Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра информационных технологий**

**ОТЧЁТ №8**

**Дисциплина: Многоагентное моделирование**

Работу выполнил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. А. Иванов

Направление подготовки: 02.03.03 Математическое обеспечение и администрирование информационных систем

Преподаватель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А. А. Миков

Краснодар

2024

**Цель работы**

Запрограммировать симуляцию жизни колоний на Марсе. Колонии получаю прибыль, преодолевают трудные события и участвуют в аукционах.

**Описание задачи**

2150 год, Марс. Есть несколько колоний, управляемых агентами-администраторами. Каждая колония имеет следующие характеристики:

- уровень (максимум 10);

- баланс;

- затраты в цикл;

- доходы в цикл;

- опыт.

Максимальное время моделирования T. Цикл состоит из t\_iter итераций. Баланс обновляется в начале цикла (добавляется значение текущего дохода и вычитается значение текущих расходов). Изначально у каждой колонии одинаковый баланс B, но разное соотношение затрат и доходов (доход > затраты).

Когда обновляется баланс, также обновляется опыт колонии на разницу между предыдущим балансом и текущим (e = e + b\_current – b\_previous), опыт может уменьшаться, но не может стать отрицательным. Когда значение опыта достигает константного значения L, опыт обнуляется и уровень увеличивается (уровень не может уменьшаться).

Условием победы колонии считается достижение максимального уровня. Условием поражения и уничтожения колонии считается уход баланса в минус.

Раз в несколько циклов t\_a Земля проводит аукцион (первой или второй цены) артефактов, улучшающих состояние колонии (выигравшие колонии в аукционах не участвуют).

Артефакт может, например:

- увеличить уровень колонии;

- уменьшить затраты на обслуживание;

- увеличить доход;

- спасти колонию от разрушения (не более одного артефакта у каждой колонии одновременно).

Изменение может иметь как разовый эффект, так и постоянный, например для увеличения дохода может быть как единовременная выплата, так и выплаты на протяжении нескольких циклов.

Также раз в несколько циклов t\_e происходят события среды:

- пылевая буря (уменьшает доход колонии на g пунктов и увеличивает расход на j);

- «ренессанс» (эффект прямо противоположный пылевой буре).

Эффект события одноразовый.

**Задача**

Каждому разработчику выдаётся набор из 5 артефактов с различными эффектами. Эффекты являются параметризированными, то есть конкретные значения выбираются разработчиком самостоятельно, равно как и стоимость каждого артефакта.

Разработать алгоритм участия агента в аукционе. В данном случае понятно, что все артефакты только улучшают текущее состояние колонии, поэтому есть смысл участвовать в каждом аукционе. Однако, агент должен понимать, что на следующую ставку ему может не хватить баланса, или что покупка может лишь усугубить состояние (например, снижение расхода на h пунктов не выровняет соотношение доход/расход, но баланс станет меньше, что может привести к поражению раньше).

Построить график распределения вероятностей времени жизни колонии (до поражения) для различных сочетаний входных параметров.

Построить график зависимостей вероятности побед и поражений колоний в зависимости от входных параметров (провести n экспериментов, к концу моделирования рассчитать количество выигравших и проигравших колоний).

**Описание решения**

Для написания программы был выбран язык Python ввиду его простой реализации ООП.

Для реализации программы было создано 6 файлов: constants.py, Colony.py, Artifact.py, Agent.py, Mars.py, main.py.

Описание файлов:

constants.py: содержит основные константы, которые влияют на процесс симуляции (кол-во опыта для получения уровня, диапазон значений дохода/расхода, интервал между событиями среды и аукционами и т.д.).

Colony.py: класс, реализующий Колонию. Включает основные поля и методы, такие как подсчёт прибыли, увеличения опыта, переход на новый уровень.

Artifact.py: класс, реализующий Артефакты. Включает в себя базовые поля артефактов, а также содержит метод-обёртку, который вызывает приватные методы-реализации конкретных артефактов (вызов методов происходит через сравнение ID артефакта).

Agent.py: класс, отвечающий за логику участия агента в аукционе.

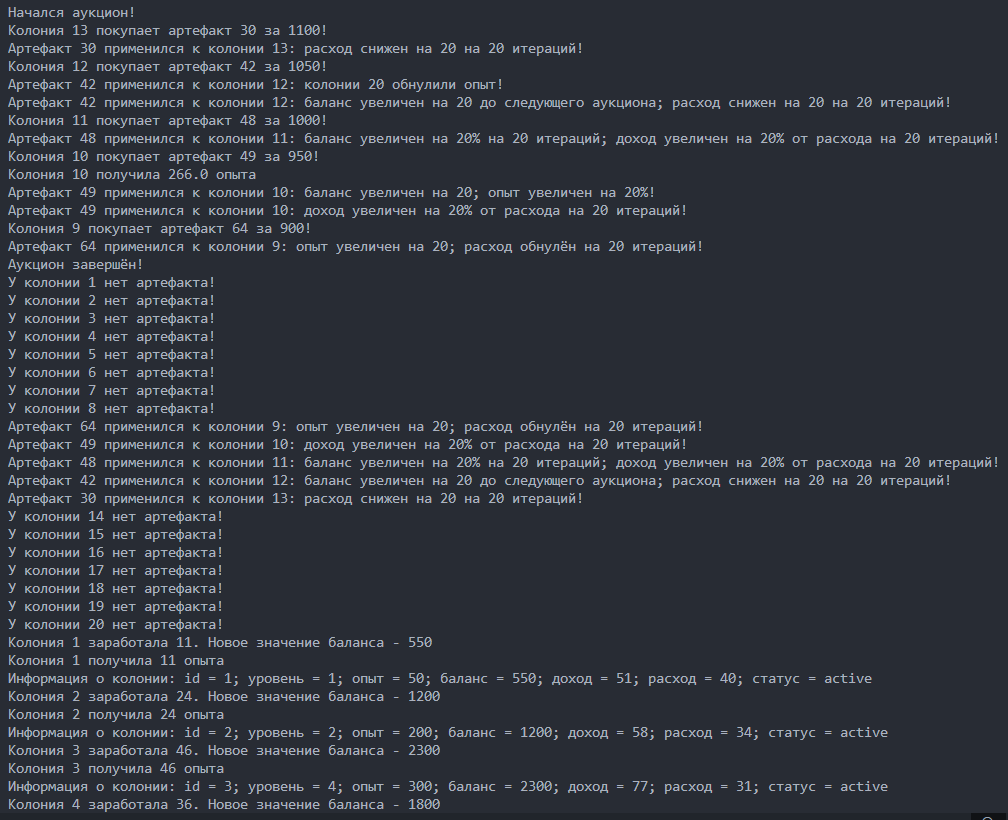
Mars.py: основной класс программы, отвечающий за всё моделирование. Данный класс запускает основной цикл, в котором через определённое число итераций происходят аукционы, возникают события среды, колонии зарабатывают доход и получают опыт.

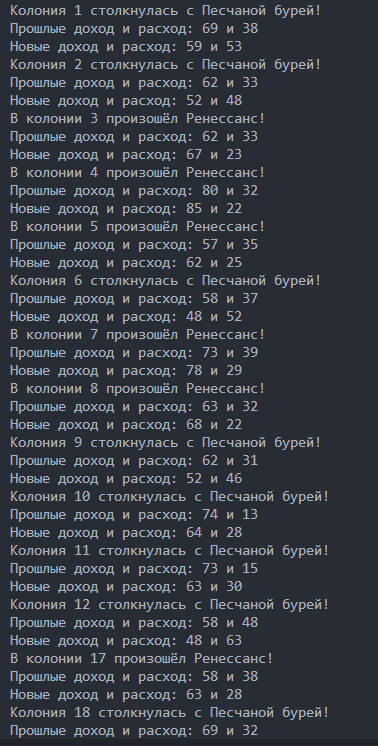
main.py: файл, который запускает симуляцию и строит графики.

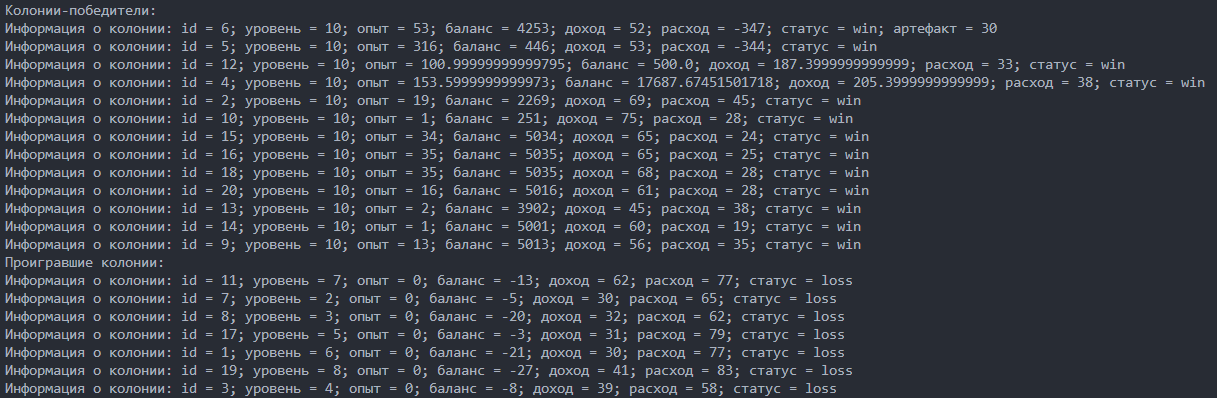
**Код программы**

Код программы решено не вставлять в отчёт текстом ввиду его слишком большого объёма (600+ строк). Код будет прикреплён архивом к сообщению с отчётом.

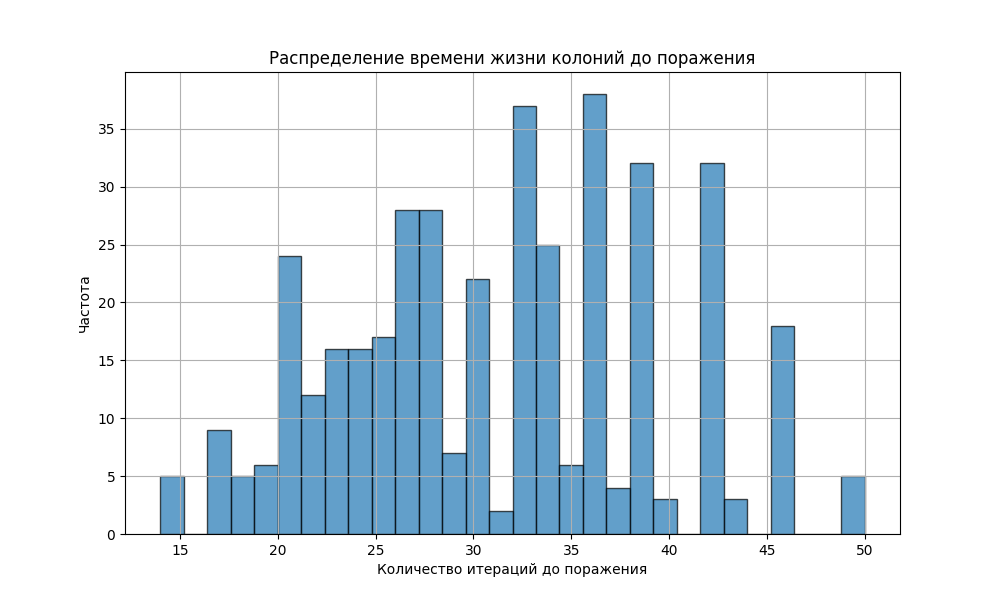
**Пример вывода программы (подробный)**

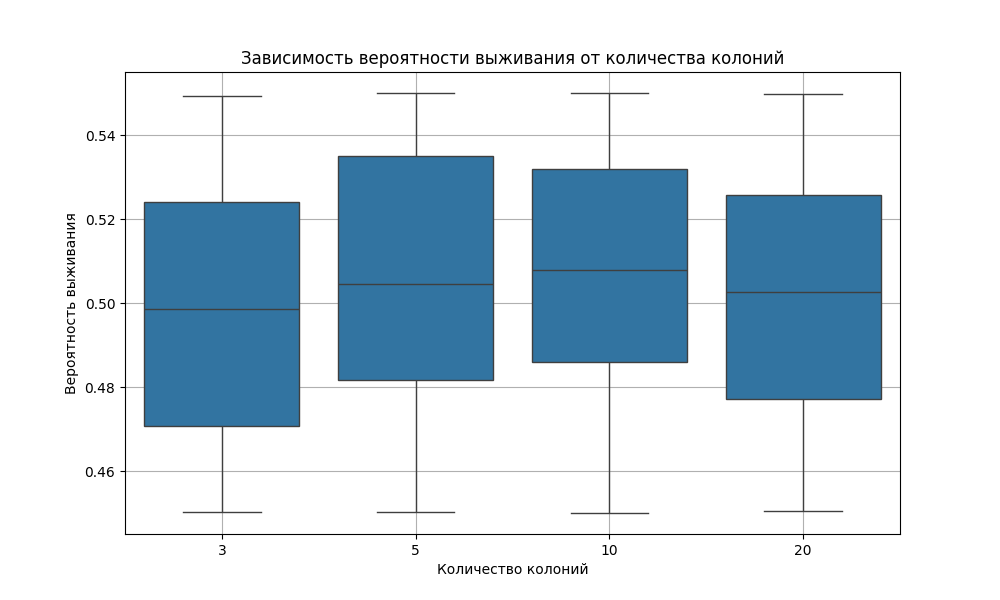
****

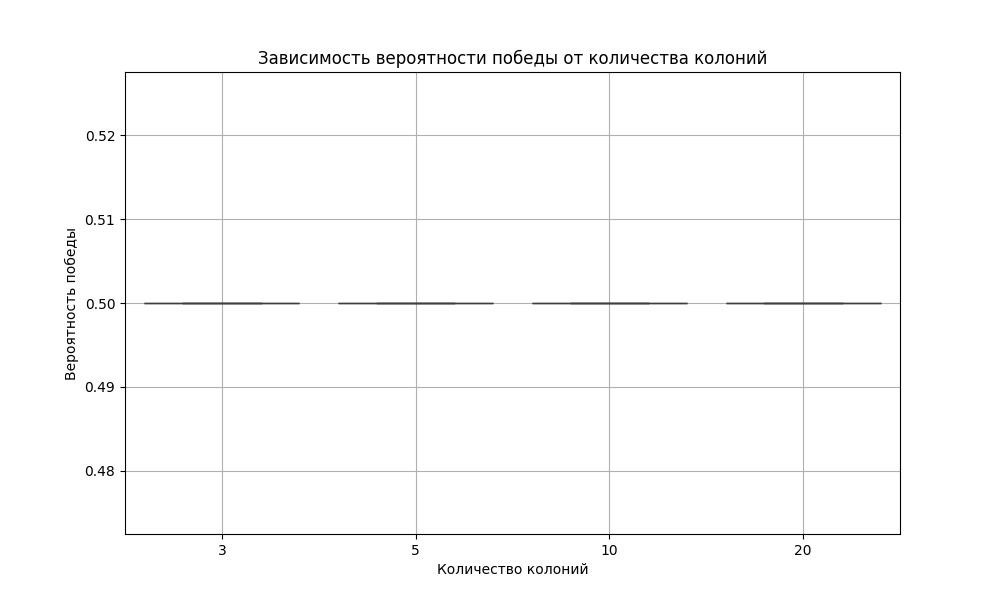
****

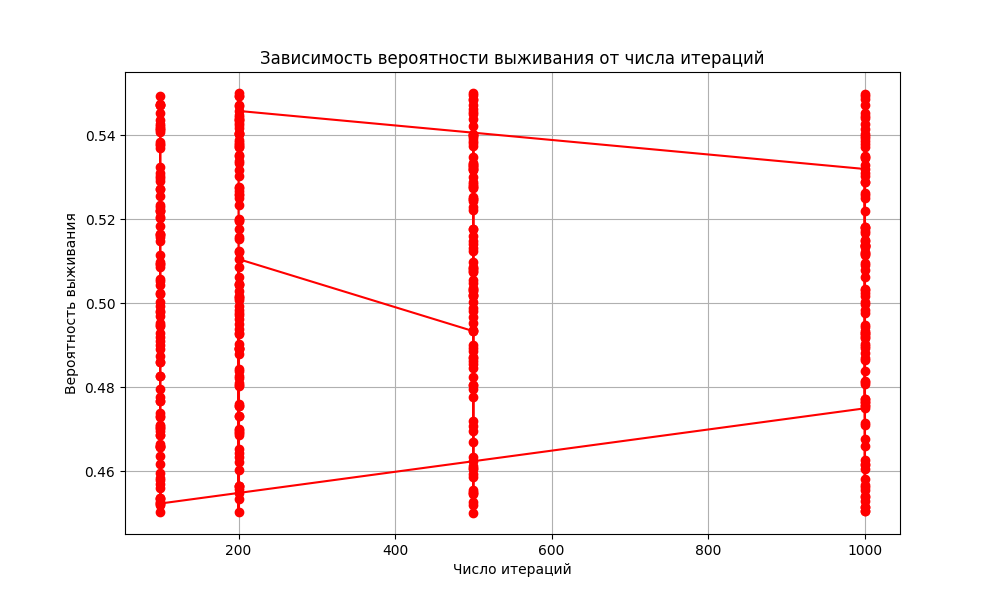
****

**Графики**

****

****

****

****