{ei}^t, I_{ei}^t, O_{ei}^t)\right) \to 0 \\ \left|T^d\right| \to |T| \end{cases}. \tag{4}$$

1.3 Анализ метода обнаружения нарушений содержательной целостности на основе репутационных механизмов

Существующие методы и подходы по защите информации от СДИВ, которые основаны на репутационных механизмах, используют показатели репутации и доверия для обнаружения нарушений содержательной целостности [14-17]. В случаях, когда показатели сформировать невозможно, например в момент инициализации системы, за значение показателя берут среднее значение из диапазона допустимых значений. В общем случае среднее значение равно 0.5, а сам диапазон определяется, как $[0; 1]$.

В рамках моделей репутации и доверия принято считать, что репутация является функцией, которая зависит от доверия, которое вычисляется в предыдущий момент времени. Отсюда следует, что доверие оказывает косвенное влияние на формирование репутации. В случае линейности функции репутации изменение данной функции будет недостаточно резким при резком изменении доверия, что приведет к неверным решениям относительно доверенности

элемента при его продолжительном функционировании. Так, например, диверсант, репутация которого была занижена при определении нарушения, сможет восстановить прежний уровень репутации за относительно малое время. Решение данной проблемы было представлено в диссертации И.И. Викснина [6].

В рамках данной диссертации метод обнаружения нарушений содержательной целостности базируется на модифицированной системе показателей, в которую входят показатели истинности, репутации и доверия.

Под истинностью понимается субъективная оценка информации в определенный момент времени, формируемая элементом-субъектом, получающим информацию от элемента-объекта, чье поведение оценивается. Под репутацией понимается показатель, формируемый элементом-субъектом во времени и в процессе оценки истинности. Под доверием понимается субъективная оценка поведения элемента-объекта, формируемая элементом-субъектом на основе репутации элемента-объекта, формируемой КФС во времени, и показателя истинности, формируемом элементом-субъектом.

Для данных показателей существуют следующие допущения:

– истинность $Truth \in [0; 1]$,

– репутация $R \in [0; 1]$,

– доверие $Trust \in [0; 1]$.

Опишем логику метода. Пусть $\exists e_i, e_j \in E: i \neq j$, где e_i – элемент-субъект, e_j – элемент-объект, а $E = \{e_k | k = \overline{1, N}\}$ – множество элементов КФС. Каждый элемент e_k обладает пассивными и активными знаниями, где пассивные знания KN_{ekpas} – информация, полученная благодаря устройствам элемента e_k или других элементов, которая не влияет прямо на функционирование КФС, а активные KN_{ekact} – информация, полученная в результате активной работы элементов множества E, направленной на достижение цели КФС. Пусть e_j передает информацию $S = \{s_l | l = \overline{1, bl}\}$ элементу e_i, где s_l – один из bl блоков информации. Тогда показатель $Truth_{eiej}^{S}$ будет рассчитываться по формуле:

$$Truth^{S}_{eiej} = \frac{\sum_{l=1}^{bl} Truth^{sl}_{eiej}}{bl}, \quad (5)$$

где $Truth^{sl}_{eiej}$ – оценка истинности блока l, рассчитываемая, как:

$$Truth^{sl}_{eiej} = \begin{cases} 1, \text{если информация корректная} \\ 0, \text{если информация некорректная} \end{cases}. \quad (6)$$

Важно отметить, что (6) справедлива только в том случае, когда e_i обладает информацией, входящей в KN_{eipas} и способствующей оценке блока полученной информации. В ином случае e_i может запросить оценку у других элементов $e_k \in E, k \neq i, k \neq j$ имеющих возможность ИВ с элементом e_i. Тогда истинность блока информации $Truth^{sl}_{eiej}$ будет формироваться как усредненное значение полученных оценок:

$$Truth^{sl}_{ej} = \frac{\sum_{k=1}^{ntruth} Truth^{sl}_{ekej}}{n_{truth}}, \quad (7)$$

где $Truth^{sl}_{ekej}$ – субъективная оценка блока информации s_l, формируемая элементом e_k и рассчитываемая по формуле (6),

n_{truth} – количество элементов, имеющих возможность оценить информацию.

В случаях, когда элемент-субъект не имеет возможности сформировать оценку на основе своих пассивных знаний или на основе оценок других элементов, показатель $Truth$ принимается за значение 0.5, то есть среднее значение, при котором информация не оценивается ни как корректная, ни как некорректная. Тогда в общем виде, с использованием (6) и (7), показатель истинности блока s_l формируется по следующей формуле:

$$Truth_{eiej}^{sl} = \begin{cases} \begin{cases} 1, s_l \text{ корректен} \\ 0, s_l \text{ некорректен} \end{cases}, e_i \text{ обладает } Inf \in KN_{eipas} \\ \frac{\sum_{k=1}^{ntruth} Truth_{ekej}^{sl}}{n_{truth}}, e_i \text{ не обладает } Inf \in KN_{eipas} \text{ и } n_{truth} \neq 0 \text{'} \\ 0.5, e_i \text{ не обладает } Inf \in KN_{eipas} \text{ и } n_{truth} = 0 \end{cases} \quad (8)$$

где Inf – информация, способствующая формированию показателя истинности.

Тогда во всех случаях, основываясь на (8), можно сформировать показатель истинности по формуле (5).

Как было сказано ранее, показатель репутации рассчитывается во времени на основе показателя истинности. Для решения проблемы недостаточно резкого убывания функции в случае выявления нарушения, автор взял за основу функцию распределения Вейбулла [18] с коэффициентами $k = 1, \lambda = 1$, что позволяет экспоненциально снизить значение репутации в случае нарушения. Для расчета репутации, когда информация корректна, характер функции остается линейным, что позволяет повышать репутацию элемента во времени при отсутствии нарушений. Таким образом показатель репутации R_{eiejt} в момент времени t рассчитывается с по формуле:

$$R_{eiejt} = \begin{cases} \frac{\sum_{k=0}^{t-1} R_{eiejk} + Truth_{eiejt}^{S}}{t+1}, Truth_{eiejt}^{S} \geq \alpha \\ \frac{\sum_{k=1}^{t-1} R_{eiejk} - \left(\frac{\sum_{k=1}^{t-1} R_{eiejk}}{t-1} - e^{(1-Trutht)*t} \right)}{t+1}, Truth_{eiejt}^{S} < \alpha \end{cases}, \quad (9)$$

где α – граничный уровень истинности, при котором информация S определяется как корректная и который определяется эмпирически.

В общем случае $\alpha = 0.5$. Согласно автору метода для расчета R в момент времени $t = 0$ показатель репутации можно принимать за показатель истинности, следовательно $R_{eiej0} = Truth_{eiej0}^{S}$.

Последний показатель доверия является функцией от показателей $Truth$ и R. Он вводится для уменьшения количества ошибок первого и второго рода при ошибочной оценке $Truth$ или R. То есть элемент-субъект должен иметь

возможность корректно оценить поведение элемента-объекта в тех случаях, когда оценка истинности была сформирована некорректно, но оценка репутации, за счет ретроспективы, остается корректной, либо, когда оценка истинности была сформирована корректно в настоящий момент времени, но при формировании репутации в прошлые моменты времени ее оценка была некорректной. Тогда показатель доверия $Trust$ может быть сформирован на основе (5) и (9) следующим образом:

$$Trust_{eiejt} = \gamma Truth^{S}_{eiejt} + (1-\gamma)R_{eiejt-1},$$

где $\gamma \in [0;1]$ – коэффициент реактивности функции.

Тогда нарушение со стороны элемента-объекта будет отсутствовать при $Trust_{eiejt} \geq \alpha_{trust}$, где α_{trust} – пороговое значение доверия, определяемое эмпирически. Коэффициент реактивности может быть определен при анализе статистических данных, что позволит установить его эмпирически и снизить количество ошибок первого или второго рода. Также автором предлагается другой подход к формированию показателя доверия. Если рассмотреть множество возможных комбинаций показателей истинности и репутации как точки на графике с осями R и $Truth$, то показатель доверия можно рассчитать исходя из удаленности точки $\left(R_{eiejt-1}, Truth^{S}_{eiejt}\right)$ от положений $(R_{min}, Truth_{min})$, в котором поведение элемента-объекта однозначно определяется как некорректное, и $(R_{max}, Truth_{max})$, в котором поведение элемента-объекта однозначно определяется как корректное. В общем виде подобный подход отображен на рисунке 1.

Тогда, используя формулы расчета расстояния на плоскости, показатель доверия можно определить следующим образом:

$$Trust_{eiejt} = \sqrt{\left(R_{min} - R_{eiejt-1}\right)^2 + \left(Truth_{min} - Truth^S_{eiejt}\right)^2} -$$

$$-\sqrt{\left(R_{max} - R_{eiejt-1}\right)^2 + \left(Truth_{max} - Truth^S_{eiejt}\right)^2}. \quad (10)$$

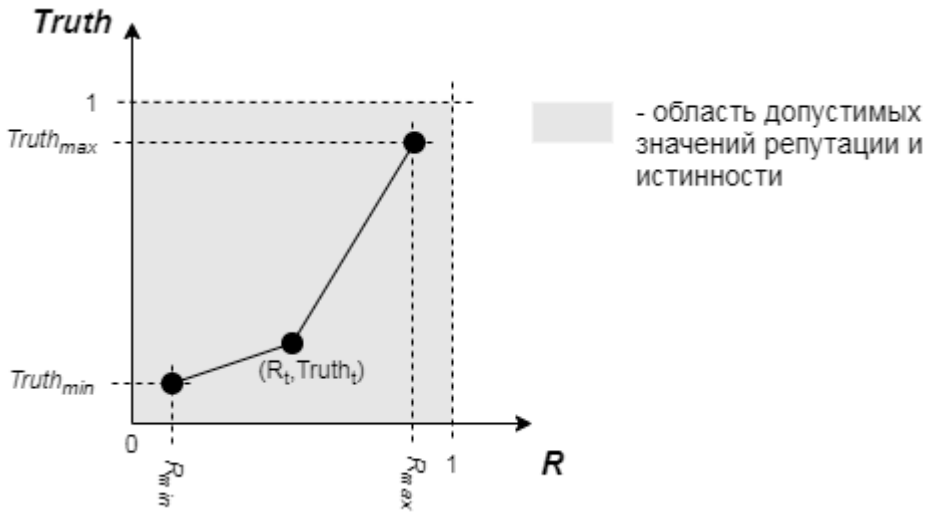

Рисунок 1 – Общий вид расчета показателя доверия по удалению от позиций некорректного и корректного поведения

Таким образом формирование доверия по (10) позволяет определять показатель без предварительного проведения экспериментов и анализа статистических результатов, то есть нет необходимости в предварительном поиске оптимального коэффициента γ.

1.4 Постановка задачи

Согласно пункту 1.2 КФС в данной работе рассматривается как децентрализованная МАС, в которой элементы гомогенны. Метод, рассмотренный в пункте 1.3, применяется в рамках группы беспилотных транспортных средств, которая отвечает требованиям гомогенности и децентрализации и классифицируется как КФС, применяемая в сфере транспорта.

В рамках настоящей работы область применения не ограничивается, более того модель КФС, представленная в 1.2, сформирована в общем виде, что создает новый научный задел – применение метода репутации и доверия в рамках обобщенной модели КФС.

Рассмотренный метод использует экспоненциальное снижение репутации, в случае некорректного поведения элемента, и несколько показателей при формировании показателя доверия, что позволяет снизить процент ошибок первого и второго рода. Рассматривая формирование оценки блока по (8), видно, что существует ситуация, когда показатель истинности невозможно определить по (6) или (7), и за показатель принимается значение 0.5. Такое значение не позволяет охарактеризовать информацию ни как корректную, ни как некорректную, что добавляет элемент непредсказуемости при дальнейшей оценке поведения элемента.

В настоящей работе было выдвинуто предположение о повышении точности метода, описанного в 1.3, за счет определения показателя истинности в описанной выше ситуации, основанного на влиянии информации на достижение цели КФС. Подобный подход был реализован в работе [19], где для формирования показателя использовалась теория игр. В результате было получено повышение точности на 1% и снижение ошибки второго рода примерно в 8 раз. В месте с этим было получено увеличение ошибок первого рода примерно в 12 раз. Эти результаты были получены при решении игры в чистых стратегиях, что повлекло формирование показателя $Truth = 0$ для любого вида информаций. Как видно данный подход имеет недостатки, что говорит о необходимости использования иного подхода.

В настоящей работе предполагается, что, в случае неполноты данных – когда элемент не имеет возможности оценить информацию на основе своих пассивных знаний или оценок других элементов, вероятностная оценка истинности информации на основе ее влияния на цель позволит повысить точность метода. Подобный подход можно реализовать с использованием теории игр, а именно с применением смешанного расширения игры, при котором ситуация равновесия существует всегда [20], что позволяет сформировать показатель истинности в любых случаях.

Выводы по главе 1

В данной главе была разработана обобщенная модель КФС целью которой является использование минимального количества ресурсов при максимальном количестве выполняемых задач. На достижение цели влияет такая информация, как информация о техническом состоянии, состоянии выполнения задач, окружающей среде, информация о других элементах и иная информация, необходимая для достижения цели. Нарушение содержательной целостности данной информации приведет к нарушению функционирования элемента или всей КФС, поэтому был рассмотрен метод обнаружения нарушений, основанный на репутационных механизмах. Данный метод нацелен на оценку поведения элемента по трем критериям: истинности, репутации и доверия. Было выявлено, что в ситуации неполноты данных за показатель истинности принимается значение 0.5, что не позволяет оценить информацию ни как корректную, ни как некорректную, поэтому была поставлена задача по разработке подхода к формированию показателя истинности в ситуации неполноты данных для вероятностного определения корректности или некорректности информации. Для решения подобных задач подходит теория игр, а именно смешанное расширение игры.

Учитывая неполноту данных, можно выдвигать предположение о корректности информации, основываясь на ее влиянии на цель, поэтому необходимо классифицировать нарушения содержательной целостности, не позволяющие КФС достигать цели и определить характер влияния последствий нарушений на цель.

2 РАЗРАБОТКА МОДЕЛИ НАРУШЕНИЙ СОДЕРЖАТЕЛЬНОЙ ЦЕЛОСТНОСТИ ИНФОРМАЦИОННОГО СООБЩЕНИЯ В ИНФОРМАЦИОННОМ ВЗАИМОДЕЙСТВИИ ЭЛЕМЕНТОВ КИБЕР-ФИЗИЧЕСКОЙ СИСТЕМЫ

2.1 Классификация нарушений содержательной целостности

Как было выделено ранее, нарушение содержательной целостности подразумевает нарушение идентичности смысла (содержания) информации. Таким образом, если элемент передает информацию, значение которой отличается от реального, его поведение должно быть определено как некорректное. Далее для обозначения подобного элемента будет использоваться термин «диверсант».

Поскольку нарушение является результатом реализации угрозы, в первую очередь необходимо ввести классификацию нарушений по намеренности их организации диверсантом. Поэтому, классифицируя нарушения по намеренности их организации, нарушения можно разделить на:

– преднамеренные;

– непреднамеренные.

Под преднамеренными понимаются нарушения, которые были вызваны умышлено для нарушения функционирования элемента или КФС. Такие нарушения являются следствием внедрения в КФС диверсанта или получения несанкционированного доступа к элементу злоумышленником. В любом случае возникновение данного нарушения не связано с исправностью устройств элемента. Обратное можно сказать для преднамеренных нарушений, возникновение которых является следствием возникновения нарушений функционирования элемента. Подобные нарушения возникают в результате неисправности устройств, анализирующих окружающую среду, или обрабатывающих информацию.

Рассматривая цель КФС по (4) можно выделить две составляющих цели: основную, смысл которой заключен в снижении затрат на выполнение действий, и дополнительную, при которой КФС стремится выполнить максимальное

количество задач из множества существующих. Тогда нарушения можно классифицировать по характеру влияния на цель:

– прямое влияние;

– косвенное влияние;

– комплексное влияние.

При прямом влиянии затраты на выполнение задач увеличиваются, при этом нарушение не влияет на количество выполняемых задач. Если обозначить затраты в случае нарушения как $c_{\widetilde{CPS}}$, а множество выполненных задач как $\widetilde{T^d}$, то (4) в случае прямого влияния нарушения на цель можно представить как:

$$\begin{cases} c_{\widetilde{CPS}} > c_{CPS} \\ \left|\widetilde{T^d}\right| \cong |T^d| \end{cases}.$$

Результатом прямого влияния может быть также снижение количества выполненных задач, например, когда затраты были увеличены на столько, что элементы не обладают ресурсами на выполнение остальных задач. Но нарушение оказывающее прямое влияние на цель не нацелено на изменение количества выполненных задач, поэтому считается, что количество выполненных задач с нарушением и без приблизительно равны. Если рассматривать косвенное влияние, то нарушение содержательной целостности приводит к снижению количества выполненных задач при неизменных затратах. Тогда (4) можно представить как:

$$\begin{cases} c_{\widetilde{CPS}} \cong c_{CPS} \\ \left|\widetilde{T^d}\right| < |T^d| \end{cases}.$$

Комплексное влияние является следствием прямого или косвенного влияния, то есть когда прямое влияние приводит к косвенному влиянию, либо косвенное влияние приводит к прямому. Тогда влияние на цель выглядит следующим образом:

$$\begin{cases} c_{\widetilde{CPS}} > c_{CPS} \\ \left|\widetilde{T^d}\right| < |T^d| \end{cases}.$$

Соотнесение нарушения в рамках данной классификации реализуется с помощью наблюдения за изменением цели и определения наиболее резкого изменения параметров.

В разделе 1.2.1 выделены виды информации, которые циркулируют внутри КФС. Поскольку данные виды применяются в рамках КФС для разных целей их нарушение будет оказывать различное влияние. Как было выделено в разделе 1.3 элементы КФС обладают пассивными и активными знаниями. Пассивные знания KN_{pas} не участвуют в принятии элементами КФС коллективно-выработанного решения, но могут косвенно влиять на эти решения. К таким знаниям можно отнести информацию об окружающей среде. К активным знаниям KN_{act} относятся информация, которая активно используется при распределении задач и формировании иных коллективных. Тогда к активным знаниям относится информация о техническом состоянии и статусе элемента. Поскольку в модели КФС четко не оговорено как именно иная информация влияет на цель, ее можно отнести и к пассивным, и к активным знаниям. Важно, то что характер возникновения описанной информации не влияет на соотнесение к виду знаний. То есть если вид информации может быть получен элементом благодаря его устройствам или ИВ с другими элементами, то в любом случае данный вид информации будет отнесен к одному и тому же виду знаний. Таким образом, классифицируя нарушения по виду информации, их можно разделить на:

1) пассивные знания:

– информация об окружающей среде;

– иная информация.

2) активные знания:

– техническое состояние;

– статус;

– иная информация.

Тогда общая классификация нарушений, которая объединяет все 3 классификации, представлена на рисунке 2.

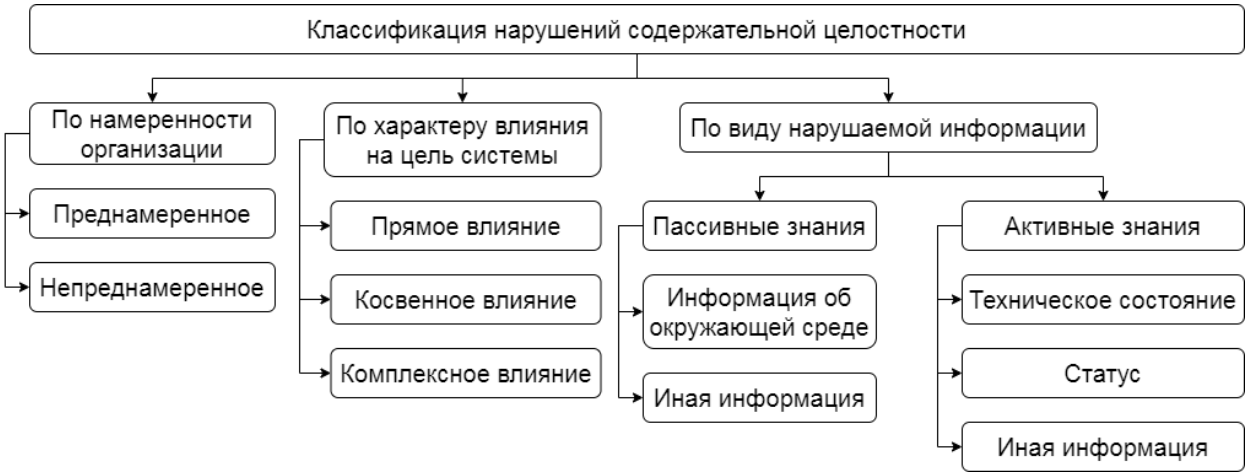

Рисунок 2 – Общая классификация нарушений содержательной целостности в рамках модели кибер-физической системы

Из цели КФС, описанной по (4), видно, что информация явным образом влияет на затраты, поэтому далее необходимо рассмотреть как именно прямое влияние нарушений изменяет затраты КФС на выполнение поставленных целей.

2.2 Влияние нарушений на цель кибер-физической системы

Как видно из (4) все ранее оговоренные виды информаций влияют на достижение цели КФС. Для оценки данного введем понятие ценности информации в момент времени t, где под ценностью будем понимать количественную характеристику вида информации в системе, характеризующую влияние информации на достижение цели в момент времени t.

Пусть $\mathrm{Inf}_{ei}^{t} \in \{TS_{ei}^{t}, S_{ei}^{t}, E_{ei}^{t}, I_{ei}^{t}, O_{ei}^{t}\}$ – вид информации, которым владеет элемент e_i в момент времени t. Пусть $\forall Inf_{ei}^{t} \exists val_{max}(Inf_{ei}^{t}): val_{max}(Inf_{ei}^{t}) > 0$ – функция, оценивающая влияние информации на цель системы не зависимо от времени. Тогда существует функция оценки влияния информации на цель КФС в момент времени t:

$$v_{Inftei} = val(Inf_{ei}^{t}, t', t) = val_{\max}(Inf_{ei}^{t}) * rel(Inf_{ei}^{t}, t', t), \quad (11)$$

где v_{Inftei} – ценность информации Inf_{ei}^{t} в момент времени t, $rel(Inf_{ei}^{t}, t', t) \in (0; 1)$ – функция расчета коэффициента актуальности информации в момент времени t, полученной в момент времени t', где $0 \leq t' < t$, а $\lim\limits_{\substack{t'=const \\ t\to\infty}} rel(Inf_{ei}^{t}, t', t) = 0$, рассчитываемая следующим образом:

$$rel(Inf_{ei}^{t}, t', t) = \left(a_{Inf}\right)^{t-t'}. \quad (12)$$

Как видно из (12) за функцию актуальности выбрана параметрическая функция, где основание $a_{Inf} \in (0,1)$ зависит от вида информации Inf.

В данной работе вводиться ограничение $rel(Inf_{ei}^{t}, t', t) \neq 0$, поскольку в данном случае информация будет считаться неактуальной и ее ценность будет равна нулю. Подобный случай может быть только, когда элемент получит новую информацию, которую он считает более актуальной, поэтому, пока элемент владеет только этой информацией, информация не может быть неактуальной.

Исходя из (12) можно представить (11), как:

$$v_{Inftei} = val(Inf_{ei}^{t}, t', t) = val_{\max}(Inf_{ei}^{t}) * \left(a_{Inf}\right)^{t-t'}. \quad (13)$$

Тогда, основываясь на (13), ценность информации Inf_{ei}^{t} в момент времени t можно выразить следующим образом:

$$v_{inftei} = \begin{cases} val_{\max}(Inf_{ei}^{t}), t' = t \\ val(Inf_{ei}^{t}, t', t), t' \neq t \end{cases}. \quad (14)$$

Необходимо отметить, что использование (14) допустимо только тогда, когда информация является корректной. В случае некорректной информации, которая в дальнейшем будет обозначена как «дезинформация», необходимо определять ее ценность исходя из ее влияния на цель и влияния корректной информации. То есть, если корректная информация изменяет график затрат с

определенной резкостью, то дезинформация изменяет график с большей резкостью, по сравнению с корректной актуальной или менее актуальной информацией. Для этого определим изменения при корректной информации.

Из раздела 1.2.2 видно, что затраты c_{ei}^{t}, рассчитываемые по (1), влияют на цель КФС согласно (4), поэтому оценку влияния информации Inf_{ei}^{t} на цель КФС будем производить по функции (1). Учитывая, что (1) недостаточно определена для математического анализа, в дальнейшем будем считать, что в случае с корректной актуальной информацией рост затрат обладает линейным характером, поэтому функцию (1) можно представить как:

$$c_{ei}(t) = k * (t - (t' - 1)) + b, \tag{15}$$

где t' – момент времени получения актуальной информации, $b = c_{ei}(t' - 1)$ – затраты до получения актуальной информации.

Введем допущение, что при оперировании корректной актуальной информацией коэффициент k в (15) является константой и $k > 0$.

При оперировании корректной но менее актуальной информацией затраты растут быстрее чем при актуальной информации. Учитывая параметрический характер функции (13), по которой определяется ценность менее актуальной информации, функция затрат также должна сохранять параметрический характер. Тогда функцию роста затрат для менее актуальной информации можно сформировать, основываясь на (15), относительно отличия ценностей затрат актуальной и менее актуальной информаций следующим образом:

$$c_{ei}(t) = k\frac{val_{\max}(Inf_{ei}^{t})}{val(Inf_{ei}^{t},t',t)}(t - (t' - 1)) + b = \frac{k}{(a_{Inf})^{t-t'}}(t - (t' - 1)) + b$$

или

$$c_{ei}(t) = k * \left(a_{Inf}\right)^{t\prime - t} * (t - (t' - 1)) + b. \tag{16}$$

Рассмотрим ситуацию при оперировании дезинформацией. Когда в систему попадает дезинформация затраты увеличиваются быстрее, чем при оперировании менее актуальной информацией, поэтому (16) можно представить следующим образом:

$$c_{ei}(t) = k * \left(a'_{Inf}\right)^{t'-1-t} * (t - (t' - 1)) + b, \quad (17)$$

где $a'_{Inf} < a_{Inf}$ – коэффициент влияния дезинформации на цель.

Для (15)-(17) вводится допущение: $b = 0$ при $t' = 0$.

Отсюда можно определить ценность дезинформации. Ценность дезинформации не может быть положительна, поскольку в таком случае возникает противоречие в рамках значения дезинформации. Также, как было сказано ранее, существует неактуальная информация, ценность которой равна нулю. Поскольку влияние такого вида информации не определено, то в дальнейшем будем относить ее к дезинформации, так как ее значение противоречит с реальными данными. Чем дольше дезинформация находится в рамках КФС, тем больше она изменяет затраты, следовательно, ее ценность также снижается во времени. Таким образом ценность дезинформации может быть рассчитана, как:

$$v_{inftei} = val_{\max}(Inf_{ei}^{t}) * \left(\left(a'_{Inf}\right)^{t-(t'-1)} - 1\right). \quad (18)$$

Тогда ценность информации всегда находится в диапазоне $[\, val_{\max}(Inf_{ei}^{t}); 0) \cup \left(0; -val_{\max}(Inf_{ei}^{t})\right)$.

Для того, чтобы определить как именно повлияла полученная информация на цель КФС предлагается следующий подход. Пусть в момент времени t элемент получает информацию. Тогда влияние полученной информации на цель можно определить с помощью изменения скорости прироста затрат. Пусть

необходимо определить влияние информации в момент времени t. Тогда $\exists\delta_c$ – критерий изменения функции затрат, причем:

$$\delta_c = \left(c_{ei}(t)\right)' - \left(c_{ei}(t'-1)\right)', \quad (19)$$

где $t'-1$ – момент времени, предшествующий моменту получения информации.

В результате вычислений по (19) можно заключить вывод о влиянии полученной информации таким образом, что полученная информация Inf_{ei}^{t} является:

– актуальной при $\delta_c \leq 0$;

– менее актуальной при $\delta'_c = \delta_c > 0$;

– дезинформацией при $\delta_c > \delta'_c$.

Отметим, что $\delta_c < 0$ возможно только тогда, когда в момент времени t была получена актуальная информация, а в момент времени $t-1$ элемент оперировал менее актуальной информацией или дезинформацией.

В качестве примера применения разработанного подхода к определению влияния информации на цель предлагается информация, параметры которой определяются следующим образом:

– $val_{\max}(Inf_{ei}^{t}) = 10$,

– $a_{Inf} = 0.7$,

– $a'_{Inf} = 0.3$.

Графики функции ценности информации, строящиеся по данным коэффициентам в промежутке [0;5] представлены на рисунке 3.

Предположим, что в промежуток $t \in [0;2]$, элемент оперировал актуальной информацией, в промежуток $t \in [3;4]$ – менее актуальной, которая появилась в следствие оперирования информацией, полученной в момент времени $t = 2$, в промежуток $t \in [5;6]$ – дезинформацией и $t \in [7;9]$ – снова актуальной информацией. Тогда $t' = 2$ для функции затрат при менее актуальной информации и $t' = 5$ – для дезинформации. В качестве

коэффициента k было взято значение 1. Тогда функции затрат на промежутке [0;9] будут выглядеть как на рисунке 4.

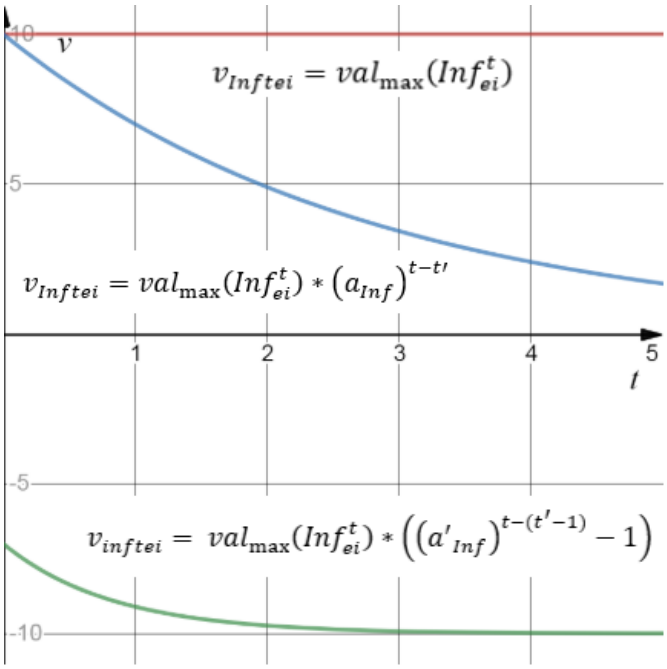

Рисунок 3 – Графики функций ценности актуальной и менее актуальной корректных информаций и дезинформации

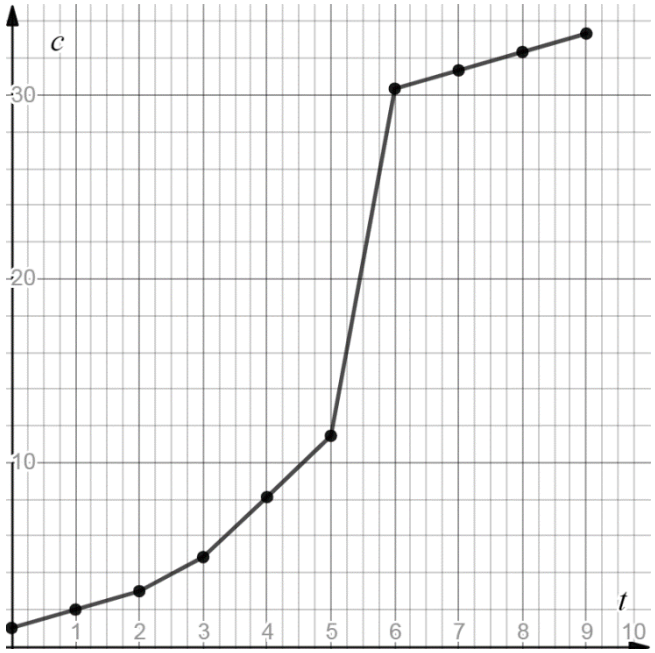

Рисунок 4 – График функции затрат

Для оценки влияния информации будем рассчитывать коэффициент δ_c в последний момент времени оперирования информацией. Тогда δ_c для информации в промежутке $t \in [3; 4]$ равен:

$$\delta_c^1 = -\ln 0.7 * (0.7)^{-2} * 3 + (0.7)^{-2} - 1 = 3.22.$$

Для $t \in [5; 6]$:

$$\delta_c^2 = -\ln 0.3 * (0.3)^{-2} * 2 + (0.3)^{-2} + \ln 0.7 * (0.7)^{-2} * 3 - (0.7)^{-2} = \\ = 8.63.$$

Для $t \in [7; 9]$:

$$\delta_c^3 = 1 + \ln 0.3 * (0.3)^{-2} * 2 - (0.3)^{-2} = -8.21.$$

Анализируя полученные данные видно, что в $t \in [7; 9]$ информация является актуальной, так как $\delta_c^3 \leq 0$, а $\delta_c^3 < 0$ указывает на то, что в промежутке $t \in [5; 6]$ элемент оперировал менее актуальной информацией или дезинформацией. Исходя из того, что $0 < \delta_c^1 < \delta_c^2$ – в промежуток $t \in [3; 4]$ элемент оперировал менее актуальной информацией, а в $t \in [5; 6]$ – дезинформацией.

Стоит отметить, что если бы элемент оперировал дезинформацией только в момент времени $t = 5$, то $\delta_c^1 \approx \delta_c^2$. Поэтому для увеличения точности получаемого значения необходимо рассматривать влияние при $t > t'$.

Для практического применения данного подхода, когда достоверно не известно, какую именно нужно использовать функцию, можно использовать аппроксимацию линейной функции, изменение скорости которой поможет определять δ_c.

Выводы по главе 2

В данной главе был проведен анализ влияния информации на цель КФС. В результате формирования классификации нарушений содержательной целостности были сформированы три вида классификации, применение которых поможет установить характер нарушения для дальнейшего выявления причин возникновения СДИВ.

Основываясь на цели КФС было определено влияние на нарушения на затраты КФС посредством введения понятия ценности информации. Определение параметров максимальной ценности, коэффициента актуальности и коэффициента влияния дезинформации на цель, позволит выявлять повышение затрат КФС при выполнении задач и определять наличие нарушения. Сформированные функции затрат позволят элементам КФС принимать решения о корректности или некорректности информации в случаях неполноты данных, поэтому данные функции будут использованы для описания выигрышей при использовании подходов теории игр.

3 РАЗРАБОТКА МОДЕЛИ ОБНАРУЖЕНИЯ НАРУШЕНИЙ СОДЕРЖАТЕЛЬНОЙ ЦЕЛОСТНОСТИ В КИБЕР-ФИЗИЧЕСКОЙ СИСТЕМЕ С ИСПОЛЬЗОВАНИЕМ ТЕОРИИ ИГР

3.1 Адаптация метода обнаружения нарушений содержательной целостности

В рамках модели КФС метод, описанный в 1.3, может быть применим для видов информации, описанных в 1.2.1. Под информацией S, по которой производится оценка поведения элемента-объекта e_j элементом-субъектом e_i, далее будем понимать I_{eiej} – информацию, получаемою в процессе ИВ между e_i и e_j. Согласно разработанной классификации нарушений содержательной целостности к пассивным знаниям KS_{eipas} элемента e_i относятся информация об окружающей среде E_{ei} и E_{eiej} и иная информация O_{ei} и O_{eiej}. Таким образом в дальнейшем для формул (5)-(7) будет пониматься $S = I_{eiej}$, где s_l принадлежит только к одному из множеств среди TS_{eiej}, S_{eiej}, E_{eiej}, O_{eiej}.

3.2. Метод формирования показателя истинности в случае неполноты данных

3.2.1. Формулировка игры

Для формирования показателя $Truth$, в случаях неполноты данных, будем рассматривать процесс получения информации как игру с двумя лицами в нормальной форме, где у каждого лица конечное число возможных стратегий. Согласно [21] игру можно охарактеризовать, как:

– дискретную – множество стратегий дискретно;

– конечную – множество стратегий конечно;

– стратегическую – неопределенность происходит от другого игрока;

– в нормальной форме – существует матрица платежей;

– антагонистическую – проигрыш одного игрока равен выигрышу другого.

Тогда определим игру Г, согласно определению антагонистической игры в нормальной форме, данному в работе [20]:

$$\Gamma = (X, Y, K),$$

где X, Y – множества стратегий игроков 1 и 2 соответственно, а $K: X \times Y \to \mathbb{R}$ – функция выигрыша игрока 1.

В данном случае под игроком 1 далее будем понимать элемент e_T, принимающий информацию, который является доверенным, а под игроком 2 – элемент e_U, передающий информацию, который возможно является диверсантом.

У e_T и e_U существует по две стратегии $x_i \in X, i \geq 1$ и $y_j \in Y, j \geq 1$, описанные естественным языком в таблице 1.

Таблица 1 – Стратегии игроков

l	x_l	y_l
1	Оценить информацию как корректную	Передать дезинформацию
2	Оценить информацию как некорректную	Передать достоверную информацию

3.2.2 Решение игры в чистых стратегиях

Сформируем матрицу платежей. Для этого определим функцию выигрыша $K(x_i, y_j)$ элемента e_T следующим образом. Пусть существует функция расчета скорости прироста затрат, зависящая от выбранных элементами стратегий $H(x_i, y_i)$. Из смысла исходов стратегий из таблицы 1 информация, которой будет оперировать элемент e_T является актуальной при $(x_1; y_2)$, менее актуальной при $(x_2; y_1)$ и $(x_2; y_2)$ и дезинформацией при $(x_1; y_1)$. В данном случае для дезинформации $t' = t$, тогда, определяя производную первой степени функций (15)-(17) получим:

$$H(x_i, y_j) = \begin{cases} k * \left(a'_{Inf}\right)^{-1}(-\ln a'_{Inf} + 1), i = j = 1 \\ k,\ i = 1, j = 2 \\ k * \left(a_{Inf}\right)^{t'-t}(-\ln a_{Inf} * (t - (t' - 1)) + 1),\ \forall j, i = 2 \end{cases}. \quad (20)$$

Пусть существует функция определения оптимальной стратегии элемента e_T, зависящая от выбранной элементом e_U стратегии, $opt(y_j)$, тогда:

$$opt(y_j) = \begin{cases} x_1, & j = 2 \\ x_2, & j = 1 \end{cases}. \quad (21)$$

Тогда функцию выигрыша $K(x_i, y_j)$ можно определить, как разницу между скоростью прироста затрат, которая должна быть в случае полноты данных – когда элемент e_T знает стратегию e_U – определяемую с помощью (20) и (21) и скоростью прироста затрат, зависящую от выбранных элементами стратегий, определяемую согласно (20):

$$K(x_i, y_j) = H(opt(y_j), y_j) - H(x_i, y_j). \quad (22)$$

Пусть $H(x_1, y_1) = H_{inc}$, а $H(x_2, y_j) = H_{Cor}$. Тогда, основываясь на (22), значения $K(x_i, y_j)$ для всех стратегий приведены в таблице 2.

Таблица 2 – Выигрыши игрока e_T

$\boldsymbol{i}$	$\boldsymbol{j}$	$\boldsymbol{H(opt(y_j), y_j)}$	$\boldsymbol{H(x_i, y_j)}$	$\boldsymbol{K(x_i, y_j)}$
1	1	H_{cor}	H_{inc}	$H_{cor} - H_{inc}$
1	2	k	k	0
2	1	H_{cor}	H_{cor}	0
2	2	k	H_{cor}	$k - H_{cor}$

Значения $K(x_i, y_j)$, определенные в таблице 2, являются выигрышами игрока e_T, тогда матрицу платежей можно представить согласно таблице 3.

Для дальнейших операций упростим матрицу платежей разделив выигрыши на k. Учитывая, что $H'_{cor} = \frac{H_{cor}}{k}$ и $H'_{inc} = \frac{H_{inc}}{k}$, получим матрицу A вида:

$$A = \begin{pmatrix} H'_{cor} - H'_{inc} & 0 \\ 0 & 1 - H'_{cor} \end{pmatrix}, \quad (23)$$

где $a_{i-1j-1} = K(x_i, y_j)$.

Таблица 3 – Матрица платежей игры

		$\boldsymbol{e_U}$	
		$\boldsymbol{y_1}$	$\boldsymbol{y_2}$
$\boldsymbol{e_T}$	$\boldsymbol{x_1}$	$H_{cor} - H_{inc}$	0
	$\boldsymbol{x_2}$	0	$k - H_{cor}$

Найдем цену данной игры. Для этого воспользуемся определением максимина как нижней цены игры и минимакса как верхней цены игры, согласно [20].

Границы игры будут рассчитываться по следующим формулам:

$$\underline{v_\Gamma} = \max_{i=\{1,2\}} \min_{j=\{1,2\}} K(x_i, y_j), \quad (24)$$

$$\overline{v_\Gamma} = \min_{j=\{1,2\}} \max_{i=\{1,2\}} K(x_i, y_j), \quad (25)$$

где $\underline{v_\Gamma}$ – нижняя граница цены игры, $\overline{v_\Gamma}$ – верхняя, при этом $\underline{v_\Gamma} \leq \overline{v_\Gamma}$.

Тогда игра Г разрешима в чистых стратегиях только тогда, когда:

$$v_\Gamma = \underline{v_\Gamma} = \overline{v_\Gamma}, \quad (26)$$

где v_Γ – цена игры Г.

Тождество (26) объясняется наличием ситуаций равновесия (x^*, y^*), при котором:

$$\begin{cases} K(x^*, y^*) \geq K(x, y^*) \\ K(x^*, y^*) \leq K(x^*, y) \end{cases}.$$

То есть элемент в матрице платежей является одновременно наибольшим в своем столбце и наименьшим в своей строке. Только в таком случае ситуация (x^*, y^*) является оптимальной.

Определим разрешимость игры в чистых стратегиях, используя (24) и (25).

Из (23) видно, что $\overline{v_\Gamma} = 0$, а $\underline{v_\Gamma} = \max\big((H'_{cor} - H'_{inc}), (1 - H'_{cor})\big)$ не может быть определена в общем виде. Основываясь на ограничениях, сформулированных для (16) и (17), $\overline{v_\Gamma} \neq \underline{v_\Gamma}$ и тождество (26) не выполнимо, а значит игра Г не может быть решена в чистых стратегиях.

Согласно теореме фон Неймана [22] всякая матричная игра имеет ситуацию равновесия в смешанных стратегиях. Следовательно, необходимо найти решение в смешанных стратегиях.

3.2.3 Решение игры в смешанных стратегиях

Определим понятие смешанных стратегий элементов.

Пусть $\exists \xi_i: 0 \leq \xi_i \leq 1, \sum_{i=1}^{|X|} \xi_i = 1, 1 \leq i \leq |X|$ – вероятность выбора чистой стратегии x_i и $\exists \eta_j: 0 \leq \eta_j \leq 1$, $\sum_{y=1}^{|Y|} \eta_j = 1, 1 \leq j \leq |Y|$ – вероятность выбора чистой стратегии y_j. Тогда $\bar{x} = \big(\xi_1, \dots, \xi_{|X|}\big)$ и $\bar{y} = \big(\eta_1, \dots, \eta_{|Y|}\big)$ – смешанные стратегии элементов e_T и e_U соответственно.

Для смешанных стратегий выигрыш в ситуации $(\bar{x}, \bar{y})$ будет рассчитываться следующим образом:

$$v_\Gamma = K(\bar{x}, \bar{y}) = \sum_{i=1}^{|X|} \sum_{j=1}^{|Y|} a_{ij} \xi_i \eta_j, \tag{27}$$

где a_{ij} – элемент из матрицы (23).

Тогда, при использовании одним из элементов чистой стратегии x_i или y_j, (26) можно представить как:

$$K(x_i, \bar{y}) = \sum_{j=1}^{|Y|} a_{ij} \eta_j, \tag{28}$$

$$K\big(\bar{x}, y_j\big) = \sum_{i=1}^{|X|} a_{ij} \xi_i. \tag{29}$$

Тогда, используя (28) и (29), можно определит смешанные стратегии $\bar{x}$ и $\bar{y}$ элементов e_T и e_U соответственно, используя следующие системы уравнений:

$$\begin{cases} a_{00}\xi_1 + a_{10}\xi_2 = v_\Gamma \\ a_{01}\xi_1 + a_{11}\xi_2 = v_\Gamma \\ \xi_1 + \xi_2 = 1 \end{cases}, (30)$$

$$\begin{cases} a_{00}\eta_1 + a_{01}\eta_2 = v_\Gamma \\ a_{10}\eta_1 + a_{11}\eta_2 = v_\Gamma \\ \eta_1 + \eta_2 = 1 \end{cases}, (31)$$

Определим $\bar{x}$, используя (31):

$$\begin{cases} (H'_{cor} - H'_{inc})\xi_1 = v_\Gamma \\ (1 - H'_{cor})\xi_2 = v_\Gamma \\ \xi_1 + \xi_2 = 1 \end{cases} \Rightarrow \begin{cases} (H'_{cor} - H'_{inc})\xi_1 = (1 - H'_{cor})\xi_2 \\ \xi_2 = 1 - \xi_1 \end{cases} \Rightarrow$$

$$\Rightarrow (H'_{cor} - H'_{inc})\xi_1 = (1 - H'_{cor}) - (1 - H'_{cor})\xi_1 \Rightarrow \xi_1 = \frac{1-H'_{cor}}{1-H'_{inc}} \Rightarrow$$

$$\Rightarrow \xi_2 = \frac{H'_{cor}-H'_{inc}}{1-H'_{inc}}.$$

Следовательно, оптимальная смешанная стратегия элемента e_T $\bar{x} = \left(\frac{1-H'_{cor}}{1-H'_{inc}}, \frac{H'_{cor}-H'_{inc}}{1-H'_{inc}}\right)$.

Найдем оптимальную смешанную стратегию элемента e_U, основываясь на (31):

$$\begin{cases} (H'_{cor} - H'_{inc})\eta_1 = v_\Gamma \\ (1 - H'_{cor})\eta_2 = v_\Gamma \\ \eta_1 + \eta_2 = 1 \end{cases} \Rightarrow \begin{cases} (H'_{cor} - H'_{inc})\eta_1 = (1 - H'_{cor})\eta_2 \\ \eta_2 = 1 - \eta_1 \end{cases} \Rightarrow$$

$$\Rightarrow (H'_{cor} - H'_{inc})\eta_1 = (1 - H'_{cor}) - (1 - H'_{cor})\eta_1 \Rightarrow \eta_1 = \frac{1-H'_{cor}}{1-H'_{inc}} \Rightarrow$$

$$\Rightarrow \eta_2 = \frac{H'_{cor}-H'_{inc}}{1-H'_{inc}}.$$

Следовательно, оптимальная смешанная стратегия элемента e_U $\bar{y} = \left(\frac{1-H'_{cor}}{1-H'_{inc}}, \frac{H'_{cor}-H'_{inc}}{1-H'_{inc}}\right)$.

Тогда цена игры, определяемая по (27), равна:

$$v_\Gamma = (H'_{cor} - H'_{inc})\left(\frac{1-H'_{cor}}{1-H'_{inc}}\right)^2 + (1 - H'_{cor})\left(\frac{H'_{cor}-H'_{inc}}{1-H'_{inc}}\right)^2 =$$
$$= \frac{(1-H'_{cor})(H'_{cor}-H'_{inc})}{1-H'_{inc}}.$$

Теперь, основываясь на найденных оптимальных смешанных стратегиях, определим порядок формирования показателя $Truth$.

3.2.4 Формирование показателя истинности на основе исхода игры

Опишем подход к формированию $Truth$ в случае неполноты данных, основываясь на оптимальных смешанных стратегиях $\bar{x}$ и $\bar{y}$.

Пусть s_l – блок информации, который передает элемент e_U элементу e_T в момент времени t. Тогда необходимо определить $Truth^{sl}_{eTeU}$.

Учитывая описания чистых стратегий x_i элемента e_T (таблица 1), проведем параллель с определением показателя $Truth$. Так как данный показатель не что иное, как субъективная оценка истинности информации, то показатель можно формировать основываясь на полученной оптимальной смешанной стратегии так, что вероятность выбора стратегии доверия x_1 напрямую влияет на формируемый показатель $Truth$, следовательно:

$$Truth^{sl}_{eTeU} = \xi_1 = \frac{1-H'_{cor}}{1-H'_{inc}}$$

или

$$Truth^{sl}_{eTeU} = \frac{1-\left(a_{Inf}\right)^{t'-t}\left(-\ln a_{Inf}*(t-(t'-1)\right)+1)}{1-\left(a'_{Inf}\right)^{-1}(-\ln a'_{Inf}+1)}. \quad (32)$$

Таким образом формирование показателя $Truth$ для блока информации, с учетом (32) и (8), производится по следующей формуле:

$$Truth_{eiej}^{sl} = \begin{cases} \begin{cases} 1, s_l \text{ корректен} \\ 0, s_l \text{ некорректен} \end{cases}, e_i \text{ обладает } Inf \in KN_{eipas} \\ \frac{\sum_{k=1}^{ntruth} Truth_{ekej}^{sl}}{n_{truth}}, e_i \text{ не обладает } Inf \in KN_{eipas} \text{ и } n_{truth} \neq 0 \\ \frac{1-(a_{Inf})^{t'-t}\left(-\ln a_{Inf}*(t-(t'-1))\right)+1)}{1-(a'_{Inf})^{-1}(-\ln a'_{Inf}+1)}, e_i \text{ не обладает } Inf \in KN_{eipas} \text{ и } n_{truth} = 0 \end{cases} .$$

Поскольку (32) может быть больше 1, что не попадает под допущения, оговоренные в разделе 1.3, то при $\frac{1-(a_{Inf})^{t'-t}\left(-\ln a_{Inf}*(t-(t'-1))\right)+1)}{1-(a'_{Inf})^{-1}(-\ln a'_{Inf}+1)} > 1$ показатель истинности принимается за 1.

3.3 Оценка эффективности разработанной модели

В рамках серии экспериментов областью применения модели КФС является управление беспилотным транспортом. Целью такой системы является максимизация пропускной способности элементов на пересечении дорог (перекрестке) или, формулируя цель в контексте формулы (4), использование минимальных временных ресурсов при проезде максимального количества элементов, где элементы являются беспилотными транспортными средствами.

Под перекрестком понимается пересечение двух сторонних дорог, расположенных на модели участка городской инфраструктуры размером 10x10 клеток, где размер перекрестка 2x2 клетки (рисунок 5).

В рамках экспериментов элементам присущи следующие характеристики:

– скорость движения элементов – 1 клетка за 1 дискрету времени;

– радиус действия устройств анализа окружающей среды – 1 клетка – то есть совокупно элемент «видит» 8 клеток вокруг себя;

– местоположение элемента в рамках модели участка городской инфраструктуры.

Каждый элемент решает задачу по достижению конкретного местоположения. Так как траектории движения элементов могут пересекаться,

то необходимо обеспечить безопасный разъезд в рамках перекрестка. В рамках симуляции все элементы, появляющиеся на границе участка городской инфраструктуры, обмениваются с другими существующими элементами, определяют возможные конфликты и перестраивают свои маршруты с учетом максимизации пропускной способности. Для достижения цели КФС элементам необходимо определять отклонения в маршруте для выявления диверсанта, намеренно передавшего неверный путь или сбившегося с пути из-за нарушения функционирования. Для этого элементы должны обмениваться информацией о своем местоположении и сравнивать ее с показаниями устройств. В рамках экспериментов вводится допущение о том, что элементы способны различать и однозначно идентифицировать другие элементы с помощью устройств анализа окружающей среды.

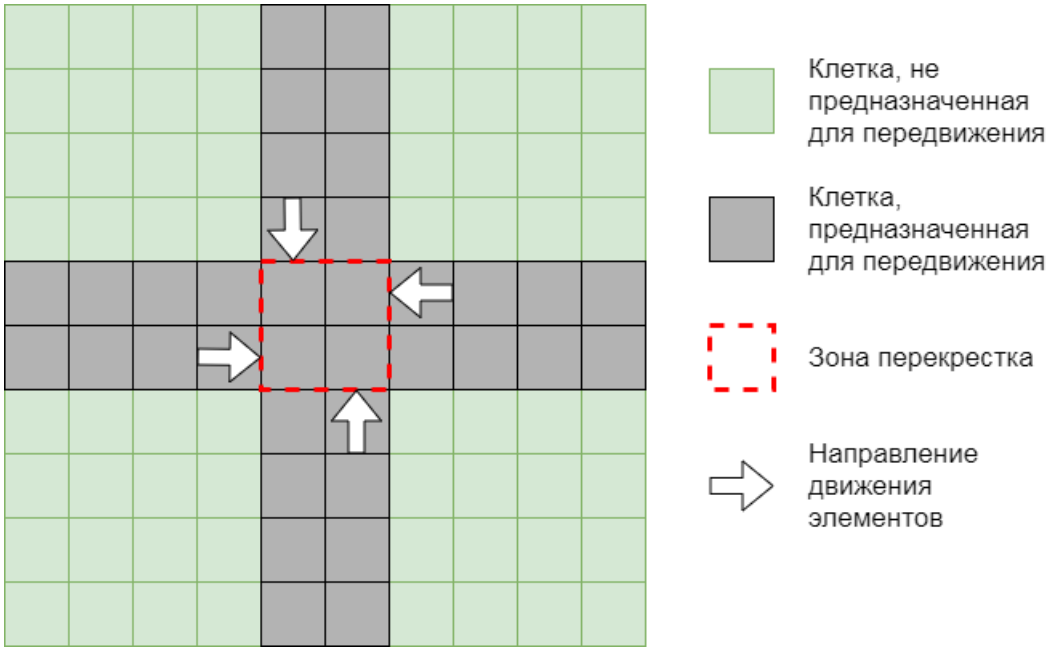

Рисунок 5 – Модель участка городской инфраструктуры

Для проведения экспериментов также были сформированы следующие условия:

– в каждом эксперименте генерируется 1000 элементов на границах участка городской инфраструктуры в направлении перекрестка;

– за 1 итерацию эксперимента может быть сгенерировано от 0 до 4 элементов с вероятностями 20%;

– в каждом эксперименте присутствует от 10% до 40% диверсантов от общего количества элементов.

Эксперименты разделены на 2 группы: классический метод обнаружения нарушений содержательной целостности и метод обнаружения нарушений с использованием теории игр.

Каждая группа разделена на подгруппы, основываясь на количестве диверсантов в КФС: 10%, 20%, 30%, 40% от общего количества элементов. Каждая подгруппа является серией из 1000 экспериментов.

Для описания поведения диверсанта была использована концепция on-off атаки [23]. При проведении таких атак диверсант находится либо в состоянии on, либо в состоянии off. В первом случае диверсант старается максимально нарушить функционирование КФС, но, так как данное нарушение легко выявить при его длительном существовании, диверсанту необходимо прекращать атаку на время, чтобы не быть исключенным из ИВ. Для этого существует off состояние, когда диверсант ведет себя как исправный элемент. В рамках проведения подобной атаки время состояния on должно быть больше времени состояния off для максимизации ущерба, наносимого диверсантом системе, за минимальное время.

В рамках эксперимента элементы обмениваются информацией о местоположении. Поскольку местоположение при динамическом характере элемента изменяется часто, то и актуальность данной информации должна снижаться как можно быстрее. Будем считать, что актуальность такого вида информации снижается в 2 раза относительно предыдущего момента времени, а влияние дезинформации такого вида на цель повышает затраты в 5 раз. Тогда $a_{Inf} = 0.5$ и $a'_{Inf} = 0.2$. Принимая за ценность информации $val_{max}(Inf) = 1$, графики ценности актуальной, менее актуальной информации и дезинформации будут выглядеть как на рисунке 6.

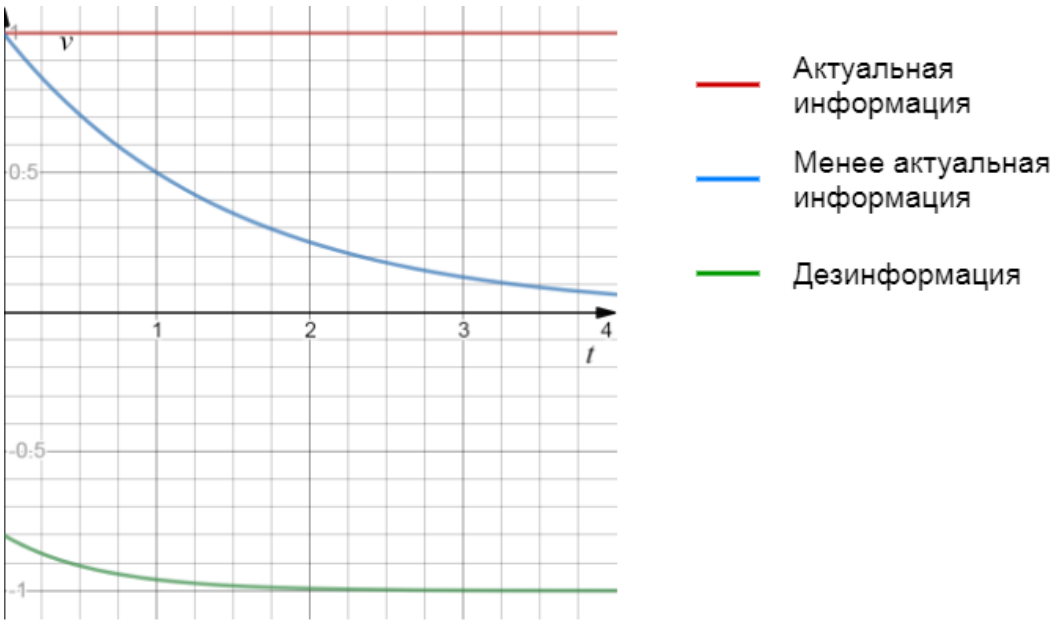

Рисунок 6 – Графики функции ценности актуальной информации, менее актуальной информации и дезинформации

График функции показателя истинности, рассчитываемого по формуле (32), от разницы между моментом получения информации и настоящим моментом времени для $a_{Inf} = 0.5$ и $a'_{Inf} = 0.2$ представлен на рисунке 7.

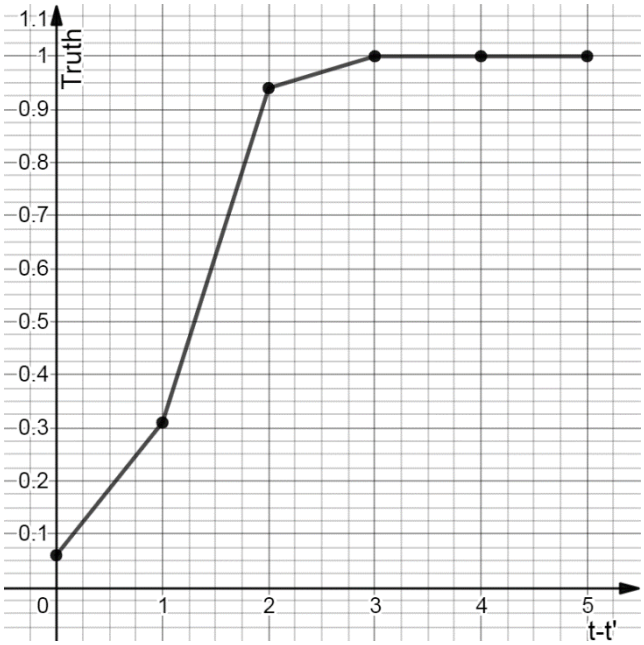

Рисунок 7 – График функции расчета показателя истинности в случае неполноты данных

Как видно из рисунка 7 элементу-субъекту выгодно оценивать информацию как корректную при оперировании менее актуальной информацией более 2 дискрет времени.

В рамках данного симулятора поведение элемента однозначно можно определить как некорректное при $R_{min} = 0$ и $Truth_{min} = 0$, а как корректное при $R_{max} = 1$ и $Truth_{max} = 1$. Основываясь на формуле (10), расчет показателя доверия будет производиться по формуле:

$$Trust_{eiejt} = \sqrt{\left(R_{eiejt-1}\right)^2 + \left(Truth^S_{eiejt}\right)^2} - \\ - \sqrt{\left(1 - R_{eiejt-1}\right)^2 + \left(1 - Truth^S_{eiejt}\right)^2}.$$

Для определения элемента как диверсанта устанавливается пороговое значение $\alpha_{trust} = 0$, поскольку увеличение или уменьшение значения приведет к росту ошибок в определении корректного и некорректного поведения [6].

Вышеописанные условия были реализованы на языке программирования *Python*. Исходный код реализованного симулятора представлен в Приложении А.

Для оценки точности модели определим нулевую гипотезу о существовании нарушения и альтернативную гипотезу об отсутствии нарушения. Тогда могут существовать ошибки первого и второго рода, где ошибка первого рода – неверное отрицание нулевой гипотезы, а ошибка второго рода – неверное отрицание альтернативной гипотезы. Отсюда следует, что для оценки модели необходимо оценить следующие показатели:

– TP – процент верно заблокированных диверсантов от всех элементов;

– TN – процент верно незаблокированных доверенных элементов от всех элементов;

– FP – процент неверно заблокированных доверенных элементов от всех элементов;

– FN – процент неверно незаблокированных диверсантов от всех элементов.

Для сравнения эффективности предложенного решения с классическим методом репутации и доверия будут рассчитаны следующие метрики:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN},$$

$$Precision = \frac{TP}{TP+FP},$$

$$Recall = \frac{TP}{TP+FN},$$

где $Accuracy$ – доля правильно подтвержденных гипотез, $Precision$ – доля верно определенных диверсантов по отношению ко всем элементам, определенным как диверсанты, $Recall$ – доля верно определенных диверсантов по отношению ко всем элементам, которые являются диверсантами.

Для оценки точности разработанной модели и точности метода репутации и доверия будет рассчитана метрика F-мера, которая объединяет метрики $Precision$ и $Recall$ и позволяет найти среднее гармоническое значение между данными метриками:

$$F_\beta = (\beta^2 + 1)\frac{Precision*Recall}{\beta^2*Precision+Recall},$$

где β – коэффициент, определяющий влияние метрик $Precision$ и $Recall$ при формировании F-меры. Так, если $0 < \beta < 1$, то в рамках разработанной модели метрика $Precision$ важнее $Recall$. Если $\beta = 1$, то в модели одинаково важны обе метрики. Иначе, при $\beta > 1$, метрика $Recall$ важнее $Precision$.

Результаты экспериментов представлены в таблице 4. Как видно из результатов, доля верно определенных разработанной моделью гипотез выше в среднем на 5% относительно метода репутации и доверия, а доля диверсантов, классифицированных как элементы с корректным поведением, ниже на в среднем на 5%. Также было получено снижение ошибки 1-го рода в среднем на 15%. Для дальнейшего анализ необходимо более подробно сравнить метрики $Precision$ и $Recall$, а также F-меру. Для анализа F-меры были выбранные значения β равные 0.5, 1 и 2, где $\beta = 0.5$ можно трактовать как $Precision$ в 2 раза важнее $Recall$, а $\beta = 2$ – $Recall$ в 2 раза важнее $Precision$.

Таблица 4 – Результаты оценки эффективности разработанной модели

Доля нарушителей, %	Модель (метод)	TP	FP	TN	FN	$Accuracy$	$Precision$	$Recall$	$F_{0.5}$	F_1	F_2	Ошибка	
												первого рода	второго рода
10	Разработанная модель	0.13	0.47	0.39	0.01	0.52	0.21	0.94	0.25	0.35	0.56	0.06	0.55
	Метод репутации и доверия	0.11	0.28	0.59	0.03	0.70	0.28	0.79	0.32	0.41	0.58	0.21	0.32
20	Разработанная модель	0.25	0.40	0.33	0.02	0.58	0.39	0.94	0.44	0.55	0.73	0.06	0.55
	Метод репутации и доверия	0.21	0.23	0.50	0.06	0.71	0.48	0.79	0.52	0.59	0.70	0.21	0.32
30	Разработанная модель	0.37	0.33	0.28	0.02	0.64	0.52	0.94	0.58	0.67	0.81	0.06	0.55
	Метод репутации и доверия	0.31	0.19	0.41	0.08	0.72	0.61	0.79	0.64	0.69	0.75	0.21	0.32
40	Разработанная модель	0.38	0.32	0.27	0.02	0.65	0.54	0.94	0.59	0.69	0.82	0.06	0.55
	Метод репутации и доверия	0.32	0.19	0.40	0.09	0.73	0.63	0.79	0.66	0.70	0.75	0.21	0.32
10-40 (средние значения)	Разработанная модель	0.28	0.38	0.32	0.02	0.60	0.42	0.94	0.48	0.59	0.76	0.06	0.55
	Метод репутации и доверия	0.24	0.22	0.48	0.06	0.71	0.51	0.79	0.55	0.62	0.71	0.21	0.32

Сравнение метрик $Precision$, $Recall$, $F_{0.5}$, F_1 и F_2 представлено на рисунках 8-12.

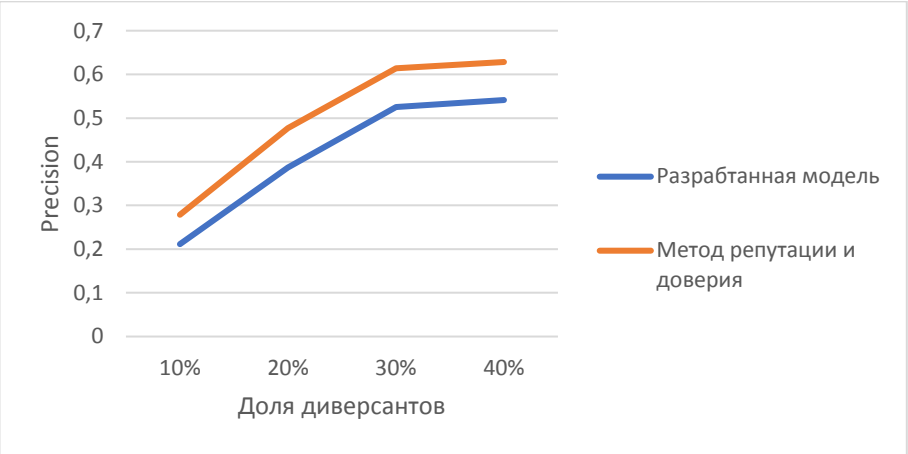

Рисунок 8 – Метрика $Precision$

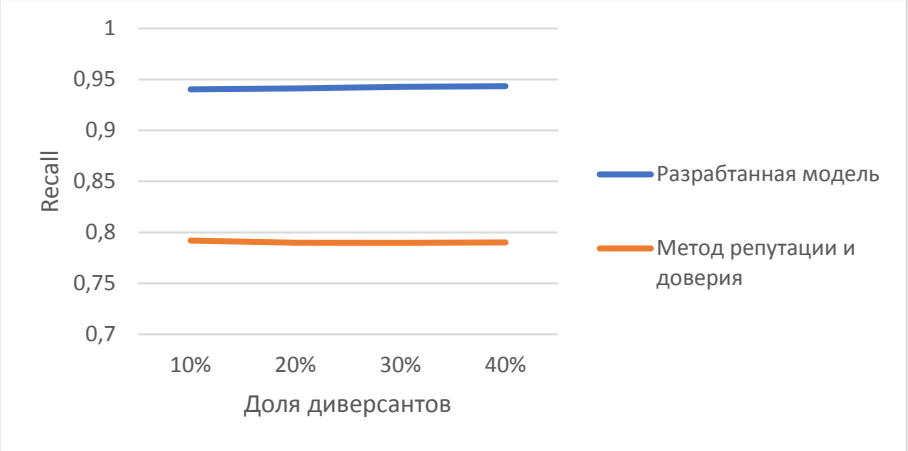

Рисунок 9 – Метрика $Recall$

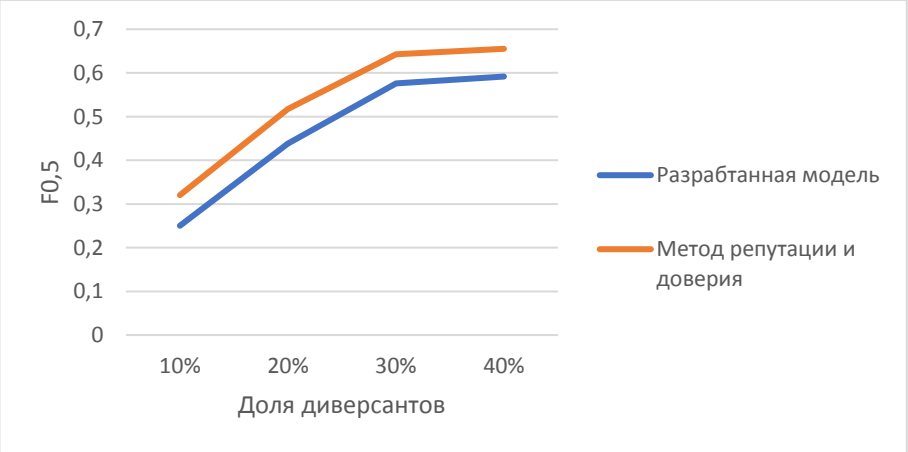

Рисунок 10 – Метрика $F_{0.5}$

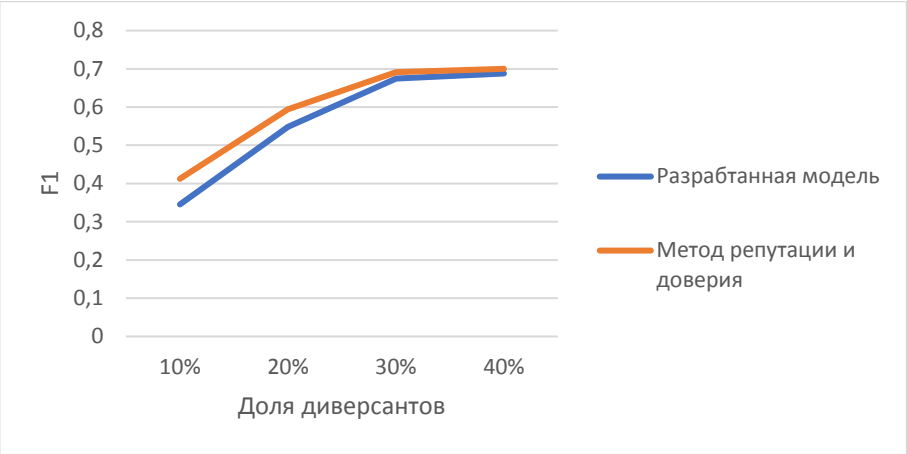

Рисунок 11 – Метрика F_1

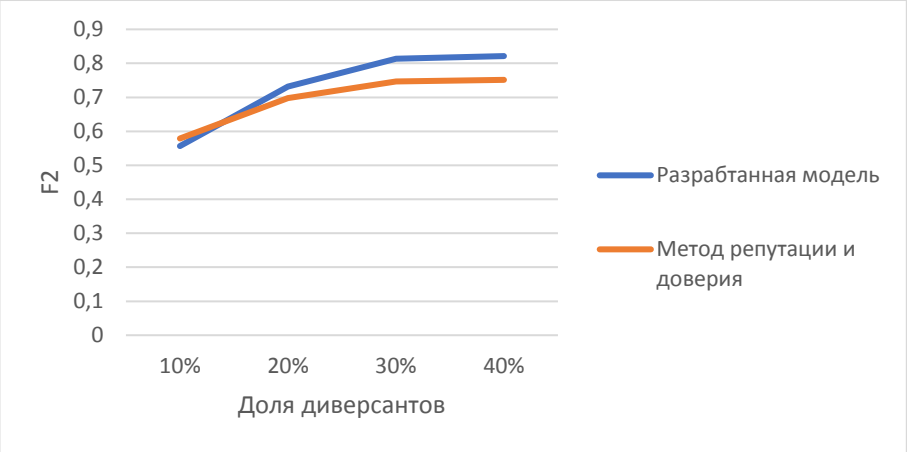

Рисунок 12 – Метрика F_2

Анализ результатов показывает, что несмотря на снижение доли верно определенных диверсантов относительно элементов, классифицированных как диверсанты, доля верно определяемых диверсантов относительно всех диверсантов выросла примерно на 15% и, при этом, не зависит от доли диверсантов относительно всех элементов в КФС.

Снижение метрики $Precision$ на 9% компенсируется повышением точности в определении всех существующих диверсантов на 15%. Следовательно, если говорить о критически важной инфраструктуре, то разработанная модель эффективнее метода репутации и доверия, что показывает метрика F_2, поскольку в подобной инфраструктуре рост ошибки 1-го рода нанесет больше ущерба, чем рост ошибок 2-го рода. При этом, разработанная модель недостаточно эффективна в случаях, когда необходимо свести ошибку 2-го рода к минимуму.

Выводы по главе 3

В рамках данной главы был разработан подход к формированию показателя истинности в случае неполноты данных. Процесс оценки был представлен в терминах теории игр, поэтому была введена классификация игры, как антагонистической игры двух лиц в нормальной форме. В рамках игры были обозначены стратегии элемента-субъекта и элемента-объекта оценки и подход к формированию выигрышей элемента-субъекта. В рамках данной игры выигрышем является разница между скоростью изменения затрат элемента-субъекта при выборе оптимальной стратегии, то есть когда элемент знает о стратегии элемента-объекта, и при выборе стратегии независимо от элемента-объекта. На основе этого была сформирована матрица платежей и представлено решение игры в чистых и в смешанных стратегиях. Решение игры в чистых стратегиях не привело к нахождению ситуации равновесия, поэтому были определены смешанные стратегии элементов. Нахождение смешанных стратегий – вероятностей выбора той или иной чистой стратегии для получения ситуации равновесия – позволило сформировать показатель истинности в случае неполноты данных. Так как одной из стратегий элемента-субъекта является оценка информации как корректной, то показатель истинности приравнивается к вероятности выбора данной стратегии исходя из смысла показателя истинности.

Для подтверждения эффективности разработанного подхода были проведены серии экспериментов, в которых метод репутации и доверия был применен к кибер-физической системе, используемой в области транспорта. В результате был получен прирост доли верно определенных диверсантов относительно всех диверсантов на 15%, что говорит об эффективности метода в случаях, когда ошибка 1-го рода критичнее ошибки 2-го рода.

ЗАКЛЮЧЕНИЕ

В данной работе была разработана модель обнаружения нарушений содержательной целостности в кибер-физических системах с использованием теории игр. В рамках адаптированной модели кибер-физической системы было выявлено, что существует ситуация неполноты данных, при которой элемент-субъект оценки не может определить субъективную оценку информации. Для этого был сформирован подход по формированию данного показателя в терминах теории. Решение игры в смешанных стратегиях позволило сформировать показатель в случае неполноты данных исходя из вероятности оценки элементом-субъектом информации как корректной.

Для проверки эффективности был разработан симулятор, реализующий модель кибер-физической системы в области беспилотного транспорта. Проведение серии экспериментов для метода доверия и репутации без использования разработанного подхода и с его использованием показало увеличение точности в выявлении некорректного поведения среди существующих диверсантов на 15%.

В дальнейшем планируется проведение экспериментов в рамках разработанной группы моделей беспилотных транспортных средств для оценки эффективности модели в реальных условиях. Также анализ результатов показал, что метрика $Precision$ увеличивается при увеличении доли диверсантов относительно всех элементов в системе, что позволяет предполагать о повышении точности в верном определении диверсантов относительно элементов, определенных как диверсанты, при доле диверсантов больше 50%, что является еще одним научным заделом для будущих исследований.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Griffor E.R., Greer C., Wollman D.A., Burns M.J. Framework for cyber-physical systems: Volume 1, overview [Электронный ресурс] // Special Publication (NIST SP)-1500-201. – 2017. URL: https://doi.org/10.6028/NIST.SP.1500-201 (дата обращения 16.02.20).

2. Зикратов И.А., Зикратова Т.В., Лебедев И.С., Гуртов А.В. Построение модели доверия и репутации к объектам мультиагентных робототехнических систем с децентрализованным управлением // Научно-технический вестник информационных технологий, механики и оптики. – 2014. – N 3 (91). – С. 30 – 38.

3. Юрьева Р.А., Комаров И.И., Дородников Н.А. Построение модели нарушителя информационной безопасности для мультиагентной робототехнической системы с децентрализованным управлением // Программные системы и вычислительные методы. – 2016. – N 1 (14). – С. 42 – 48.

4. ISO/IEC 27000:2018 Information technology – Security techniques – Information security management systems – Overview and vocabulary. – 2018. URL: https://www.iso.org/standard/75281.html (дата обращения: 16.02.20).

5. Shannon C.E., Weawer W. The mathematical theory of communication. – Urbana, USA: University of Illinois Press. – 1963. – 119 с.

6. Викснин И.И. Модели и методы обнаружения нарушений целостности информации в группах беспилотных транспортных средств: диссертация ... канд. тех. наук: 05.13.19 / Викснин Илья Игоревич. – СПб., 2018. – 207 с.

7. Тарасов И.В. Индустрия 4.0: понятие, концепции, тенденции развития [Электронный ресурс] // Стратегии бизнеса. – 2018. – N 6 (50). – С. 57 – 63. URL: https://doi.org/10.17747/2311-7184-2018-5-43-49 (дата обращения: 04.03.20).

8. Юдина М.А. Индустрия 4.0: перспективы и вызовы для общества [Электронный ресурс] // Государственное управление. Электронный вестник. – 2017. N 60. С. 197 – 216. URL: http://e-

journal.spa.msu.ru/uploads/vestnik/2017/vipusk__60._fevral_2017_g./problemi_upravlenija_teorija_i_praktika/yudina.pdf (дата обращения: 04.03.20).

9. Каляев И.А., Гайдук А.Р., Капустян С.Г. Модели и алгоритмы коллективного управления в группах роботов. – М.: Физмалит, 2009. – 280 с.

10. Прангишвили И.В. Энтропийные и другие системные закономерности: Вопросы управления сложными системами. – М.: Наука, 2003. – 428 с.

11. Новиков Д.А. Теория управления организационными системами. М.: МПСИ, 2005. – 584 с.

12. Городецкий В.И. Самоорганизация и многоагентные системы. I. Модели многоагентной самоорганизации // Известия Российской академии наук. Теория и системы управления. – 2012. – N 2. – С. 92 – 120.

13. Li H., Lai L., Poor H.V. Multicast routing for decentralized control of cyber physical systems with an application in smart grid // IEEE Journal on Selected Areas in Communications. – 2012. – Vol. 30, N 6. – P. 1097 – 1107.

14. Guo J., Chen I. A classification of trust computation models for service-oriented internet of things systems // 2015 IEEE International Conference on Services Computing. – 2015. – P. 324 – 331.

15. Bankovic Z., Vallejo J.C., Fraga D., Moya J.M. Detecting false testimonies in reputation systems using self-organizing maps // Logic Journal of the IGPL. – 2013. – Vol. 21, N 4. – P. 549 – 559.

16. Li W., Song H., Zeng F. Policy-based secure and trustworthy sensing for internet of things in smart cities // IEEE Internet of Things Journal. – 2017. – Vol. 5, N 2. – P. 716 – 723.

17. Rawat D.B., Yan G., Bista B.B., Weigle M.C. Trust on the security of wireless vehicular ad-hoc networking // Ad Hoc & Sensor Wireless Networks. – 2015. – Vol. 24, N 3-4. – P. 283 – 305.

18. Зикратов И.А., Зикратова Т.В., Лебедев И.С. Доверительная модель информационной безопасности мультиагентных робототехнических систем с децентрализованным управлением // Научно-технический вестник

информационных технологий, механики и оптики. – 2014. – N 2 (90). – С. 47 – 52.

19. Marinenkov E., Chuprov S., Viksnin I., Kim I. Empirical study on trust, reputation, and game theory approach to secure communication in a group of unmanned vehicles // CEUR Workshop Proceedings – 2020. Vol. 2590. P. 1 – 12.

20. Петросян Л.А., Зенкевич Н.А., Шевкопляс Е.В. Теория игр. – 2-е изд., перераб. и доп. – СПб.: БХВ-Петербург, 2017. – 432 с.

21. Кузин Л.Т. Основы кибернетики: В 2-х т. Т. 2. Основы кибернетических моделей. – М.: Энергия, 1979. – 584 с.

22. фон Нейман Дж., Моргенштерн О. Теория игр и экономическое поведение / перевод с англ. под ред. и с доб. Н.Н. Воробьева. – М.: Наука, 1970. – 708 с.

23. Perrone L.F., Nelson S.C. A Study of On-Off Attack Models for Wireless Ad Hoc Networks // 2006 1st Workshop on Operator-Assisted (Wireless Mesh) Community Networks. – 2006. – P. 1 – 10.

ПРИЛОЖЕНИЕ А

Исходный код симулятора

Основной файл симулятора interaction.py:

```
from entities.road import *
from entities.field import *
from entities.vehicle import *
from math import exp, pow, log, sqrt
import numpy as np
import logging

logger = logging.getLogger(__name__)
logging.basicConfig(level=logging.INFO)

row_amount = 10  # количество рядов
column_amount = 10  # количество столбцов
cell_size = 10  # длина/ширина одного элементарного участка

# Параметры эксперимента
data_type_amount = 1    # количество видов инфрмации
data_type_coefficients = [{"a": 0.5, "a'": 0.2}]    # массив показателей a и a'
saboteur_percentage = 0.1   # доля диверсантов
experiment_amount = 1000  # количество экспериментов в серии
total_UV = 1000    # количество элементов в эксперименте
sensors_radius = 1  # радиус действия сенсоров
communication_radius = 10   # радиус действия модуля связи

experiment_type = "Classic reputation"  # Тип эксперимента
# experiment_type = "Game theory"

path_to_file = experiment_type + "_" + str(saboteur_percentage) + ".txt"
```

```
roads = [  # карта дорог
    Road(
        road_type="vertical",
        road_direction="codirectional",
        road_cells=[4, 14, 24, 34, 44, 54, 64, 74, 84, 94]
    ),
    Road(
        road_type="vertical",
        road_direction="contradirectional",
        road_cells=[95, 85, 75, 65, 55, 45, 35, 25, 15, 5]
    ),
    Road(
        road_type="horizontal",
        road_direction="contradirectional",
        road_cells=[49, 48, 47, 46, 45, 44, 43, 42, 41, 40]
    ),
    Road(
        road_type="horizontal",
        road_direction="codirectional",
        road_cells=[50, 51, 52, 53, 54, 55, 56, 57, 58, 59]
    )
]

# поле заданной длины и ширины с картой дорог
field = Field(row_amount=row_amount, column_amount=column_amount,
roads=roads)


def check_if_conflicts_are_presented(vehicle_routes, iteration):
```

```
    conflicts_are_presented = False
    conflicts_info, unique_conflicting_pairs = [], []
    for i in range(len(vehicle_routes)):
        for j in range(len(vehicle_routes)):
            if i != j:
                min_vehicle_route_length = min(len(vehicle_routes[i]),
len(vehicle_routes[j]))
                for elementary_area_in_route in range(min_vehicle_route_length):
                    if vehicle_routes[i][elementary_area_in_route] ==
vehicle_routes[j][elementary_area_in_route]:
                        if i not in unique_conflicting_pairs and j not in
unique_conflicting_pairs:
                            unique_conflicting_pairs.append(i)
                            unique_conflicting_pairs.append(j)
                            conflicts_info.append(
                                {
                                    "elementary_area":
vehicle_routes[i][elementary_area_in_route],
                                    "index_in_vehicle_route": elementary_area_in_route
                                }
                            )
                            conflicts_are_presented = True
    return conflicts_are_presented, conflicts_info


def calculate_intersection_bandwidth(vehicle_routes, route_in_conflict,
index_in_vehicle_route,
                                     route_index_in_conflict):
    elementary_areas_on_intersection = field.get_elementary_areas_on_intersections()
    if route_in_conflict[index_in_vehicle_route] in elementary_areas_on_intersection
```

```
and \
            route_in_conflict[index_in_vehicle_route - 1] not in
elementary_areas_on_intersection:
        if route_in_conflict[index_in_vehicle_route] -
route_in_conflict[index_in_vehicle_route - 1] == 2:
            route_in_conflict.insert(index_in_vehicle_route,
route_in_conflict[index_in_vehicle_route - 1] + 1)
        elif route_in_conflict[index_in_vehicle_route] -
route_in_conflict[index_in_vehicle_route - 1] == 1:
            route_in_conflict.insert(index_in_vehicle_route,
route_in_conflict[index_in_vehicle_route - 1])
        elif route_in_conflict[index_in_vehicle_route - 1] -
route_in_conflict[index_in_vehicle_route] == 2:
            route_in_conflict.insert(index_in_vehicle_route,
route_in_conflict[index_in_vehicle_route - 1] - 1)
        elif route_in_conflict[index_in_vehicle_route - 1] -
route_in_conflict[index_in_vehicle_route] == 1:
            route_in_conflict.insert(index_in_vehicle_route,
route_in_conflict[index_in_vehicle_route - 1])
        elif route_in_conflict[index_in_vehicle_route] -
route_in_conflict[index_in_vehicle_route - 1] == 20:
            route_in_conflict.insert(index_in_vehicle_route,
route_in_conflict[index_in_vehicle_route - 1] + 10)
        elif route_in_conflict[index_in_vehicle_route] -
route_in_conflict[index_in_vehicle_route - 1] == 10:
            route_in_conflict.insert(index_in_vehicle_route,
route_in_conflict[index_in_vehicle_route - 1])
        elif route_in_conflict[index_in_vehicle_route - 1] -
route_in_conflict[index_in_vehicle_route] == 20:
            route_in_conflict.insert(index_in_vehicle_route,
```

```
route_in_conflict[index_in_vehicle_route - 1] - 10)
        elif route_in_conflict[index_in_vehicle_route - 1] -
route_in_conflict[index_in_vehicle_route] == 10:
            route_in_conflict.insert(index_in_vehicle_route,
route_in_conflict[index_in_vehicle_route - 1])
    elif route_in_conflict[index_in_vehicle_route] in
elementary_areas_on_intersection and \
            route_in_conflict[index_in_vehicle_route - 1] in
elementary_areas_on_intersection:
        route_in_conflict.insert(index_in_vehicle_route,
route_in_conflict[index_in_vehicle_route - 1])
    vehicle_routes[route_index_in_conflict] = route_in_conflict
    max_crossing_intersection_time = 0
    crossing_intersection_steps = 0
    for vehicle_route in vehicle_routes:
        crossing_intersection_time = 0
        for elementary_area in vehicle_route:
            if elementary_area in elementary_areas_on_intersection:
                crossing_intersection_time += 1
        if crossing_intersection_time > max_crossing_intersection_time:
            max_crossing_intersection_time = crossing_intersection_time
        for elementary_area in list(set(vehicle_route)):
            if elementary_area in elementary_areas_on_intersection:
                crossing_intersection_steps += 1
    del route_in_conflict[index_in_vehicle_route]
    return crossing_intersection_steps / max_crossing_intersection_time

def resolve_vehicle_conflicts(vehicles, vehicle_routes, conflict_info):
    is_saboteur = False
    routes_in_conflict, routes_indexes_in_conflict = [], []
```

```
    for i in range(len(vehicle_routes)):
        if len(vehicle_routes[i]) >= conflict_info["index_in_vehicle_route"] + 1:
            if vehicle_routes[i][conflict_info["index_in_vehicle_route"]] == \
                    conflict_info["elementary_area"]:
                routes_in_conflict.append(vehicle_routes[i])
                routes_indexes_in_conflict.append(i)
    need_to_reduce_route = False
    difference = 0
    car_already_stays = False
    for i in range(len(routes_in_conflict)):
        if
routes_in_conflict[i].count(routes_in_conflict[i][conflict_info["index_in_vehicle_rou
te"]]) > 1:
            car_already_stays = True
            if routes_in_conflict[i][conflict_info["index_in_vehicle_route"]] ==
routes_in_conflict[i][
                conflict_info["index_in_vehicle_route"] - 1]:
                need_to_reduce_route = True
                index_to_go = i
    elementary_areas_on_intersection = field.get_elementary_areas_on_intersections()
    if need_to_reduce_route:
        possible_vehicle_routes = [[50, 51, 52, 53, 54, 55, 56, 57, 58, 59],
                                   [50, 51, 52, 53, 54, 55, 45, 35, 25, 15, 5], [50, 51, 52, 53, 54,
64, 74, 84, 94],
                                   [49, 48, 47, 46, 45, 44, 43, 42, 41, 40],
                                   [49, 48, 47, 46, 45, 44, 54, 64, 74, 84, 94], [49, 48, 47, 46, 45,
35, 25, 15, 5],
                                   [95, 85, 75, 65, 55, 45, 35, 25, 15, 5], [95, 85, 75, 65, 55, 56,
57, 58, 59],
                                   [95, 85, 75, 65, 55, 45, 44, 43, 42, 41, 40],
```

```
                    [4, 14, 24, 34, 44, 54, 64, 74, 84, 94], [4, 14, 24, 34, 44, 43, 42,
41, 40],
                    [4, 14, 24, 34, 44, 54, 55, 56, 57, 58, 59]]
        route_to_be_deleted =
list(set(vehicle_routes[routes_indexes_in_conflict[index_to_go]]))
        max_conincidences = -1
        for possible_vehicle_route in possible_vehicle_routes:
            count = 0
            for elementary_area1 in possible_vehicle_route:
                for elementary_area2 in route_to_be_deleted:
                    if elementary_area1 == elementary_area2:
                        count += 1
                        if count > max_conincidences:
                            max_conincidences = count
                            needed_possible_vehicle_route = possible_vehicle_route
        difference = len(needed_possible_vehicle_route) - len(route_to_be_deleted)

        if vehicles[routes_indexes_in_conflict[index_to_go]].is_saboteur:
            is_saboteur = True
        del vehicle_routes[routes_indexes_in_conflict[index_to_go]]  # если с данной
машиной нет разъезда, удалить ее
        del vehicles[routes_indexes_in_conflict[index_to_go]]
    else:
        if car_already_stays:
            max_stay = -1
            for i in range(len(routes_in_conflict)):
                if routes_in_conflict[i].count(
                        routes_in_conflict[i][conflict_info["index_in_vehicle_route"]]) >
max_stay:
                    max_stay = routes_in_conflict[i].count(
```

```
                routes_in_conflict[i][conflict_info["index_in_vehicle_route"]])
            index_to_go = i
    else:
        bandwidths = []
        for i in range(len(routes_in_conflict)):
            bandwidths.append(calculate_intersection_bandwidth(vehicle_routes,
routes_in_conflict[i],
                                                    conflict_info["index_in_vehicle_route"],
                                                    routes_indexes_in_conflict[i]))
        index_to_go = bandwidths.index(min(bandwidths))
    for i in range(len(routes_indexes_in_conflict)):
        if i != index_to_go:
            if vehicle_routes[routes_indexes_in_conflict[i]][
                conflict_info["index_in_vehicle_route"]] in
elementary_areas_on_intersection and \
                    vehicle_routes[routes_indexes_in_conflict[i]][
                        conflict_info["index_in_vehicle_route"] - 1] not in
elementary_areas_on_intersection:
                if
vehicle_routes[routes_indexes_in_conflict[i]][conflict_info["index_in_vehicle_route"
]] - \
                        vehicle_routes[routes_indexes_in_conflict[i]][
                            conflict_info["index_in_vehicle_route"] - 1] == 2:

vehicle_routes[routes_indexes_in_conflict[i]].insert(conflict_info["index_in_vehicle_
route"],
                                                     vehicle_routes[
                                                         routes_indexes_in_conflict[i]][
                                                         conflict_info[
                                                             "index_in_vehicle_route"] - 1] +
```

```
1)
                elif
vehicle_routes[routes_indexes_in_conflict[i]][conflict_info["index_in_vehicle_route"
]] - \
                        vehicle_routes[routes_indexes_in_conflict[i]][
                            conflict_info["index_in_vehicle_route"] - 1] == 1:

vehicle_routes[routes_indexes_in_conflict[i]].insert(conflict_info["index_in_vehicle_
route"],
                                                     vehicle_routes[
                                                        routes_indexes_in_conflict[i]][
                                                        conflict_info[
                                                            "index_in_vehicle_route"] - 1])
                elif
vehicle_routes[routes_indexes_in_conflict[i]][conflict_info["index_in_vehicle_route"
] - 1] - \

vehicle_routes[routes_indexes_in_conflict[i]][conflict_info["index_in_vehicle_route"
]] == 2:

vehicle_routes[routes_indexes_in_conflict[i]].insert(conflict_info["index_in_vehicle_
route"],
                                                     vehicle_routes[
                                                        routes_indexes_in_conflict[i]][
                                                        conflict_info[
                                                            "index_in_vehicle_route"] - 1] -
1)
                elif
vehicle_routes[routes_indexes_in_conflict[i]][conflict_info["index_in_vehicle_route"
] - 1] - \
```

```
vehicle_routes[routes_indexes_in_conflict[i]][conflict_info["index_in_vehicle_route"
]] == 1:

vehicle_routes[routes_indexes_in_conflict[i]].insert(conflict_info["index_in_vehicle_
route"],
                                                    vehicle_routes[
                                                        routes_indexes_in_conflict[i]][
                                                        conflict_info[
                                                            "index_in_vehicle_route"] - 1])
                elif
vehicle_routes[routes_indexes_in_conflict[i]][conflict_info["index_in_vehicle_route"
]] - \
                        vehicle_routes[routes_indexes_in_conflict[i]][
                            conflict_info["index_in_vehicle_route"] - 1] == 20:

vehicle_routes[routes_indexes_in_conflict[i]].insert(conflict_info["index_in_vehicle_
route"],
                                                    vehicle_routes[
                                                        routes_indexes_in_conflict[i]][
                                                        conflict_info[
                                                            "index_in_vehicle_route"] - 1] +
10)
                elif
vehicle_routes[routes_indexes_in_conflict[i]][conflict_info["index_in_vehicle_route"
]] - \
                        vehicle_routes[routes_indexes_in_conflict[i]][
                            conflict_info["index_in_vehicle_route"] - 1] == 10:

vehicle_routes[routes_indexes_in_conflict[i]].insert(conflict_info["index_in_vehicle_
```

```
route"],
                                                vehicle_routes[
                                                    routes_indexes_in_conflict[i]][
                                                    conflict_info[
                                                        "index_in_vehicle_route"] - 1])
                elif
vehicle_routes[routes_indexes_in_conflict[i]][conflict_info["index_in_vehicle_route"
] - 1] - \
                        vehicle_routes[routes_indexes_in_conflict[i]][
                            conflict_info["index_in_vehicle_route"]] == 20:

vehicle_routes[routes_indexes_in_conflict[i]].insert(conflict_info["index_in_vehicle_
route"],
                                                vehicle_routes[
                                                    routes_indexes_in_conflict[i]][
                                                    conflict_info[
                                                        "index_in_vehicle_route"] - 1] -
10)
                elif
vehicle_routes[routes_indexes_in_conflict[i]][conflict_info["index_in_vehicle_route"
] - 1] - \
                        vehicle_routes[routes_indexes_in_conflict[i]][
                            conflict_info["index_in_vehicle_route"]] == 10:

vehicle_routes[routes_indexes_in_conflict[i]].insert(conflict_info["index_in_vehicle_
route"],
                                                vehicle_routes[
                                                    routes_indexes_in_conflict[i]][
                                                    conflict_info[
                                                        "index_in_vehicle_route"] - 1])
```

```
            elif vehicle_routes[routes_indexes_in_conflict[i]][
                conflict_info["index_in_vehicle_route"]] in
elementary_areas_on_intersection and \
                    vehicle_routes[routes_indexes_in_conflict[i]][
                        conflict_info["index_in_vehicle_route"] - 1] in
elementary_areas_on_intersection:

vehicle_routes[routes_indexes_in_conflict[i]].insert(conflict_info["index_in_vehicle_
route"],

vehicle_routes[routes_indexes_in_conflict[i]][
                                                    conflict_info[
                                                        "index_in_vehicle_route"] - 1])
            elif vehicle_routes[routes_indexes_in_conflict[i]][
                conflict_info["index_in_vehicle_route"]] not in
elementary_areas_on_intersection and \
                    vehicle_routes[routes_indexes_in_conflict[i]][
                        conflict_info["index_in_vehicle_route"] - 1] not in
elementary_areas_on_intersection:

vehicle_routes[routes_indexes_in_conflict[i]].insert(conflict_info["index_in_vehicle_
route"],

vehicle_routes[routes_indexes_in_conflict[i]][
                                                    conflict_info[
                                                        "index_in_vehicle_route"] - 1])
    return vehicle_routes, need_to_reduce_route, difference, is_saboteur


def calculate_reputation(vehicle_subj, vehicle_obj, truth):
```

```
    vehicle_subj.reputation_time[vehicle_obj] += 1
    if truth >= 0.5:
        sum_action = vehicle_subj.sum_action[vehicle_obj] + truth
        reputation = sum_action / (vehicle_subj.reputation_time[vehicle_obj])
        vehicle_subj.sum_action[vehicle_obj] = sum_action
        vehicle_subj.reputation[vehicle_obj] = reputation
        return reputation
    else:
        sum_action = vehicle_subj.sum_action[vehicle_obj] + truth - \
                (vehicle_subj.reputation[vehicle_obj] - exp(-(1 - truth *
(vehicle_subj.reputation_time[vehicle_obj]))))
        reputation = sum_action / (vehicle_subj.reputation_time[vehicle_obj])
        vehicle_subj.sum_action[vehicle_obj] = sum_action
        vehicle_subj.reputation[vehicle_obj] = reputation
        return reputation

def ask_other(vehicle_routes, vehicles, index_obj, index_subj):
    n_truth = 0
    # truth = 0
    for vehicle_route in vehicle_routes:
        if vehicle_routes.index(vehicle_route) != index_obj and
vehicle_routes.index(vehicle_route) != index_subj:
            if not
vehicles[vehicle_routes.index(vehicle_route)].is_blocked[vehicles[index_obj]]:
                if is_visible(vehicle_routes, index_obj,
vehicle_routes.index(vehicle_route)):
                    n_truth += 1
    return n_truth

def is_visible(vehicle_routes, index_obj, index_subj):
```

```
    vehicle_route_obj = vehicle_routes[index_obj]
    vehicle_route_subj = vehicle_routes[index_subj]
    if abs(vehicle_route_obj[0] // 10 - vehicle_route_subj[0] // 10) <= sensors_radius \
            and abs(vehicle_route_obj[0] % 10 - vehicle_route_subj[0] % 10) <=
sensors_radius:
        return True
    return False

def communication_is_possible(vehicle_route_obj, vehicle_route_subj):
    if abs(vehicle_route_obj[0] // 10 - vehicle_route_subj[0] // 10) <=
communication_radius \
            and abs(vehicle_route_obj[0] % 10 - vehicle_route_subj[0] % 10) <=
communication_radius:
        return True
    return False

def check_performance(vehicles, vehicle_routes, iteration,
total_not_full_data_situation, truth_total):
    for i in range(len(vehicles)):
        for j in range(len(vehicles)):
            if i != j and not vehicles[i].is_blocked[vehicles[j]] and \
                    communication_is_possible(vehicle_routes[j], vehicle_routes[i]):
                truth = 0
                for data_type_index in range(data_type_amount):
                    iteration_old = vehicles[i].old_time[vehicles[j]]
                    # action
                    if vehicles[j].is_saboteur:
                        if iteration % 3 == 0 or iteration % 3 == 1:
                            action = 1
                        else:
```

```
                action = 0
            else:
                action = np.random.choice(np.arange(0, 2), p=[0.9, 0.1])
            # truth
            if action == 1:
                if is_visible(vehicle_routes, j, i):
                    truth = 0
                elif ask_other(vehicle_routes, vehicles, j, i) > 0:
                    truth = 0
                else:
                    if experiment_type == "Classic reputation":
                        truth = 0.5
                    else:
                        value = (1 - pow(data_type_coefficients[data_type_index]["a"],
                                         iteration_old - iteration)
                                 * (-log(data_type_coefficients[data_type_index]["a"])
                                    * (iteration - (iteration_old - 1))
                                    + 1)) / (1 -
pow(data_type_coefficients[data_type_index]["a'"], -1)
                                             * (-
log(data_type_coefficients[data_type_index]["a'"]) + 1))

                        if value <= 1:
                            truth = value
                        else:
                            truth = 1

                    truth_total += truth
                    total_not_full_data_situation += 1
```

```
                else:
                    if is_visible(vehicle_routes, j, i):
                        truth = 1
                    elif ask_other(vehicle_routes, vehicles, j, i) > 0:
                        truth = 1
                    else:
                        if experiment_type == "Classic reputation":
                            truth = 0.5
                        else:
                            value = (1 - pow(data_type_coefficients[data_type_index]["a"],
iteration_old - iteration)
                                     * (-log(data_type_coefficients[data_type_index]["a"])
                                       * (iteration - (iteration_old - 1))
                                       + 1)) / (1 -
pow(data_type_coefficients[data_type_index]["a'"], -1)
                                                * (-
log(data_type_coefficients[data_type_index]["a'"]) + 1))
                            if value <= 1:
                                truth = value
                            else:
                                truth = 1

                        truth_total += truth
                        total_not_full_data_situation += 1
                R = vehicles[i].reputation[vehicles[j]]
                calculate_reputation(vehicles[i], vehicles[j], truth)
                trust = sqrt(pow(R, 2) + pow(truth, 2)) - sqrt(pow(1 - R, 2) + pow(1 - truth,
2))

                if trust < 0:
```

```
                    vehicles[i].is_blocked[vehicles[j]] = True

                vehicles[i].old_time[vehicles[j]] = iteration
    return total_not_full_data_situation, truth_total

def start_moving(experiment_number):
    total_generated = 0
    total_finished = 0
    total_vehicle_amount = 0  # Сколько всего машин прошло перекресток за заданное число шагов
    total_saboteur_amount = 0
    total_step_amount = 0  # Сколько всего шагов было сделано машинами, чтобы пройти перекресток
    vehicle_routes = []
    vehicles = []
    tp, fp, tn, fn, detected_saboteurs = 0, 0, 0, 0, 0

    iteration = 0
    total_not_full_data_situation = 0
    truth_total = 0

    logger.info("Mode: %s Percentages %f Experiment %d", experiment_type, saboteur_percentage, experiment_number)

    while total_finished < total_UV:
        iteration += 1
        if iteration != 0:
            total_not_full_data_situation,  truth_total = check_performance(vehicles, vehicle_routes, iteration, total_not_full_data_situation, truth_total)
            for j in range(len(vehicle_routes)):  # удаление пройденных машинами
```

```
элементарных участков
                del vehicle_routes[j][0]
                total_step_amount += 1
                vehicles[j].steps += 1
            copy_vehicle_routes = vehicle_routes.copy()
            copy_vehicles = vehicles.copy()

            for i in range(len(copy_vehicle_routes)):
                if not copy_vehicle_routes[i]:
                    for key in vehicles[i].is_blocked.keys():
                        if not vehicles[i].is_blocked[key] and not key.is_saboteur:
                            tn += 1
                        if not vehicles[i].is_blocked[key] and key.is_saboteur:
                            fn += 1
                        if vehicles[i].is_blocked[key] and key.is_saboteur:
                            tp += 1
                        if vehicles[i].is_blocked[key] and not key.is_saboteur:
                            fp += 1
                    total_finished += 1
            vehicles = [copy_vehicles[i] for i in range(len(copy_vehicles)) if
copy_vehicle_routes[i]]
            vehicle_routes = [copy_vehicle_route for copy_vehicle_route in
copy_vehicle_routes if copy_vehicle_route]

            del copy_vehicle_routes
            del copy_vehicles

        if total_generated < total_UV:
            if total_UV - total_generated >= 4:
                new_vehicle_amount = np.random.choice(np.array([0, 1, 2, 3, 4]), p=[0.2,
```

```
0.2, 0.2, 0.2, 0.2])
        elif total_UV - total_generated == 3:
            new_vehicle_amount = np.random.choice(np.array([0, 1, 2, 3]), p=[0.25,
0.25, 0.25, 0.25])
        elif total_UV - total_generated == 2:
            new_vehicle_amount = np.random.choice(np.array([0, 1, 2]), p=[0.33, 0.33,
0.34])
        elif total_UV - total_generated == 1:
            new_vehicle_amount = np.random.choice(np.array([0, 1]), p=[0.5, 0.5])
        else:
            new_vehicle_amount = 0
        total_generated += new_vehicle_amount
        for new_vehicle in range(new_vehicle_amount):
            if new_vehicle == 0 and total_saboteur_amount <= total_vehicle_amount *
saboteur_percentage:
                is_saboteur = True
            else:
                is_saboteur = False
            vehicle = Vehicle(
                initial_speed=1,
                field=field,
                existing_vehicle_routes=vehicle_routes,
                is_saboteur=is_saboteur
            )
            vehicle_route = vehicle.determine_vehicle_route()
            if vehicle_route:
                vehicles.append(vehicle)
                vehicle_routes.append(vehicle_route)
                total_vehicle_amount += 1
                if is_saboteur:
```

```
                total_saboteur_amount += 1

        # Проверка на наличие конфликтов
        conflicts_are_presented, conflicts_info =
check_if_conflicts_are_presented(vehicle_routes, iteration)
        conflicts_info_check = []
        while conflicts_are_presented:
            for conflict_info in conflicts_info:
                if conflicts_info_check:
                    if conflict_info not in conflicts_info_check:
                        break
                vehicle_routes, deleted_cells, difference, saboteur = \
                    resolve_vehicle_conflicts(vehicles, vehicle_routes, conflict_info)

                if deleted_cells:
                    total_step_amount -= difference
                    total_generated -= 1
                    if saboteur:
                        total_saboteur_amount -= 1
                conflicts_are_presented, conflicts_info_check =
check_if_conflicts_are_presented(vehicle_routes,
                                                                    iteration)
            conflicts_are_presented, conflicts_info =
check_if_conflicts_are_presented(vehicle_routes, iteration)

        for vehicle in vehicles:
            for new_vehicle in vehicles:
                if vehicle != new_vehicle:
                    vehicle.add_vehicle(new_vehicle, iteration)
```

```
    for vehicle_route in vehicle_routes:
        total_step_amount += len(vehicle_route)
    with open(path_to_file, "a") as file:
        file.write(str(tp) + '\t' + str(fp) + '\t' + str(tn) + '\t' + str(fn) + '\n')

    return total_step_amount / total_vehicle_amount

def main():
    step_amount_per_vehicle = 0
    for mode in ["Classic reputation", "Game theory"]:
        for percent in range(1, 5, 1):
            global experiment_type, saboteur_percentage, path_to_file
            experiment_type = mode
            saboteur_percentage = percent/10
            path_to_file = mode + " " + str(saboteur_percentage) + ".txt"
            open(path_to_file, "w")
            for experiment in range(experiment_amount):
                step_amount_per_vehicle += start_moving(experiment)

if __name__ == "__main__":
    main()
```

Файл vehicle.py:

```
from random import choice
class Vehicle:
    def __init__(self, initial_speed, field, existing_vehicle_routes, is_saboteur):
        self.initial_speed = initial_speed
        self.field = field
        self.existing_vehicle_routes = existing_vehicle_routes
        # self.truth = {}
        self.reputation = dict()
```

```
        self.sum_action = dict()
        self.is_saboteur = is_saboteur
        self.steps = 1
        self.is_blocked = dict()
        self.old_time = dict()
        self.reputation_time = dict()

    def add_vehicle(self, vehicle, iteration):
        if vehicle not in self.reputation.keys():
            self.reputation[vehicle] = 0.5
            self.sum_action[vehicle] = 0.5
            self.is_blocked[vehicle] = False
            self.old_time[vehicle] = iteration
            self.reputation_time[vehicle] = 1

    # def del_vehicle(self):

    def choose_movement_direction(self):  # рандомно выбрать направление движения (прямо, налево или направо)
        possible_movement_variants = ["go_directly", "turn_left", "turn_right"]
        return choice(possible_movement_variants)

    def determine_start_point(self):
        # элементарные участки, находящиеся на краях заданного поля
        possible_start_points = [min(road.get_road_cells()) for road in self.field.get_roads()
                                 if road.get_road_direction() == "codirectional"]
        possible_start_points.extend([max(road.get_road_cells()) for road in self.field.get_roads()
                                      if road.get_road_direction() == "contradirectional"])
```

```
        # элементарные участки, которые при вхождении новым автомобилем
будут заняты другими ТС
        already_occupied_start_points = []
        for existing_vehicle_route in self.existing_vehicle_routes:
            already_occupied_start_points.append(existing_vehicle_route[0])
        free_start_points = []
        for possible_start_point in possible_start_points:
            if possible_start_point not in already_occupied_start_points:
                free_start_points.append(possible_start_point)
        if len(free_start_points) == 0:
            start_point = -1
        else:
            start_point = choice(free_start_points)
        return start_point

    def point_is_on_horizontal_road(self, point):
        for road in self.field.get_roads():
            if point in road.get_road_cells() and road.get_road_type() == "horizontal":
                return True
        return False

    def point_is_on_vertical_road(self, point):
        for road in self.field.get_roads():
            if point in road.get_cells() and road.get_road_type() == "vertical":
                return True
        return False

    def determine_vehicle_route(self):
        start_point = self.determine_start_point()
        if start_point == -1:  # если все возможные стартовые точки заняты, машина
```

```
не входит в поле
            vehicle_route = []
        else:
            vehicle_route = []
            related_elementary_areas_on_intersections = []
            step_length = self.initial_speed
            movement_direction = self.choose_movement_direction()
            if movement_direction == "go_directly":
                if self.point_is_on_horizontal_road(start_point):
                    for elementary_area_on_intersection in
self.field.get_elementary_areas_on_intersections():
                        if elementary_area_on_intersection // self.field.get_column_amount()
== start_point // self.field.get_column_amount():

related_elementary_areas_on_intersections.append(elementary_area_on_intersection)
                    if start_point < min(related_elementary_areas_on_intersections):
                        right_limit = start_point + self.field.get_column_amount()
                        while start_point < right_limit:
                            vehicle_route.append(start_point)
                            if start_point == min(related_elementary_areas_on_intersections):
# снижение скорости на перекрестке до 1
                                start_point += 1
                            else:
                                start_point += step_length
                    else:
                        left_limit = start_point - self.field.get_column_amount()
                        while start_point > left_limit:
                            vehicle_route.append(start_point)
                            if start_point == max(related_elementary_areas_on_intersections):
                                start_point -= 1
```

```
                else:
                    start_point -= step_length
            else:
                for elementary_area_on_intersection in
self.field.get_elementary_areas_on_intersections():
                    if elementary_area_on_intersection % self.field.get_row_amount() ==
start_point % self.field.get_row_amount():

related_elementary_areas_on_intersections.append(elementary_area_on_intersection)
                if start_point < min(related_elementary_areas_on_intersections):
                    bottom_limit = start_point + self.field.get_row_amount() *
self.field.get_column_amount()
                    while start_point < bottom_limit:
                        vehicle_route.append(start_point)
                        if start_point == min(related_elementary_areas_on_intersections):
                            start_point += self.field.get_column_amount()
                        else:
                            start_point += step_length * self.field.get_column_amount()
                else:
                    up_limit = start_point - self.field.get_row_amount() *
self.field.get_column_amount()
                    while start_point > up_limit:
                        vehicle_route.append(start_point)
                        if start_point == max(related_elementary_areas_on_intersections):
                            start_point -= self.field.get_column_amount()
                        else:
                            start_point -= step_length * self.field.get_column_amount()
        elif movement_direction == "turn_left":
            if self.point_is_on_horizontal_road(start_point):
                for elementary_area_on_intersection in
```

```
self.field.get_elementary_areas_on_intersections():
                if elementary_area_on_intersection // self.field.get_column_amount()
== start_point // self.field.get_column_amount():

related_elementary_areas_on_intersections.append(elementary_area_on_intersection)
            if start_point < min(related_elementary_areas_on_intersections):
                while start_point < max(related_elementary_areas_on_intersections):
                    vehicle_route.append(start_point)
                    if start_point == min(related_elementary_areas_on_intersections):
                        start_point += 1
                    else:
                        start_point += step_length
                up_limit = start_point - self.field.get_row_amount() *
self.field.get_column_amount() / 2
                while start_point >= up_limit:
                    vehicle_route.append(start_point)
                    if start_point == max(related_elementary_areas_on_intersections):
                        start_point -= self.field.get_column_amount()
                    else:
                        start_point -= step_length * self.field.get_column_amount()
            else:
                while start_point > min(related_elementary_areas_on_intersections):
                    vehicle_route.append(start_point)
                    if start_point == max(related_elementary_areas_on_intersections):
                        start_point -= 1
                    else:
                        start_point -= step_length
                bottom_limit = start_point + self.field.get_row_amount() *
self.field.get_column_amount() / 2
                while start_point <= bottom_limit:
```

```
                    vehicle_route.append(start_point)
                    if start_point == min(related_elementary_areas_on_intersections):
                        start_point += self.field.get_column_amount()
                    else:
                        start_point += step_length * self.field.get_column_amount()
            else:
                for elementary_area_on_intersection in
self.field.get_elementary_areas_on_intersections():
                    if elementary_area_on_intersection % self.field.get_column_amount()
== start_point % self.field.get_column_amount():

related_elementary_areas_on_intersections.append(elementary_area_on_intersection)
                if start_point < min(related_elementary_areas_on_intersections):
                    while start_point < max(related_elementary_areas_on_intersections):
                        vehicle_route.append(start_point)
                        if start_point == min(related_elementary_areas_on_intersections):
                            start_point += self.field.get_column_amount()
                        else:
                            start_point += step_length * self.field.get_column_amount()
                    right_limit = start_point + self.field.get_column_amount() / 2
                    while start_point <= right_limit:
                        vehicle_route.append(start_point)
                        if start_point == max(related_elementary_areas_on_intersections):
                            start_point += 1
                        else:
                            start_point += step_length
                else:
                    while start_point > min(related_elementary_areas_on_intersections):
                        vehicle_route.append(start_point)
                        if start_point == max(related_elementary_areas_on_intersections):
```

```
                    start_point -= self.field.get_column_amount()
                else:
                    start_point -= step_length * self.field.get_column_amount()
            left_limit = start_point - self.field.get_column_amount() / 2
            while start_point >= left_limit:
                vehicle_route.append(start_point)
                if start_point == min(related_elementary_areas_on_intersections):
                    start_point -= 1
                else:
                    start_point -= step_length
        else:
            if self.point_is_on_horizontal_road(start_point):
                for elementary_area_on_intersection in
self.field.get_elementary_areas_on_intersections():
                    if elementary_area_on_intersection // self.field.get_column_amount()
== start_point // self.field.get_column_amount():

related_elementary_areas_on_intersections.append(elementary_area_on_intersection)
                if start_point < min(related_elementary_areas_on_intersections):
                    while start_point < min(related_elementary_areas_on_intersections):
                        vehicle_route.append(start_point)
                        start_point += step_length
                    bottom_limit = start_point + self.field.get_row_amount() *
self.field.get_column_amount() / 2
                    while start_point < bottom_limit:
                        vehicle_route.append(start_point)
                        start_point += step_length * self.field.get_column_amount()
                else:
                    while start_point > max(related_elementary_areas_on_intersections):
                        vehicle_route.append(start_point)
```

```
                    start_point -= step_length
                up_limit = start_point - self.field.get_row_amount() *
self.field.get_column_amount() / 2
                while start_point > up_limit:
                    vehicle_route.append(start_point)
                    start_point -= step_length * self.field.get_column_amount()
            else:
                for elementary_area_on_intersection in
self.field.get_elementary_areas_on_intersections():
                    if elementary_area_on_intersection % self.field.get_column_amount()
== start_point % self.field.get_column_amount():

related_elementary_areas_on_intersections.append(elementary_area_on_intersection)
                if start_point < min(related_elementary_areas_on_intersections):
                    while start_point < min(related_elementary_areas_on_intersections):
                        vehicle_route.append(start_point)
                        start_point += step_length * self.field.get_column_amount()
                    left_limit = start_point - self.field.get_column_amount() / 2
                    while start_point > left_limit:
                        vehicle_route.append(start_point)
                        start_point -= step_length
                else:
                    while start_point > max(related_elementary_areas_on_intersections):
                        vehicle_route.append(start_point)
                        start_point -= step_length * self.field.get_column_amount()
                    right_limit = start_point + self.field.get_column_amount() / 2
                    while start_point < right_limit:
                        vehicle_route.append(start_point)
                        start_point += step_length
        return vehicle_route
```

Файл road.py:

```
class Road:
    def __init__(self, road_type, road_direction, road_cells):
        self.road_type = road_type
        self.road_direction = road_direction
        self.road_cells = road_cells

    def get_road_direction(self):
        return self.road_direction

    def get_road_type(self):
        return self.road_type

    def get_road_cells(self):
        return self.road_cells
```

Файл field.py:

```
class Field:
    def __init__(self, row_amount, column_amount, roads):
        self.row_amount = row_amount
        self.column_amount = column_amount
        self.roads = roads

    def get_row_amount(self):
        return self.row_amount

    def get_column_amount(self):
        return self.column_amount

    def get_roads(self):
        return self.roads
```

```
    def get_elementary_areas_on_intersections(self):
        vertical_roads, horizontal_roads, intersections = [], [], []
        for road in self.roads:
            if road.get_road_type() == "vertical":
                vertical_roads.append(road)
            else:
                horizontal_roads.append(road)
        for vertical_road in vertical_roads:
            for elementary_area_on_vertical_road in vertical_road.get_road_cells():
                for horizontal_road in horizontal_roads:
                    for elementary_area_on_horizontal_road in
horizontal_road.get_road_cells():
                        if elementary_area_on_horizontal_road ==
elementary_area_on_vertical_road:
                            intersections.append(elementary_area_on_vertical_road)
        return intersections
```