

# **Рабочая тетрадь**

**по курсу**

**Введение в GameDev:  
Основы игрового ИИ**

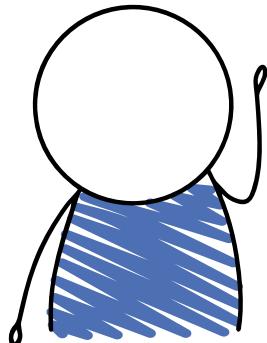


**Студент:**

**Группа:**

**Семестр:**

**Год:**



# СОДЕРЖАНИЕ

|   |    |
|---|----|
| Лекция 1. Введение в теорию игр и поиск по дереву.....      | 3  |
| Практическая работа 1. Поиск по дереву.....                 | 8  |
| Лекция 2. Ad-Hoc Behavior Authoring.....                    | 10 |
| Лабораторная работа 1. Навигация в UE4.....                 | 16 |
| Лабораторная работа 2. Деревья поведений в UE4.....         | 17 |
| Лекция 3. Навигация и технологии программирования ИИ.....   | 18 |
| Проект. Применение методов игрового ИИ в UE4.....           | 22 |
| Лекция 4. Принятие решений и элементы машинного обучения..  | 23 |
| Практическая работа 2. Принятие решений.....                | 30 |
| Лекция 5. Нечеткие модели и системы.....                    | 32 |
| Практическая работа 3. Нечеткие модели и системы.....       | 38 |
| Лекция 6. Эволюционные и роевые алгоритмы.....              | 40 |
| Практическая работа 4. Эволюционные и роевые алгоритмы..... | 48 |
| Итоговая таблица с результатами.....                        | 50 |

# ЛЕКЦИЯ 1. ВВЕДЕНИЕ В ТЕОРИЮ ИГР И ПОИСК ПО ДЕРЕВУ.

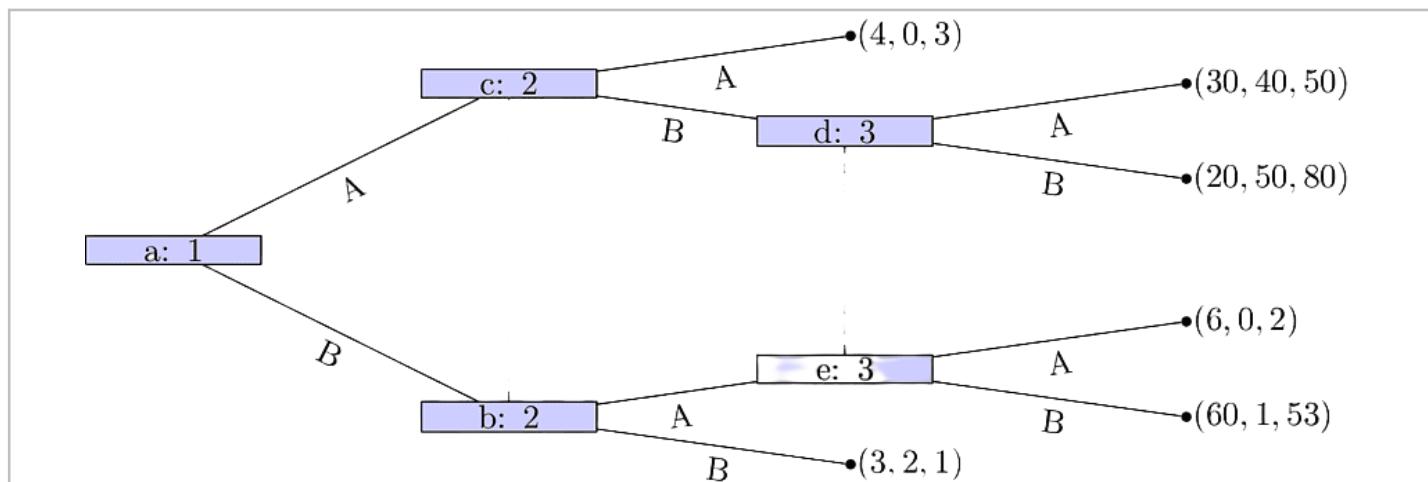
## \ Элементы теории игр. /

В теории игр рассматриваются игры в двух формах. Игры в   форме описываются **матрицей выигрыша** (платежной матрицей). Например, для игры «Числа»: «Два человека независимо друг от друга записывают одно число 1, 2 или 3. Выигрыш равен сумме чисел, причем, если сумма четное число, то выигрывает первый игрок, если нечетное, то второй», матрица выигрыша будет выглядеть следующим образом:

| И1 \ И2 | 1   | 2   | 3   |
|---------|---|---|---|
| 1       | ( <span style="border: 1px solid black; padding: 2px;"> </span> , <span style="border: 1px solid black; padding: 2px;"> </span> ) | ( <span style="border: 1px solid black; padding: 2px;"> </span> , <span style="border: 1px solid black; padding: 2px;"> </span> ) | ( <span style="border: 1px solid black; padding: 2px;"> </span> , <span style="border: 1px solid black; padding: 2px;"> </span> ) |
| 2       | ( <span style="border: 1px solid black; padding: 2px;"> </span> , <span style="border: 1px solid black; padding: 2px;"> </span> ) | ( <span style="border: 1px solid black; padding: 2px;"> </span> , <span style="border: 1px solid black; padding: 2px;"> </span> ) | ( <span style="border: 1px solid black; padding: 2px;"> </span> , <span style="border: 1px solid black; padding: 2px;"> </span> ) |
| 3       | ( <span style="border: 1px solid black; padding: 2px;"> </span> , <span style="border: 1px solid black; padding: 2px;"> </span> ) | ( <span style="border: 1px solid black; padding: 2px;"> </span> , <span style="border: 1px solid black; padding: 2px;"> </span> ) | ( <span style="border: 1px solid black; padding: 2px;"> </span> , <span style="border: 1px solid black; padding: 2px;"> </span> ) |

Игры в   форме характеризуются   (шахматы, шашки, GO и т.д.).

  (англ. Backward induction) - это итеративный процесс рассуждения в обратном направлении от конечных ситуаций в   к актуальному (или стартовому) состоянию игры для решения игр в   форме и вывода последовательности оптимальных действий. Он используется в теории игр для определения наиболее оптимальной последовательности действий, основанной на рациональном поведении.



Также видеоигры зачастую классифицируют по следующим параметрам:

(англ. Time Granularity),  
(англ. Stochasticity),  
(англ. Observability).

Примеры игр из каждой категории:

| Obs                 | Полная информация |           | Частичная информация |           |
|---------------------|-------------------|-----------|----------------------|-----------|
| St   TimeG          | Пошаговая         | Real-time | Пошаговая            | Real-time |
| Есть<br>случайность |                   |           |                      |           |
| Нет<br>случайности  |                   |           |                      |           |

## \ Поиск по дереву. /

В основном утверждалось, что большая часть, если не весь, искусственный интеллект на самом деле является просто поиском. Почти каждая проблема искусственного интеллекта может быть представлена как задача, которая может быть решена путем поиска наилучшего (по некоторым метрикам) плана, пути, модели, функции и т.д.

Поэтому поисковые алгоритмы часто рассматриваются как основа искусственного интеллекта, вплоть до того, что многие учебники (такие как знаменитый учебник Рассела и Норвига) начинаются с рассмотрения поисковых алгоритмов.

– являются базой для решения игр с полной информацией (шахматы, шашки, ...), а также и более сложных игр с природой (когда есть некоторая стохастика – «природа»), и эти алгоритмы имеют такое название, поскольку их можно рассматривать как построение дерева игры, где корнем является узел, представляющий состояние, в котором необходимо найти . Ребра в этом дереве представляют , которые агент предпринимает для перехода из одного узла в другой, а сами узлы представляют игры.

Поиск оптимального хода в играх, для которых построить полное дерево игры и воспользоваться Backward induction очень сложно, зачастую сводится к применению МинМакс алгоритма, состоящего из следующих этапов:

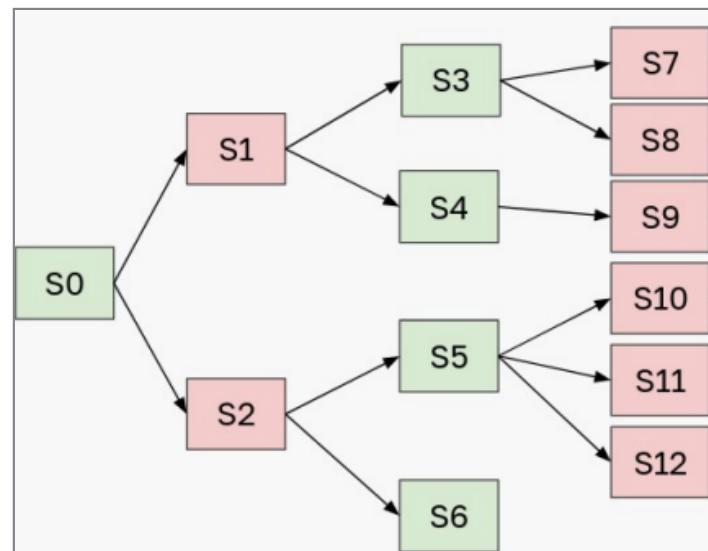
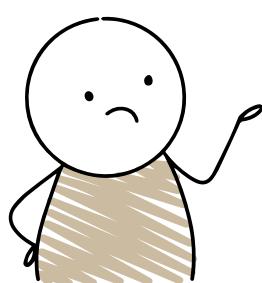
1)

2)

3) Применяем , смысл которой заключается в:

4) Выбираем следующий ход, который приведет к состоянию с наибольшим оценочным значением, полученным после процедуры.

Для построения дерева игры зачастую используются два обхода графов – обход в глубину и обход в ширину. Для примера рассмотрим следующее дерево:



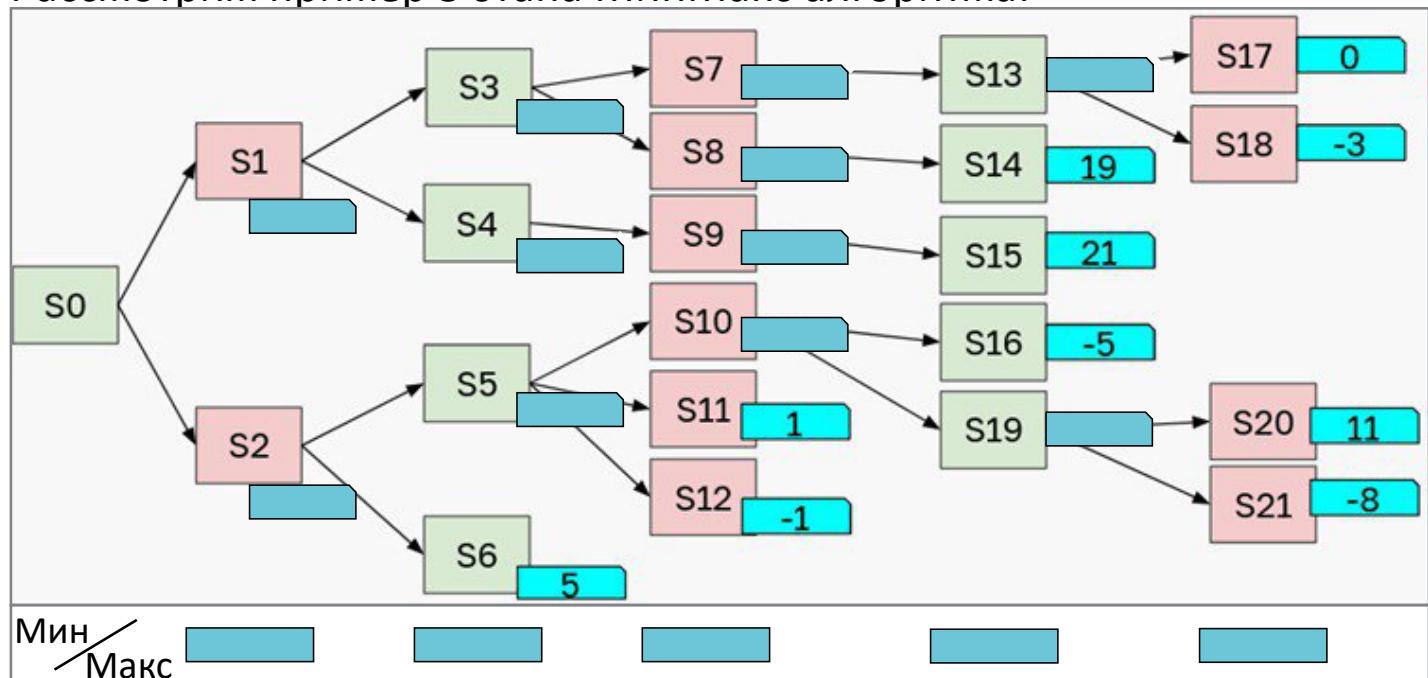
|                |  |  |
|----------------|--|--|
| Порядок обхода | 1) $S_0$ 2) 3) 4)<br>5) 6) 7) 8)<br>9) 10) 11) 12) | 1) $S_0$ 2) 3) 4)<br>5) 6) 7) 8)<br>9) 10) 11) 12) |
| Обход          | В ширину   | В глубину  |

Статическая оценочная функция (СОФ) — это

Например, для игры

, СОФ может считаться как

Рассмотрим пример 3 этапа МинМакс алгоритма:



процедура стратегия поиска хорошего хода. Чтобы сделать процедуру более экономной, необходимо вычислять статические оценки концевых вершин и минимаксные оценки промежуточных вершин одновременно с построением

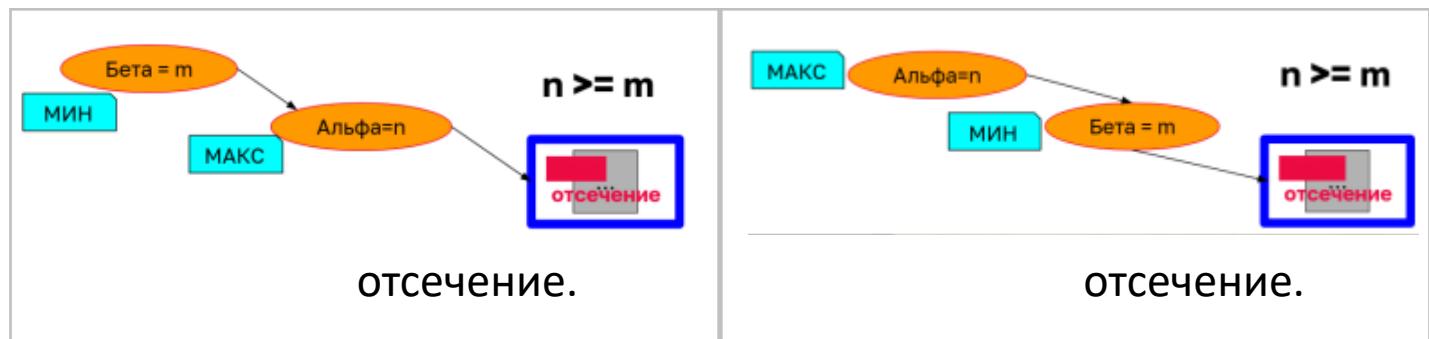
В основе лежит достаточно очевидное соображение: если есть два варианта хода одного игрока, то в ряде случаев можно сразу , не выясняя, насколько

Для применения Альфа-Бета процедуры используется обход в

значение – это значение, получаемое у состояний игры в Альфа-Бета процедуре, в котором ходим мы, а значение – противник.

Альфа отсечение. Правило:

Бета отсечение. Правило:



Альфа-бета процедура всегда приводит к тому же результату ( ), что и простая минимаксная процедура той же глубины.

Score: /4

## Литература:

1. Georgios N. Yannakakis and Julian Togelius. «[Artificial Intelligence and Games](#)» // Springer – 2018
2. Мазалов В. В. «[Математическая теория игр и приложения: Учебное пособие](#)». // 2е изд., стер. — СПб.: Издательство «Лань» – 2016. — 448 с.: ил.
3. Stuart Russell and Peter Norvig. «[Artificial Intelligence: A Modern Approach](#)» // Prentice-Hall, Englewood Cliffs – 1995 (3d edition 2010)
4. Эндрю А. «[Искусственный интеллект](#)» \\Пер. с англ./Под ред. и с предисл. Д. А. Поспелова // Москва: Мир – 1985, с.264
5. Е.И. Большакова, М.Г. Мальковский, В.Н. Пильщиков. «[Искусственный интеллект. Алгоритмы эвристического поиска](#)» // М.: Издательский отдел факультета ВМК МГУ – 2002, с.83

## ПРАКТИЧЕСКАЯ РАБОТА 1. ПОИСК ПО ДЕРЕВУ

В рамках практических работ студентам предоставляется выбор трека: Трек "Закрепление материала", в рамках которого студентам предлагаются не сложные тематические задания (как правило, расчетные); и Трек "Тру прогеры", в рамках которого предлагается одно, но сложное задание с программированием и реализацией той или иной системы по теме лекций. Для трека "Тру прогеры" предоставляются льготы, в виде повышенных баллов и расширенных дедлайнов.

Задания для траектории "Закрепление материала":

I. Палочки. Произведите анализ игры "Палочки" со следующими правилами: Перед 2 игроками лежит 31 палочка на столе, за один ход игрок может взять 1,3 или 4 палочки. Прогревает тот, кто забирает последнюю палочку со стола. Какой игрок победит при условии, что каждый игрок придерживается оптимальной стратегии (игрок №1 ходит первым, №2 вторым)?

1 балл

II. Альфа-бета отсечения. Придумайте некоторое дерево игры (как было в лекции), убедитесь на примере в утверждении "Альфа-бета процедура всегда приводит к тому же результату (наилучшему ходу), что и простая минимаксная процедура той же глубины". Для этого: итеративно проиграйте это дерево с помощью альфа-бета отсечения. Затем: проверьте результат минимаксной процедурой на полном дереве.

2 балла

III. СОФ.

- 1) Выберете и проанализируйте игру;
- 2) Предложите СОФ для этой игры;
- 3) Возьмите несколько (3-4) ситуаций игры и покажите насколько адекватна и применима предложенная СОФ. Игры (можно предложить свою): "Шашки", "Пять в ряд", "Лиса и гуси", "Бридж Ит", "Рассада", "Так-ти克儿", "Цзяньшидзы", "Щелк".

4 балла

Задание для траектории "Тру прогеры":

Выбрать игру и реализовать ее на языке высокого уровня с интерфейсом с ИИ на основе Альфа-Бета отсечений. Например, игры: "Шашки", "Пять в ряд", "Лиса и гуси", "Бридж Ит", "Рассада", "Так-ти克ль", "ЦЗЯНЬШИДЗЫ", "Шестнадцать солдат", "Пентамино", "Щелк". Вы можете предложить свою.

8+3 балла

*Score:* /8

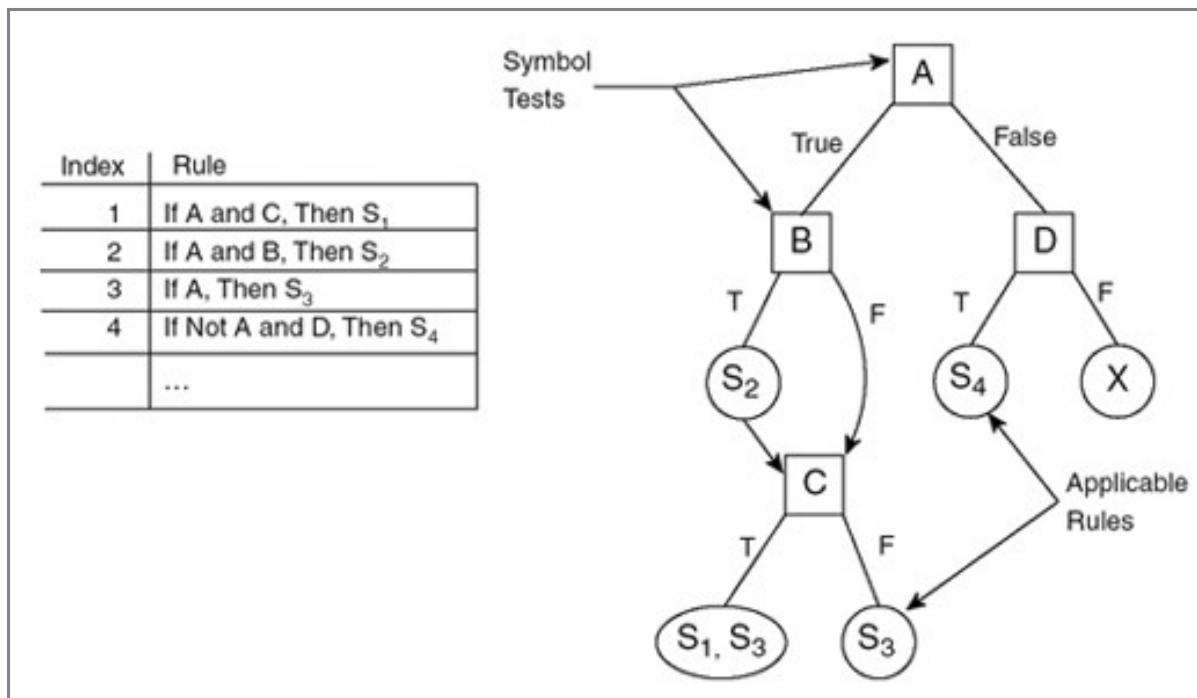
Ссылка на отчёт или гит-репозиторий:

## ЛЕКЦИЯ 2. AD-HOC BEHAVIOR AUTHORING.

\ Разработка специального поведения. /

Ad-hoc Behavior Authoring – это

Системы на основе правил — это



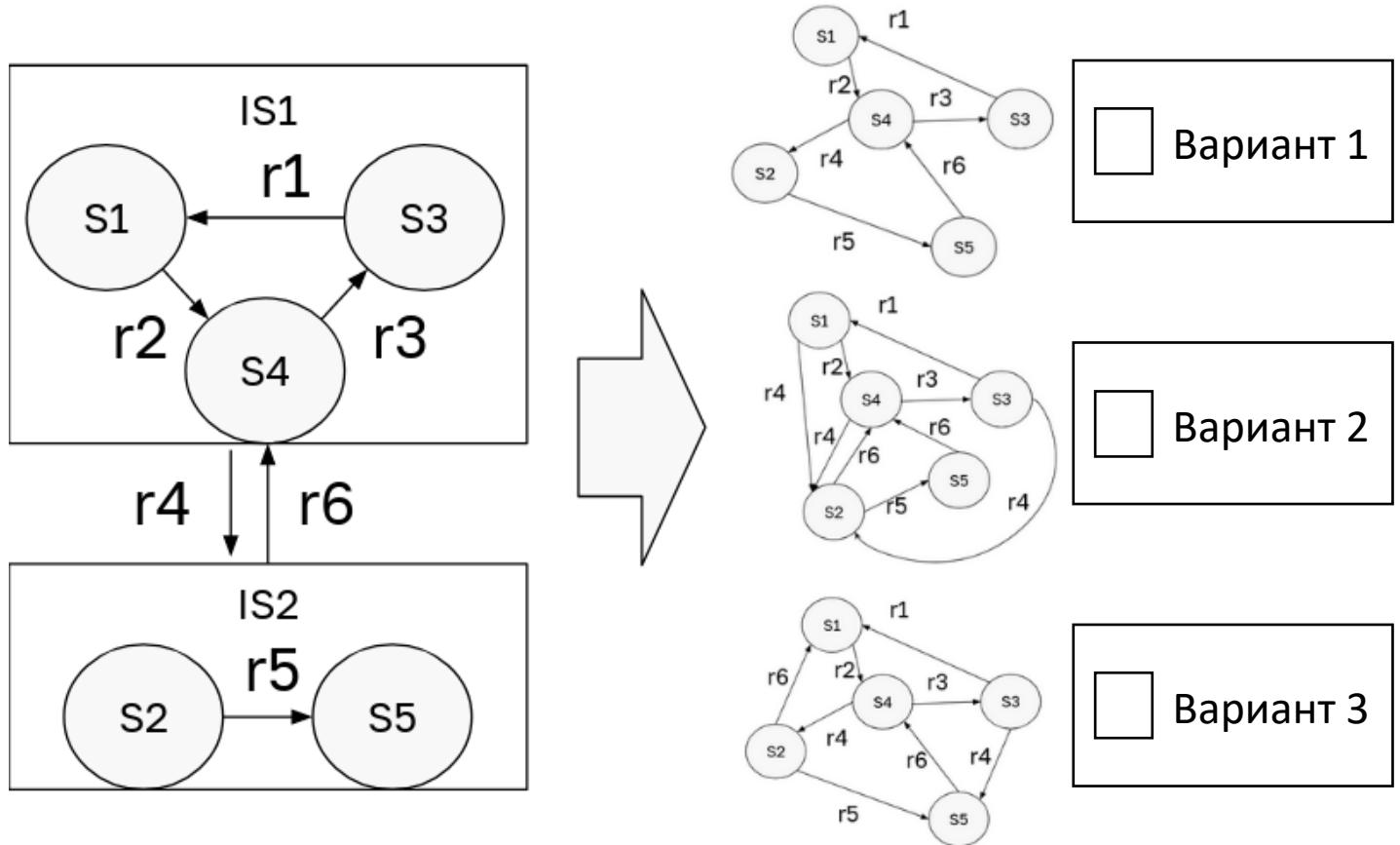
Скриптинг используется для

Конечный автомат (КА) — это

Иерархический конечный автомат (ИКА) позволяет

Однако, любой иерархический конечный автомат можно представить в виде «одномерного» конечного автомата, но переходов будет очень много.

Например (нужно выбрать правильный КА на основе ИКА):



**Дерево поведения** – это

Как правило, рассматривают две основные композитные вершины в деревьях поведения:

1. Последовательность (англ. sequence) — это вершина

2. Селектор (англ. selector) — это вершина

## Декоратор (англ.

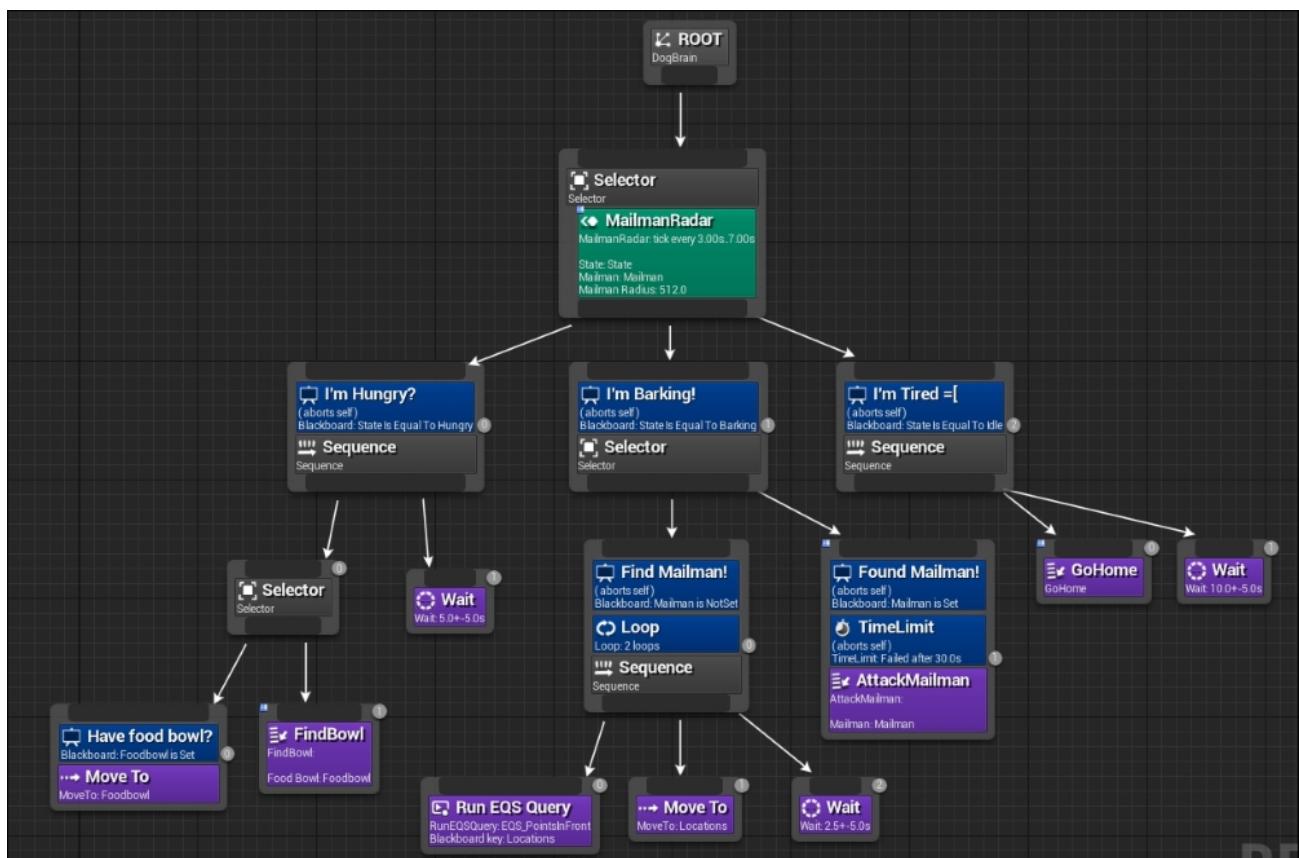
) также одна из важнейших

вершин в деревьях поведения, которая:

Декоратор, в классическом/академическом понимании, может иметь лишь одну дочернюю вершину. Примеры классических декораторов в деревьях поведения:

Вершина

(англ. Parallel) позволяет



Используя комбинацию

и

позволяет создавать сложные и адаптивные

системы поведения.

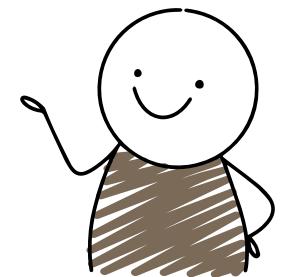
предоставляют гибкую иерархическую структуру для определения логики поведения, тогда как обеспечивают переключение между состояниями и контролль высокоуровневого поведения. Это позволяет разработчикам игр определять различные сценарии поведения ИИ, реагировать на изменяющиеся условия игры и создавать непредсказуемое и интересное поведение в игровом мире.

# \ Элементы генерации контента. /

Интеллектуальная генерация контента в играх - это

Примеры того, что можно генерировать:

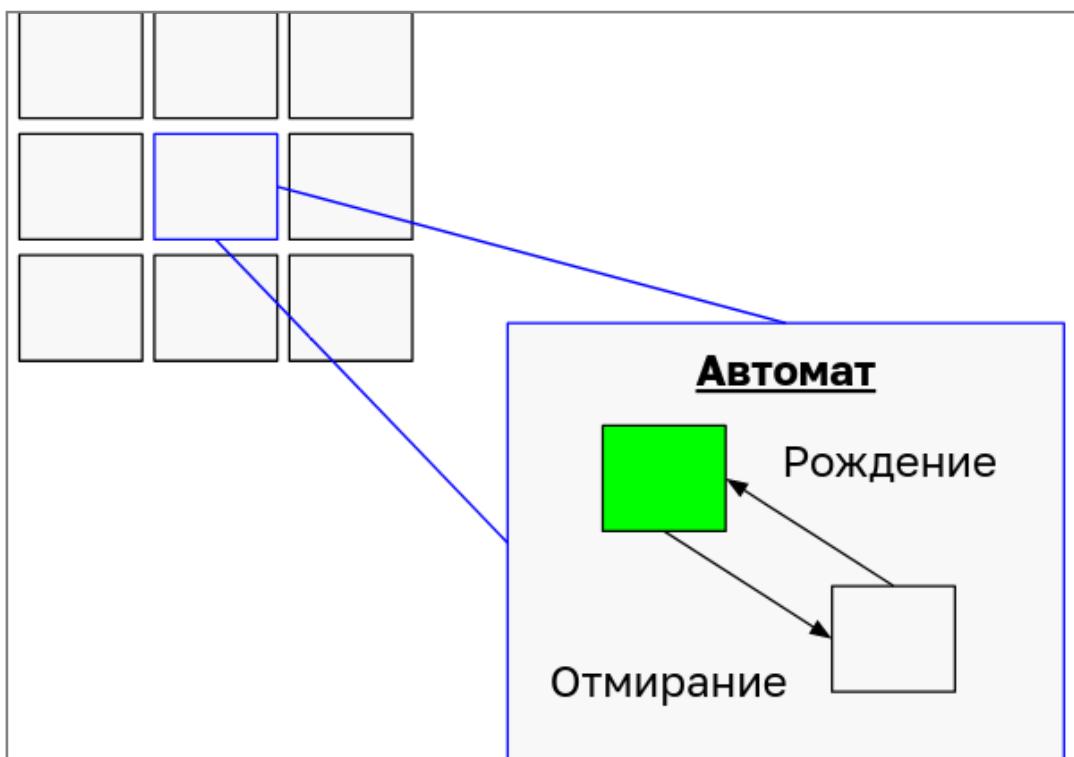
Одной из важных задач в играх является автоматическая генерация комнат (подземелий). И одним из популярных алгоритмов такой генерации — применение клеточных автоматов.



**Клеточные автоматы** – модель, в которой

Примеры клеточных автоматов:

| Правила \ Название |  |  | Игра жизнь          |
|--------------------|--|--|---------------------|
| Клетка рождается   |  |  | При 3с.             |
| Клетка живёт       |  |  | При 2-3с.           |
| Клетка умирает     |  |  | В остальных случаях |



## Обобщенный алгоритм для генерации комнат на основе клеточных автоматов:

| № | Этап | Описание |
|---|------|----------|
| 1 |      |          |
| 2 |      |          |
| 3 |      |          |
| 4 |      |          |
| 5 |      |          |

После генерации подземелья (3 этапа) могут возникнуть отдельные комнаты без переходов, которые необходимо как-то соединить. Для этого можно решать различные задачи, например, **задачу Коммивояжера** или **задачу поиска минимального остовного дерева**. Чтобы решить эту задачу необходимо граф, например, как центры отдельных комнат. Последнее можно применить, например, **Алгоритм Прима**, который состоит из следующих этапов:

1)

2)

3)

4)

*Score:* /2

## **Литература:**

1. Georgios N. Yannakakis and Julian Togelius. «[Artificial Intelligence and Games](#)» // Springer – 2018
2. Brian Schwab. «[AI game engine programming](#)» // Hingham: Charles River Media – 2004
3. Alex J. Champandard, «AI Game Development: Synthetic Creatures with Learning and Reactive Behaviors» – 2003
4. Fernando Bevilacqua. «[Finite-State Machines: Theory and Implementation](#)» // Internet Source – 2013
5. Sekhavat, Yoones A. «[Behavior trees for computer games.](#)» // International Journal on Artificial Intelligence Tools 26.02 – 2017
6. Chris Simpson. «[Behavior trees for AI: How they work](#)» // Internet Source – 2014
7. В.С. Вивденко, О.М. Копытова. «[Обзор актуальных методов процедурной генерации комнат в видеоиграх](#)» // Internet Source
8. Habr, TLHE. «[10 удивительно зрелищных простейших клеточных автоматов](#)» // Internet Source – 2023
9. Habr, Patient Zero «[Обзор техник реализации игрового ИИ](#)» // Internet Source – 2018 (Оригинал: Ben Sizer. «[The Total Beginner's Guide to Game AI](#)»)
10. Mark D. «[Behavioral mathematics for game AI](#)». // Course Technology Cengage Learning – 2009.

# ЛАБОРАТОРНАЯ РАБОТА 1. НАВИГАЦИЯ В UE4

1. Создать уровень в UE4. На одной части уровня создать бот-куб (или бот-манекен), который способен пройти лабиринт с помощью линейного лазера (или нескольких), который вы построите. В данной задаче не должно быть использовано прохождение лабиринта с помощью таких нод, как Move to или иных, простраивающих маршрут до конечной точки с помощью Nav Mesh. Бот должен самостоятельно дойти до конца. Лабиринт должен содержать минимум 3 поворота.

2 балла

2. На другой части уровня должен быть реализован игрок-манекен, который будет передвигаться к точке, которую игрок укажет на карте нажатием левой кнопки мыши. В реализации можно использовать любые изученные вами способы. Камера в проекте выставляется на усмотрение студента.

2 балла

3. Добавить на уровень бот-манекен, который будет следовать за основным игроком (задание №2). Бот должен не терять игрока, когда тот находится в прыжке, а также не останавливать дальнейшее преследование, если бот достигнет игрока. Можно использовать любую из нод следования за игроком, которая будет соответствовать требованиям.

2 балла

4. Добавить бот-манекен, который циклично и непрерывно следует между четырьмя точками, выставленными на уровне заранее (модель патрулирования: бот не должен останавливаться после достижения последней точки, а бежать снова к первой).

2 балла

2 балла за выполнение дедлайна

*Score:* /10

Ссылка на отчёт или гит-репозиторий:

## ЛАБОРАТОРНАЯ РАБОТА 2. ДЕРЕВЬЯ ПОВЕДЕНИЙ В UE4

1. Создать уровень и на данном уровне выполняются оба задания на отдельных аренах. Разработать специальное поведение для бота: разработать дерево поведений, где бот последовательно будет бегать между четырьмя точками бесконечно. Бег к каждой точке должен быть отдельной новой задачей с успешным завершением, так как должна быть возможность менять задачи местами, то есть менять маршрут следования бота. Новая задача - это отдельно созданная задача, которой изначально не было в списке Tasks, и которая прописана в отдельном blueprint. Декораторы не должны быть использованы. Когда бот добегает до каждой точки, в вывод Print String на экране слева должна быть выведена строка с текстом "Добежал"

4 балла

2. На второй арене создать бота с "Искусственным интеллектом": спроектировать и реализовать дерево поведений, содержащее не менее четырех типов задач из списка Tasks в движке. Эти задачи должны логично и последовательно переходить друг в друга, не обрывать логику на середине. Две задачи типа Wait, или иные, считаются за одну, но могут быть добавлены в логике сколько угодно раз для достижения конечного результата. Также можно добавлять собственные задачи для дополнения или связки логики, но учитываться будет только использование готовых задач из списка. Можно использовать декораторы по желанию

4 балла

2 балла за выполнение дедлайна

*Score:* /10

Ссылка на отчёт или гит-репозиторий:

## ЛЕКЦИЯ 3. НАВИГАЦИЯ И ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ ИГРОВОГО ИИ.

### \ Навигация. /

Навигация – одна из главных частей реализации видеоигр. В различных стратегиях игроки указывают своим персонажам точку назначения, а система (игра) должна проложить оптимальный маршрут, то есть по-другому решить задачу .

Для решения этой задачи, в основном, используются следующие три алгоритма:

- 1)
- 2)
- 3)

**Алгоритм Дейкстры** — один из главных алгоритмов поиска кратчайшего пути. Данный алгоритм ищет кратчайший путь от одной вершины графа до . Алгоритм Дейкстры состоит из следующих этапов:

- 1)
- 2)
- 3)
- 4)
- 5)

**Алгоритм А\*** является модификацией Алгоритма Дейкстры, в котором

Примеры эвристик:

## \ Технологии программирования ИИ. /

Реализация навигации может быть представлена в различных структурах. Основными являются навигационная сетка (navmesh) и навигационный.

более уместен, когда навигационные данные генерируются автоматически, а может быть более предпочтительным, когда требуется ввод данных левел дизайнеров. Это приводит к необходимости упрощения создания и обслуживания навигационного графа, когда он подлежит ручному изменению.

Однако, хочется, чтобы движение было правдоподобным. Для этого существует технология , которая отвечает за реалистичность движения.

Для того, чтобы наши интеллектуальные агенты могли моделировать похожее поведение из жизни, они должны иметь возможность «по-настоящему» воспринимать мир. Следующий AI Perception реализован в Unreal Engine 4 (указать истинное):

- |  |  |
|--|--|
| <input type="checkbox"/> AI Damage sense   | <input type="checkbox"/> AI Prediction |
| <input type="checkbox"/> AI Sight          | <input type="checkbox"/> AI Instinct   |
| <input type="checkbox"/> AI Sense of smell | <input type="checkbox"/> AI Save       |
| <input type="checkbox"/> AI Pain           | <input type="checkbox"/> AI Hearing    |
| <input type="checkbox"/> AI Touch          | <input type="checkbox"/> AI Team       |

Чтобы работать с информацией о мире или динамической информацией для агентов используют следующие технологии (статья Ben Sizer (рус. Набр, Patient Zero) для получения больших подробностей):

I. Тэги —

II. Умные объекты —

III. Blackboard —

IV. Influence Maps —

V. EQS —

## \ Планирование действий в среде. /

**Goal-Oriented Action Planning (GOAP)** – это

Процесс планирования в GOAP обычно состоит из следующих шагов:

1. Определение текущего состояния:

2. Определение возможных действий:

3. Планирование последовательности действий:

4. Выполнение плана:

То есть в результате мы получаем для достижения соответствующей ситуации цели.

**Hierarchical Task Network (HTN) planning** – это

В конкретных реализациях планирования в игровом ИИ, элементы HNT и GOAP могут сосуществовать, и разработчики часто комбинируют их, чтобы достичь необходимого уровня гибкости, или варьируют подходы в зависимости от задачи и требований игры.

**Score:** /2

## **Литература:**

1. Georgios N. Yannakakis and Julian Togelius. «[Artificial Intelligence and Games](#)» // Springer – 2018
2. Brian Schwab. «[AI game engine programming](#)» // Hingham: Charles River Media – 2004
3. Habr, Patient Zero «[Обзор техник реализации игрового ИИ](#)» // Internet Source – 2018 (Оригинал: Ben Sizer. «[The Total Beginner Guide to Game AI](#)»)
4. Micael DaGraca. «[Practical Game AI Programming](#)» // Packt Publishing – 2017
5. WikITMO «[Алгоритм A\\*](#)» // Internet Source
6. Gaiiden «[Navigation Graph Generation](#)» // Internet Source – 2011
7. Habr, PatientZero «[Streeng Behaviors](#)» // Internet Source – 2018
8. Documentation Unreal Engine 4 «[AIPerception](#)» // Internet Source
9. Максим Дорофеев «Джедайские техники. Как воспитать свою обезьяну, опустошить инбокс и сберечь мыслетопливо» // 368 стр. – 2018

## ПРОЕКТ. ПРИМЕНЕНИЕ МЕТОДОВ ИГРОВОГО ИИ В UE4

Групповой проект: Game Designer, Game Developer, Game AI Developer

1. Придумать небольшой уровень в игре стратегического формата с врагами и союзниками (примеры: Tower defense, зачистка карты, прохождение от одного места в другое через врагов).

2. Создать небольшую арену уровня, причём арена должна быть осмысленной, не пустой и следовать логике разработанной геймдизайнером игры. Особых требований к визуальному дизайну нет (на усмотрение студентов).

3. Игрок должен уметь двигаться по щелчку мышки в нужное место.

4. Разработать минимум два типа ботов-врагов (с разной логикой поведения). Враги могут спавниться или находиться на определенных местах на карте, если это обусловлено изначальной идеей.

5. Должен быть минимум один союзник со своим поведением, который бегает с основным игроком.

6. В рамках проектирования специального поведения врагов и союзников должны быть использованы обе модели: деревья поведений и конечные автоматы. Также враги и союзники должны иметь зрение и/или слух.

7. Опционально: использовать карты влияния от дополнительных объектов и EQS.

8. Опционально: применение других методов игрового ИИ, которые рассматриваются на лекциях (системы на основе правил/скрипting, иерархические автоматы и модели, ИИ на основе полезности, системы нечеткого вывода, нечеткие модели, эволюционные и роевые алгоритмы).

*Score:* /30

Ссылка на отчёт или гит-репозиторий:

## ЛЕКЦИЯ 4. ПРИНЯТИЕ РЕШЕНИЙ И ЭЛЕМЕНТЫ МАШИННОГО ОБУЧЕНИЯ.

### \ Принятие решений. /

#### Теория принятия оптимальных решений

Методы и подходы к принятию решений зависят от состояния окружения:

#### Модели принятия решений



- 1) – нет природы. Природа – это
- 2) – мы ничего не знаем о состояниях природы.
- 3) **Условия риска –**

В условиях риска, одним из популярных критериев для принятия решений является максимизация ожидаемого результата, выраженного в виде значения .

**Теория полезности** рассматривает вопросы построения процедур для определения предпочтений в количественной форме.

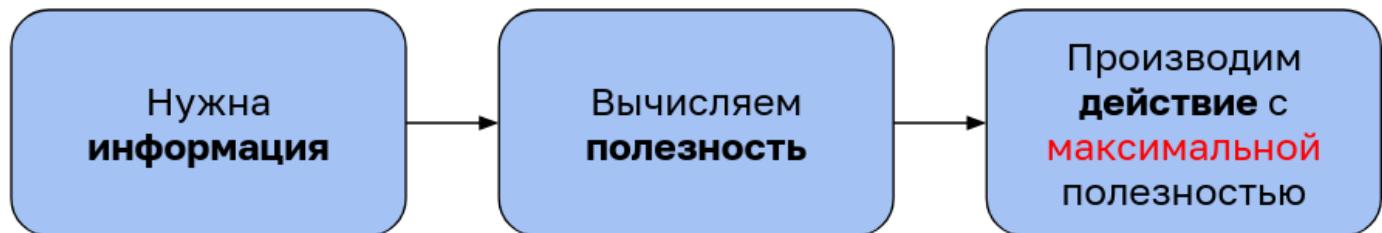
## Полезность (англ.

) может измерять что угодно – от

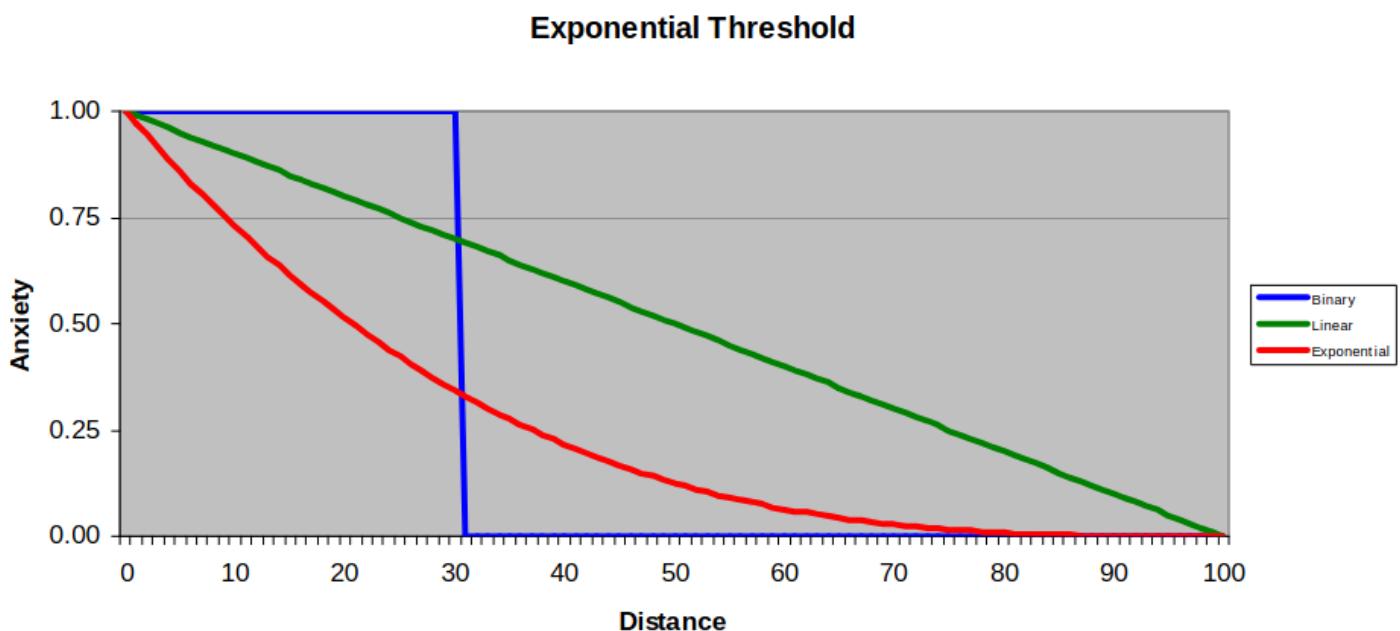
до

Различные полезные сведения о возможных действиях или решениях могут быть объединены в линейные или нелинейные формулы и направлять агента к принятию решений на основе агрегированной полезной информации.

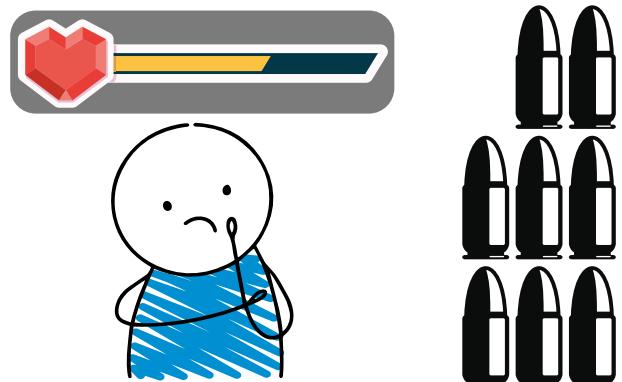
### \ Utility based AI. /

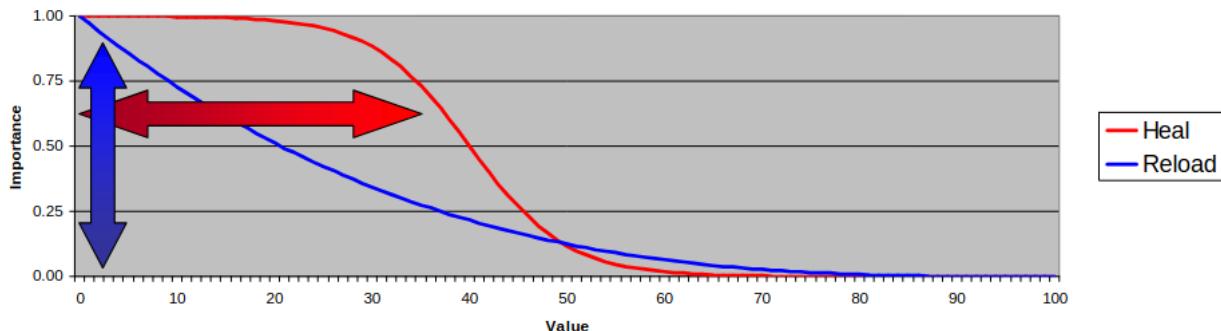


Полезность в Utility based AI может быть задана, как таблично, правилами, так и функциями полезности: например, логистическими, линейными или экспоненциальными.



Рассмотрим пример. Есть агент, у которого конечный запас патронов и здоровья. Тогда определим функции полезности:





1. По мере того как здоровье ухудшается, необходимость в лечении возрастает.
2. При этом следим, чтобы у нас не было совсем мало здоровья.
3. По мере того как количество боеприпасов уменьшается, необходимость в перезарядке возрастает.
4. Срочность достигает максимума, когда у нас заканчиваются боеприпасы.

| Параметры  | Здоровье | Патроны |
|------------|----------|---------|
| Значение   | 50       | 25      |
| Полезность |          |         |

Тогда, на основе полезности, наш агент должен:

Для усложнения логики игрового ИИ на основе полезности, вводят **композитные действия/состояния**, полезность которых вычисляется, как некоторое арифметическое действие с гиперпараметрами.

Для нашего случая можем ввести действие «Уйти в укрытие», и его полезность, например, будет вычисляться, как:

## \ Элементы машинного обучения. /

Технологии машинного обучения:

1. **Обучение с учителем** (англ.  $\text{Supervised Learning}$ ) –
2. **Обучение без учителя** (англ.  $\text{Unsupervised Learning}$ ) –

### 3. Обучение с подкреплением (англ.

) – это

инструмент для решения задач

через построение , которые

взаимодействуя , и получают

как обратную связь.

Почему целью агента является максимизация ожидаемой прибыли?

Потому что RL основан на **гипотезе вознаграждения**, которая заключается в том, что

Вот почему при обучении с подкреплением, чтобы добиться наилучшего поведения, мы стремимся научиться предпринимать действия, которые



**Среда** (англ. ) – пространство, в котором агент производит свои действия. Как правило, в RL среда может быть с описанием (или с полной информацией, например, шахматы и шашки), а может быть с (например, марио).

**Действия** (англ. ) – пространство доступных действий в среде у агента. Может быть как **конечным** количеством (например, направо и налево), или **бесконечным** (например, работа с некоторой шкалой).

**Кумулятивное вознаграждение** на каждом временном шаге  $t$  может быть записано как:

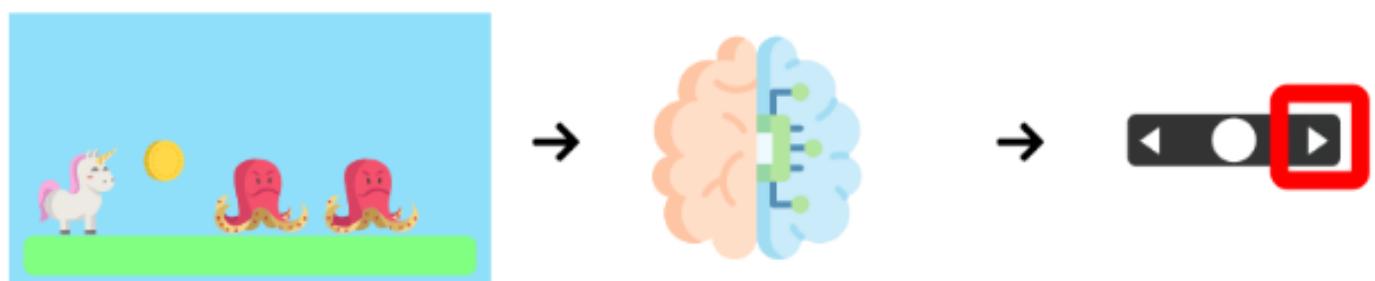
$$R(\tau) = r_{t+1} + r_{t+2} + r_{t+3} + r_{t+4} + \dots$$


Trajectory (read Tau)  
Sequence of states and actions

Однако, на самом деле мы не можем просто добавить их таким образом. Награды, которые приходят раньше (в начале игры), , поскольку они более предсказуемы, чем . Следовательно дальняя награда, даже если она потенциально больше, будет более дешевой, поскольку мы на самом деле не уверены, что сможем получить ее. Для этого добавим скидку на вознаграждение:

$$R(\tau) = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + \dots$$

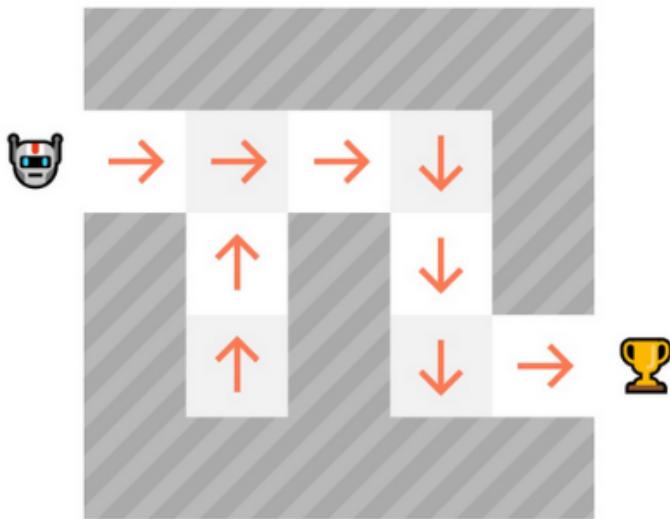

Trajectory (read Tau)  
Sequence of states and actions



**State** →  $\pi(\text{State}) \rightarrow \text{Action}$

Существует два подхода к обучению нашего агента поиску этой оптимальной политики (стратегии)  $\pi^*$ :

1. Непосредственно обучая агента тому, какие действия следует предпринять, учитывая текущее состояние:
2. Косвенно обучая агента узнавать, какое состояние является более ценным, а затем предпринимать действия, которые приводят к более ценным состояниям:



метод



метод

Для того, чтобы получить политику при полученной информации о значениях, используется:

$$\pi^*(s) = \arg \max_a Q^*(s, a)$$

При обучении системы возникает вопрос в том, а когда исследовать среду, пробуя новые действия, а когда идти по накатанной и использовать уже имеющиеся знания. Одним из простых методов является эпсилон-жадные «политики» (англ. Policy), смысл которых заключается в:

**Q-Learning** - это популярный базовый алгоритм RL, который:

- Обучает Q-функцию (функцию действия-значения),
- 
- 
- Учитывая состояние и действие, наша Q-функция выполнит поиск соответствующего значения в своей Q-таблице.
- По завершении обучения у нас будет **оптимальная Q-функция**, что означает, что у нас есть оптимальная Q-таблица.
- И если у нас есть оптимальная Q-функция, то у нас есть оптимальная политика, поскольку мы знаем, какие действия лучше всего предпринять в каждом состоянии.

---

**Algorithm 14:** Sarsamax (Q-Learning)

---

**Input:** policy  $\pi$ , positive integer  $num\_episodes$ , small positive fraction  $\alpha$ , GLIE  $\{\epsilon_i\}$   
**Output:** value function  $Q$  ( $\approx q_\pi$  if  $num\_episodes$  is large enough)  
Initialize  $Q$  arbitrarily (e.g.,  $Q(s, a) = 0$  for all  $s \in \mathcal{S}$  and  $a \in \mathcal{A}(s)$ , and  $Q(terminal-state, \cdot) = 0$ )  
**for**  $i \leftarrow 1$  **to**  $num\_episodes$  **do** Step 1  
     $\epsilon \leftarrow \epsilon_i$   
    Observe  $S_0$   
     $t \leftarrow 0$   
    **repeat**  
        Choose action  $A_t$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy) Step 2  
        Take action  $A_t$  and observe  $R_{t+1}, S_{t+1}$  Step 3  
         $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t))$  Step 4  
         $t \leftarrow t + 1$   
    **until**  $S_t$  is terminal;  
**end**  
**return**  $Q$

---

**Score:**

/4

## Литература:

1. Thomas Simonini, Omar Sanseviero, Sayak Paul «[Deep RL Course](#)» // Huggingface, Internet Source. – 2023.
2. Dave Mark, Kevin Dill «[Improving AI Decision Modeling Through Utility Theory](#)» // AI Summit GDC, Internet Source. – 2010.
3. Georgios N. Yannakakis and Julian Togelius. «[Artificial Intelligence and Games](#)». Springer, 2018
4. Черноиванова Е. А. «Исследование операций и методы оптимизации.» – 2014.
5. Лаборатория системного анализа «[Исследование операций и оптимизация](#)» // Internet Source. – 2017.
6. Майзер Х., Эйджин Н., Тролл Р. и др. “Исследование операций” в 2 томах. Т1: “Методологические основы и математические методы”, 792 стр., 1978
7. Jakob Rasmussen, «[Utility AI](#)» // Internet Source. – 2016.
8. Высшая Школа Цифровой Культуры Университета ИТМО «Продвинутое машинное обучение. Обучение с подкреплением» // Internet Source. – 2021.

## ПРАКТИЧЕСКАЯ РАБОТА 2. ПРИНЯТИЕ РЕШЕНИЙ

Задания для траектории "Закрепление материала":

I. Utility AI. Спроектируйте ИИ на основе полезности для некоторого NPC: Описание NPC, действия, функции полезности (1 балл). Есть хотя бы одно композитное действие (1 балл). Приведено несколько (от 5) примеров при различных входных данных (1 балл).

3 балла

II. Q-learning. Изучите правила и запустите моделирование на ресурсе <https://www.mladdict.com/q-learning-simulator> на 15-20 эпохах. Сохраните полученную матрицу. Приведите 1-2 итерации (не эпохи!) ручным счетом (как в лекции) (2 балла). Обучите агента 2-3 раза с нуля на сервисе с разными настраиваемыми гиперпараметрами на 10-20 эпохах. Сохраняйте результирующие матрицы. Сделайте вывод о влиянии гиперпараметров. Для каждой результирующей матрицы постройте матрицу "политик" и цепочку состояний и действий на основе этих политик от начального состояния до конечного состояния (2 балла).

4 балла

Задание для траектории "Тру прогеры":

Опция 1. Utility AI (+1 доп.балл). Спроектируйте Utility AI для некоторого NPC и реализуйте в Unreal Engine 4. Продемонстрируйте работоспособность разработанного ИИ.

Опция 2. Q-learning. Реализация Q-learning на Python и применение к двум окружениям. Описание задания и шаблон для решения: <https://huggingface.co/learn/deep-rl-course/unit2/hands-on>.

7+1 балла

*Score:* /7

Ссылка на отчёт или гит-репозиторий:

## ЛЕКЦИЯ 5. НЕЧЕТКИЕ МОДЕЛИ И СИСТЕМЫ.

### \ Теоретические основы нечеткости. /

Лингвистическая переменная – это переменная, значение которой определяется набором вербальных (то есть словесных) характеристик некоторых свойств, называемых термами.

Например:

|                |  |  |  |
|----------------|--|--|--|
| Лингв. перем.: |  |  |  |
| Термы:         |  |  |  |

Универсальное множество  $U$ . Пусть  $A$  – подмножество универсального множества. Принадлежность любого элемента  $x$  подмножеству  $A$  можно выразить с помощью функции принадлежности  $\mu_A(x)$ .

Математический объект, определяемый выражением

$$A = \left\{ \frac{p_1}{x_1} + \frac{p_2}{x_2} + \frac{p_3}{x_3} + \dots + \frac{p_n}{x_n} \right\}$$

Где  $x_i \in U$ ,  $p_i \in [0,1]$  (значение функции принадлежностей),  $n$  – количество элементов в  $A$ , будем называть нечетким подмножеством множества  $U$ .

Популярные типы **функций принадлежностей** (ФП), используемые в теории нечетких множеств:

- 1)
- 2)
- 3)

Для дизайна функций принадлежностей существуют методы, как на основе данных и знаний о системе, так и различные экспертные методы. Одним из самых простых методов является опрос экспертов в формате согласен/не согласен, состоящий из следующих этапов:

1. Подготовительный. Определить лингвистическую переменную, термы и возможные варианты/диапазоны значений.
2. Опрос экспертов.

### 3. Определение ФП.

Операции над нечеткими множествами:

$$1) = \sqrt{[\mu_A(x)]}$$

$$2) = \max(\mu_A, \mu_B)$$

$$3) = \mu_A * \mu_B$$

$$4) = [\mu_A(x)]^2$$

$$5) = \min(\mu_A, \mu_B)$$

$$6) = \frac{\mu_A(x)}{\max\{\mu_A(x)\}}$$

$$7) = \mu_A + \mu_B - \mu_A * \mu_B$$

| Операция                        | Минмаксный подход         | Вероятностный подход |
|---------------------------------|---------------------------|----------------------|
| НЕ                              | $\bar{\mu}_A = 1 - \mu_A$ |                      |
| И                               |                           |                      |
| ИЛИ                             |                           |                      |
| Нормализация                    |                           |                      |
| Квантификатор.<br>Концентрация. |                           |                      |
| Квантификатор.<br>Растяжение.   |                           |                      |

## \ Нечеткие отношения и отображения. /

**Отношение** (четкое) из множества А в множество В —

**Нечеткое (бинарное) отношение** —

Одним из популярных способов задания нечетких отношений — **матрица (таблица) нечеткого отношения**.

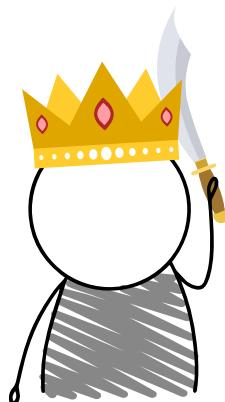
Формула нечеткого отображения:

$$(B) = \left\{ \vee_{a_i \in A} (\wedge(\mu_A(a_i), \mu_r(a_i, b_j))) \mid b_j \in U_B \right\}$$

Использование формулы:

Имеется нечеткое отношение  $r$ , заданное таблицей ниже.

| $r$  | Огненный шар | Агула | Волшебный кулак |
|--|--------------|-------|-----------------|
|   | 1,0          | 0,3   | 0,6             |
|   | 0,0          | 1,0   | 0,6             |
|   | 0,8          | 0,0   | 0,3             |
|  | 0,2          | 0,0   | 0,6             |



Выбор метода подсчёта:

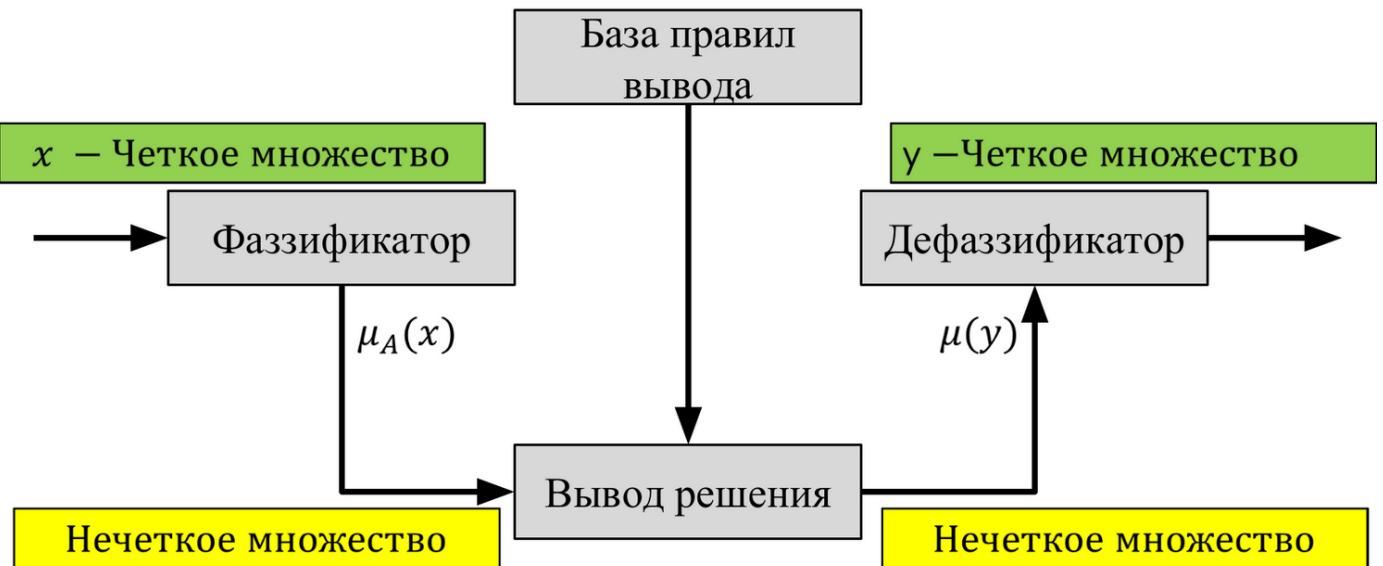
Нечеткое отображение:

$$r \left( \frac{0,2}{\text{[Portrait of a silver-skinned character]}} + \frac{0,5}{\text{[Portrait of a bald man with a red cloth]}} + \frac{0,2}{\text{[Portrait of a bird-like character]}} \right) = \{ \text{[Icon of a fireball]} + \text{[Icon of a portrait]} + \text{[Icon of a hand]} \}$$

\ Нечеткие автоматы и системы нечеткого вывода. /

Нечёткий автомат –

Система нечеткого вывода –



Процесс фаззификации –

Процесс дефаззификации –

Примеры дефаззификации:

1)

2)

Существует огромное количество алгоритмов нечеткого вывода, например: Алгоритм Мамдани:

1)

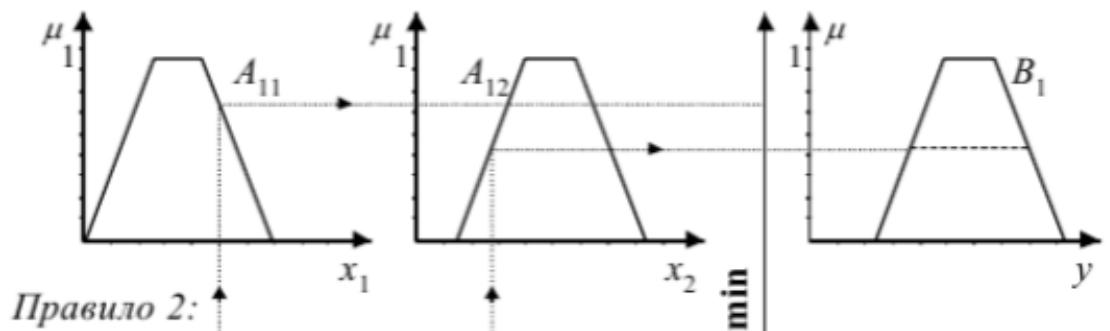
2)

3)

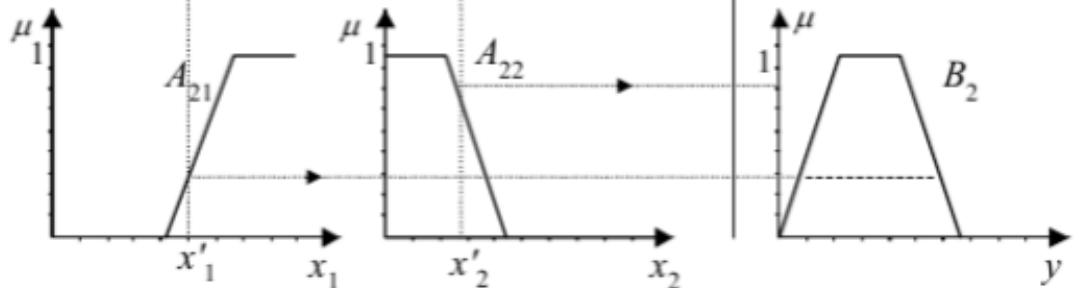
4)

5)

*Правило 1:*

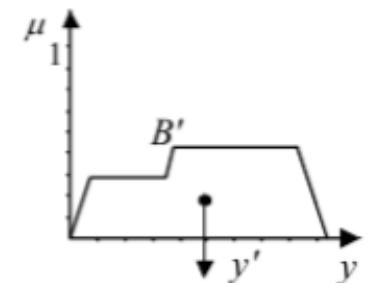


*Правило 2:*



$\Pi_1$ : ЕСЛИ  $x_1$  есть  $A_{11}$  И  $x_2$  есть  $A_{12}$ , ТО  $y$  есть  $B_1$ ,

$\Pi_2$ : ЕСЛИ  $x_1$  есть  $A_{21}$  И  $x_2$  есть  $A_{22}$ , ТО  $y$  есть  $B_2$ .



*Score:*

/4

## Литература:

1. Борисов В. В., Круглов В. В., Федулов А. С. "Нечеткие модели и сети." – 2-е изд., стереотип. – М.:Горячая линия–Телеком, 2018 – 284 с.: ил.
2. Alex J. Champandard, «AI Game Development: Synthetic Creatures with Learning and Reactive Behaviors» – 2003
3. Bourg D. M., Seemann G. «[AI for game developers](#)». // « O'Reilly Media, Inc.» – 2004.
4. Kose U. «[Developing a fuzzy logic based game system](#)» // Computer Technology and Application. – 2012. – Т. 3. – №. 7. – С. 510-517.
5. Pirovano M., Lanzi P. L. «[Fuzzy Tactics: A scripting game that leverages fuzzy logic as an engaging game mechanic](#)» // Expert Systems with Applications. – 2014. – Т. 41. – №. 13. – С. 6029-6038.

6. Pirovano M. «[The use of fuzzy logic for artificial intelligence in games](#)» // University of Milano, Milano. – 2012.
7. Vorachart V., Takagi H. «[Evolving fuzzy logic rule-based game player model for game development](#)» // International Journal of Innovative Computing, Information and Control. – 2017. – T. 13. – №. 6. – C. 1941-1951.
8. Cord O. et al. «[Genetic fuzzy systems: evolutionary tuning and learning of fuzzy knowledge bases](#)». // World Scientific, 2001. – T. 19.
9. Li Y., Musilek P., Wyard-Scott L. «[Fuzzy logic in agent-based game design](#)» // IEEE Annual Meeting of the Fuzzy Information, 2004. Processing NAFIPS'04. – IEEE, 2004. – T. 2. – C. 734-739.
10. Ismail N. et al. «[Optimization of Enemy's Behavior in Super Mario Bros Game Using Fuzzy Sugeno Model](#)» // Journal of Physics: Conference Series. – IOP Publishing, 2018. – T. 1090. – №. 1. – C. 01206.
11. Shaout A., King B. W., Reisner L. A. «[Real-Time Game Design of Pac-Man Using Fuzzy Logic](#)» // Int. Arab J. Inf. Technol. – 2006. – T. 3. – №. 4. – C. 315-325.
12. ИИ и МО. «[AIML-3-1 Нечеткие множества.](#)» // Internet Source – 2015

# ПРАКТИЧЕСКАЯ РАБОТА 3. НЕЧЕТКИЕ МОДЕЛИ И СИСТЕМЫ

Задания для траектории "Закрепление материала":

I. Операции над нечеткими множествами. Выберите лингвистическую переменную, определите термы (не менее 3 штук). Проведите построение ФП на основе экспертных оценок. Примените все изученные операции (тип по выбору (М или В)).

1 балл

II. Нечеткие отношения. Задайте нечеткое отношение по игровой тематике (например, Тип орка: Ловушки). Сделайте 2-3 примера применения.

1 балл

III. Аналогии. Задайте импликацию по игровой тематике. Постройте отношение. Приведите пример.

1 балл

IV. Системы нечеткого вывода. Разработайте небольшую систему нечеткого вывода для какой-нибудь игры. Используйте не менее: 3 лингвистических переменных, 3 действий, 5 правил. Примените полученную систему к 3 ситуациям. Оцените адекватность придуманной системы.

5 баллов

Задание для траектории "Тру прогеры":

Разработайте и имплементируйте нечеткую систему вывода в Unreal Engine 4 к некоторой игре по выбору (можно в сероблоочном формате). Создайте билд игры. Оцените адекватность системы. Также практическая работа будет засчитана, если реализуете систему нечеткого вывода в рамках проекта.

8+3 балла

*Score:* /8

Ссылка на отчёт или гит-репозиторий:

## ЛЕКЦИЯ 6. ЭВОЛЮЦИОННЫЕ И РОЕВЫЕ АЛГОРИТМЫ.

### \ Генетический алгоритм. /

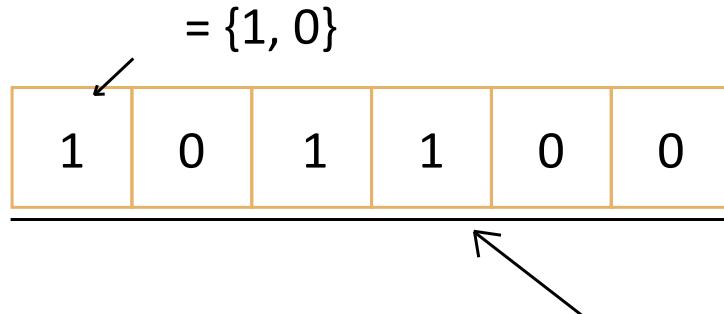
Постановка задачи глобальной оптимизации без ограничений:

В основе генетического алгоритма лежит

Основные принципы:

- 1)
- 2)
- 3)

**Суть популяционных/роевых алгоритмов:** создание популяции (где одна особь = одно решение задачи) — оценка решений — модернизация популяции.



Этапы генетического алгоритма:

- 1)
- 2)
- 3) (оператор )
- 4) (оператор )
- 5) (оператор )
- 6) Проверка критерия останова (возврат к этапу 2)
- 7) Результат —

Оператор селекции, Колесо фортуны:

Суть:

Пример:

Оператор скрещивания, Одноточечный кроссинговер:

Суть:

Пример:

Рекомендуемый диапазон вероятности:

Оператор мутации, Классический:

Суть:

Пример:

Рекомендуемый диапазон вероятности:

Три основных критерия останова:

1)

2)

3)

Алгоритм для работы с вещественными числами в генетическом алгоритме:

## \ Муравьиный алгоритм. /

Постановка задачи комбинаторной оптимизации (и примеры задач):

**Задача Коммивояжёра** часто встречается в различных играх-стратегиях (таких как Anno 1404). Муравьиный алгоритм отлично себя показывает при решении данной задачи.

Суть муравьиного алгоритма:

Этапы муравьиного алгоритма:

1)

$$P = \frac{\tau(r,u)^\alpha \times \eta(r,u)^\beta}{\sum_k \tau(r,u)^\alpha \times \eta(r,u)^\beta}.$$

-  $\tau(r,u)$  – интенсивность феромона на грани между узлами  $r$  и  $u$   
-  $\eta(r,u)$  – функция, которая представляет измерение обратного расстояния для грани  
-  $\alpha$  – вес феромона  
-  $\beta$  – коэффициент эвристики.  
Параметры  $\alpha$  и  $\beta$  определяют относительную значимость двух параметров, а также их влияние на уравнение.

$$\Delta\tau_{ij}^k(t) = \frac{Q}{L^k(t)}.$$

Переменная  $Q$  является константой.

2)

3)

4)

$$\tau_{ij}(t) = \Delta\tau_{ij}(t) + (\tau_{ij}^k(t) \times \rho).$$

Константа  $\rho$  – значение между 0 и 1  
Для испарения феромона

## \ Основы генетического программирования. /

В генетическом программировании (ГП) в качестве особи выступает , представленная в определенном формате, которая решает некоторую задачу.

Часто это выполняется с использованием обучающих данных и индуктивного вывода.

ГП очень близко к машинному обучению и поэтому в качестве достаточно часто выступают

( ).

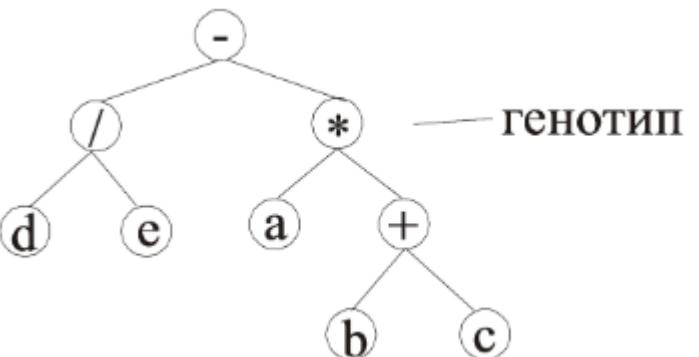
Следует отметить, что ГП работает с генетическим материалом, что требует нестандартной формы представления

и

### Функциональное множество –

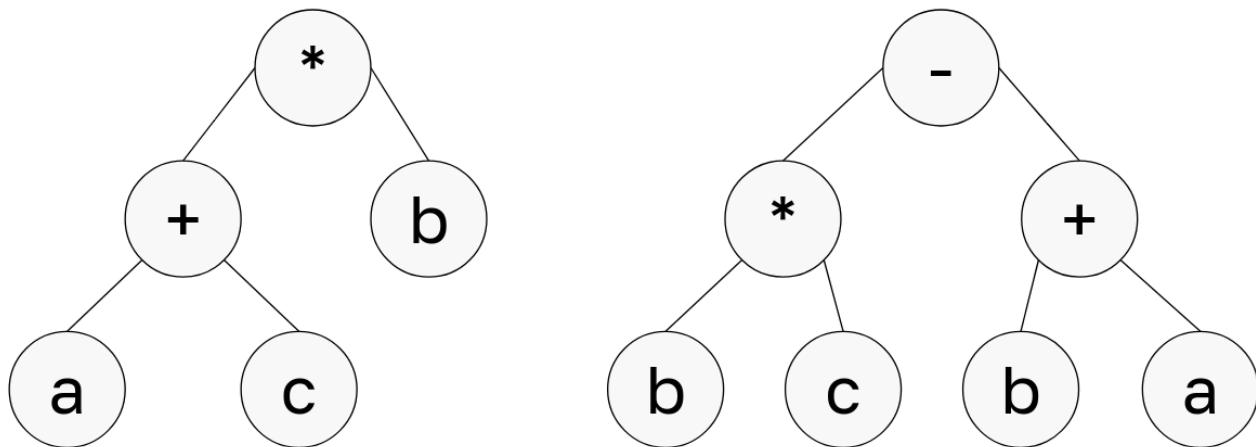
### Терминальное множество –

Одним из важнейших параметров в ГП является максимально возможный размер (сложность) программы. Часто используются следующие два ограничения:

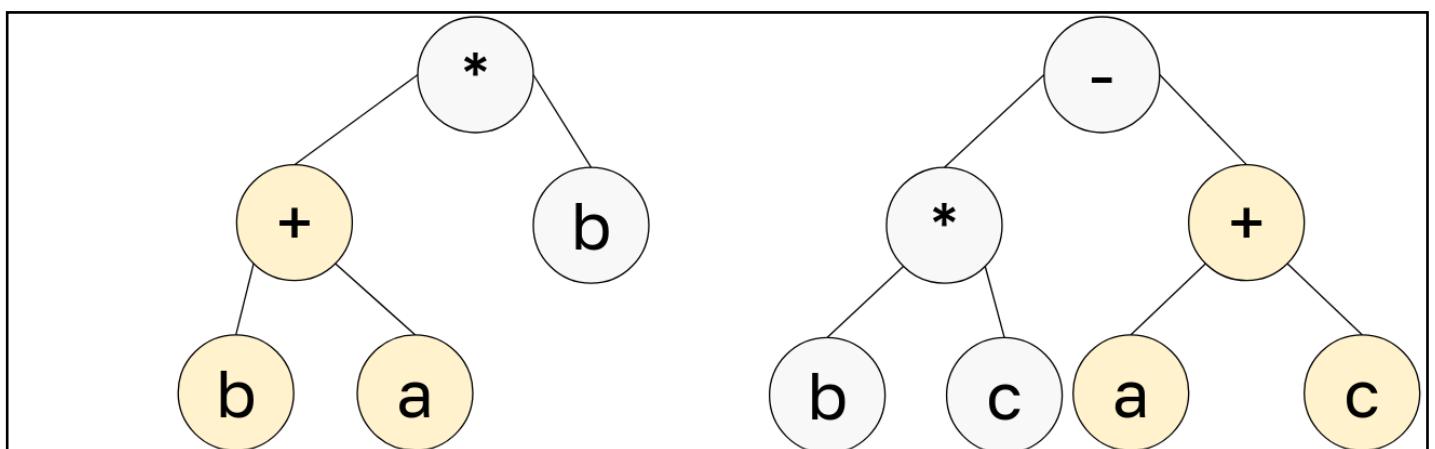


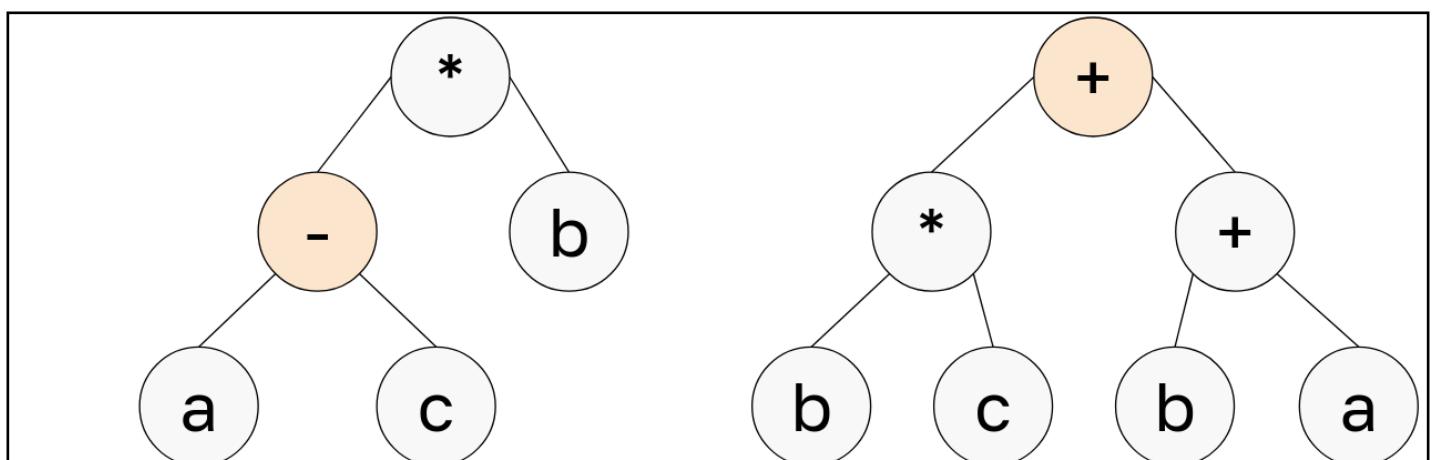
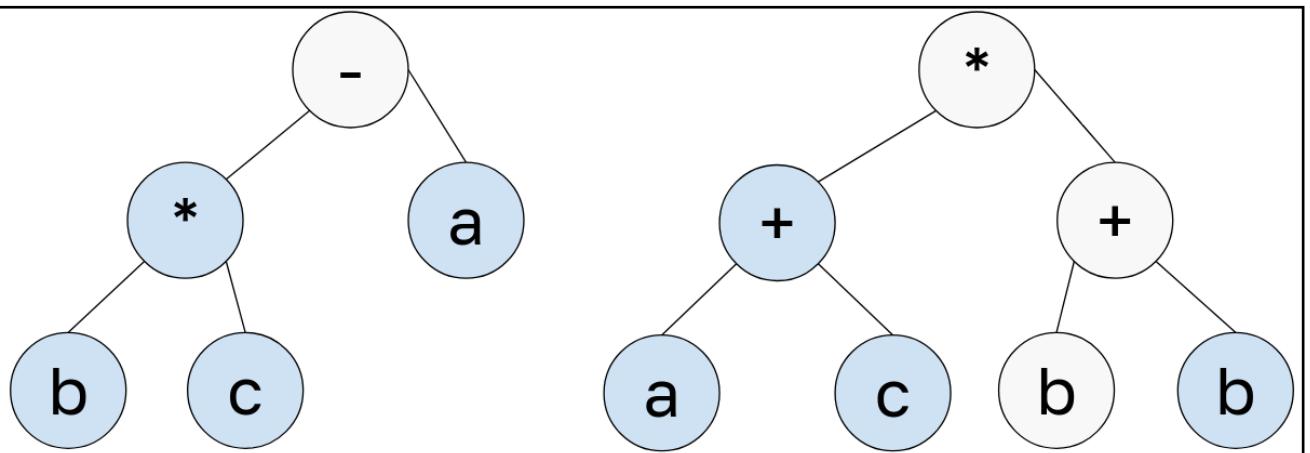
### Операторы скрещивания.

Пусть задано два решения:



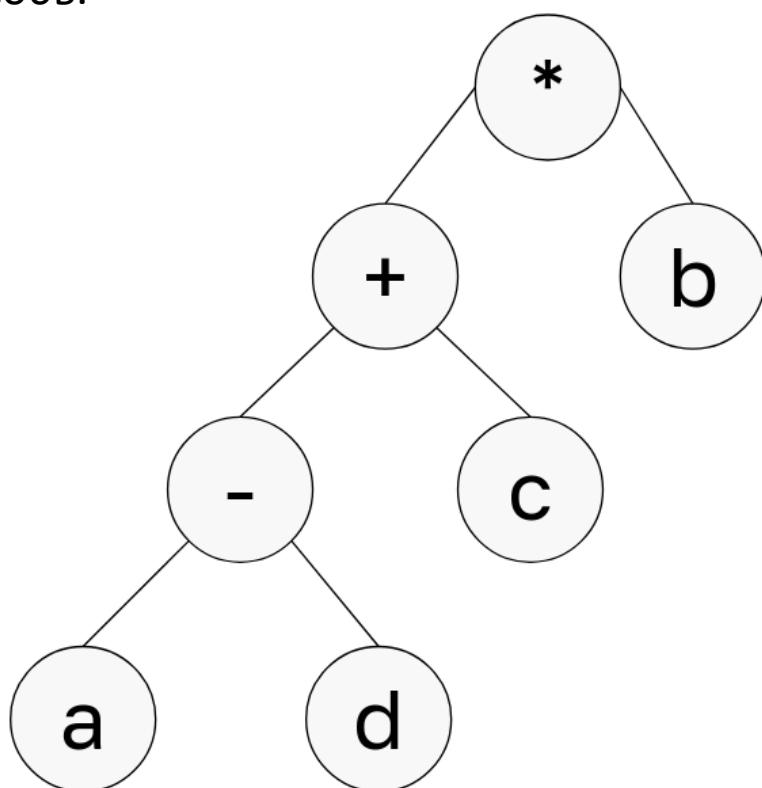
Необходимо сопоставить наименование конкретной реализации оператора и визуального результата.

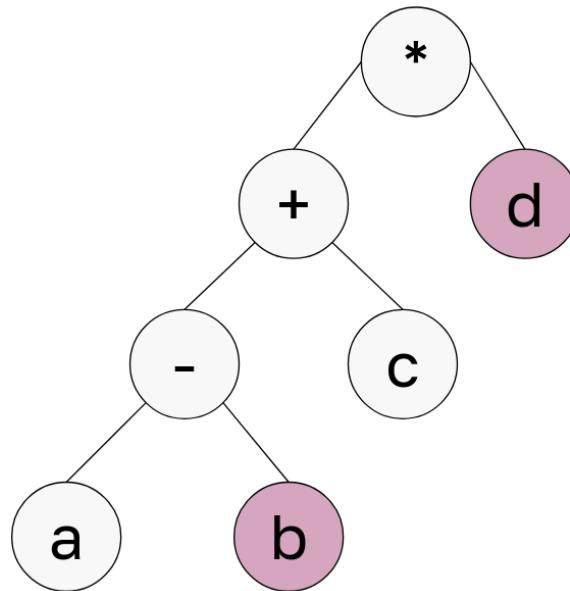
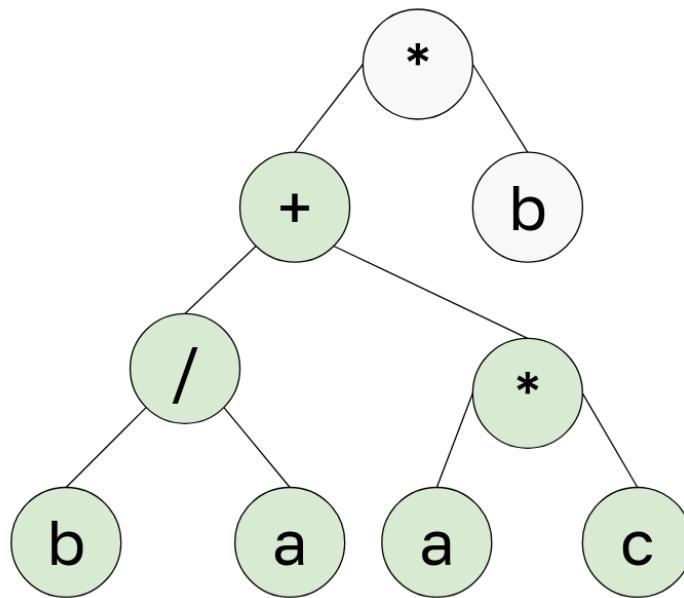
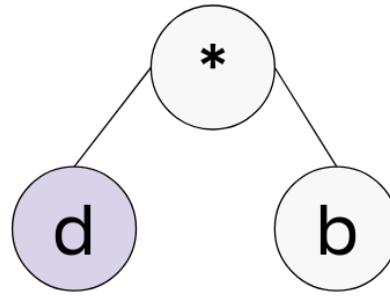




### Операторы мутации.

Пусть задана особь:





Таким образом, для решения задачи с помощью ГП необходимо выполнить описанные выше предварительные этапы:

1. Определить множество;
2. Определить множество;
3. Определить ;

4. Определить значения параметров, такие как мощность популяции, максимальный размер особи, вероятности кроссинговера и мутации, способ отбора родителей, критерий окончания эволюции (например, максимальное число поколений) и т.п.

5. После этого можно разрабатывать непосредственно сам эволюционный алгоритм, реализующий ГП для конкретной задачи.

*Score:* /4

## Литература:

1. Карпенко А. П. «Современные алгоритмы поисковой оптимизации. Алгоритмы, вдохновленные природой.» – 2016.

2. Скобцов Ю. А., Сперанский Д. В. «[Эволюционные вычисления](#)» // Национальный открытый университет «ИНТУИТ», Internet Source. – 2016.

3. Georgios N. Yannakakis and Julian Togelius. «[Artificial Intelligence and Games](#)» // Springer – 2018

4. Меженин М. Г. «[Разработка системы динамического изменения контента видеоигр на основе эволюционного моделирования](#)» // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика. – 2014. – Т. 3. – №. 1. – С. 44-54.

5. Buckland M., Collins M. «[AI techniques for game programming.](#)» // Premier press – 2002.

6. Cord O. et al. «[Genetic fuzzy systems: evolutionary tuning and learning of fuzzy knowledge bases](#)». // World Scientific – 2001. – Т. 19.

7. Schwab B., Schwab B. «[AI game engine programming](#)». // Hingham : Charles River Media – 2009. – С.710.

8. Данилов В. Р., Шалыто А. А. «[Метод генетического программирования для генерации автоматов, представленных деревьями решений](#)» // Сб. докл. XI Междунар. конф. по мягким вычислениям и измерениям. СПб: СПбГЭТУ „ЛЭТИ. – 2008. – С. 248-251.

9. Соколов Д. О. «[Применение двухэтапного генетического программирования для построения модели танка в игре «Robocode»](#) // Научно-технический вестник информационных технологий, механики и оптики. – 2011. – №. 2 (72). – С. 16-22.
10. Скобцов Ю. А., Сперанский Д. В. «Эволюционные вычисления» // Компьютерные науки и информационные технологии. – 2016. – С. 377-381.
11. Kouzehgar M., Badamchizadeh M. A. «[Fuzzy deception game using ant-inspired meta-heuristics](#)» // 2014 IEEE 23rd International Symposium on Industrial Electronics (ISIE). – IEEE, 2014. – С. 134-138.
12. Courtney J. «[Using Ant Colonization Optimization to Control Difficulty in Video Game AI](#)» – 2010.
13. Martin M. «[Using a Genetic Algorithm to Create Adaptive Enemy AI](#)» // Gamasutra–The Art & Business of Making Game. – 2011.

## ПРАКТИЧЕСКАЯ РАБОТА 4. ЭВОЛЮЦИОННЫЕ И РОЕВЫЕ АЛГОРИТМЫ

Задания для траектории "Закрепление материала":

I. Генетический алгоритм. Исследовать задачу (на выбор) и результаты генетического алгоритма на сервисе <https://geneticalgorithms.online/>. Ожидается таблица с не менее, чем 10 экспериментами, их результатами и указанными параметрами.

1 балл

II. Муравьиный алгоритм. Произвести 2 итерации муравьиного алгоритма для задачи Коммивояжера с 6 городами. Матрицу расстояний и инициализацию матрицы феромона придумать или случайно сгенерировать.

2 балла

III. Мини-реферат. Алгоритмы, вдохновленные природой. Выберите любой интересующий вас природный алгоритм, изучите его, опишите: а) биологические предпосылки; б) этапы алгоритма; в) пример одной итерации (рекомендация - Карпенко А. П. Современные алгоритмы поисковой оптимизации. Алгоритмы, вдохновленные природой. – 2016).

4 балла

Задание для траектории "Тру прогеры":

Реализовать генетическое программирование на языке высокого уровня для построения оптимальной вычислительной программы. Пример заданий можно посмотреть тут: <https://intuit.ru/studies/courses/14227/1284/lecture/24178?page=12> ИЛИ применить к созданию деревьев поведений в конкретной ситуации для интеллектуального агента.

7+3 балла

*Score:* /7

Ссылка на отчёт или гит-репозиторий:

# ИТОГОВАЯ ТАБЛИЦА С РЕЗУЛЬТАТАМИ

| Закрепление материала                  | Оценка | Макс |
|--|--------|------|
| ЛК 1. Введение в ТИ и поиск по дереву  |        | /4   |
| ПР 1. Поиск по дереву                  |        | /8   |
| ЛК 2. Ad-Hoc Behavior Authoring        |        | /2   |
| ЛР 1. Навигация в UE4                  |        | /10  |
| ЛР 2. Деревья поведений в UE4          |        | /10  |
| ЛК 3. Навигац. и техн. програм-ния ИИИ |        | /2   |
| Проект. Применение методов ИИИ в UE4   |        | /30  |
| ЛК 4. Принятие решений и элементы МО   |        | /4   |
| ПР 2. Принятие решений                 |        | /7   |
| ЛК 5. Нечеткие модели и системы        |        | /4   |
| ПР 3. Нечеткие модели и системы        |        | /8   |
| ЛК 6. Эволюц. и роевые алгоритмы       |        | /4   |
| ПР 4. Эволюц. и роевые алгоритмы       |        | /7   |

Итог:

/103

## Разработчики курса и рабочей тетради:

Абросимов Кирилл Игоревич  
abrosimov.kirill.1999@mail.ru

Атяпшева Татьяна Владимировна  
tatyana.atyapsheva@mail.ru

Китов Артём Андреевич  
artem-kitov2003@mail.ru

**Институт международного развития и партнёрства  
Университет ИТМО**  
Ломоносова, 9, Санкт-Петербург, Россия, 191002

