UDC 69

Kutekhov D.O., Areshin S.O., Sudakov V.A. Investigation of the effectiveness of machine learning methods with reinforcement in the traveling salesman problem

Исследование эффективности методов машинного обучения с подкреплением в задаче коммивояжера

Kutekhov Dmitry Olegovich
Graduate student.
federal state
budget educational institution
higher education "Moscow Aviation Institute (National Research University)"
Areshin Stanislav Olegovich
Graduate student.
federal state
budget educational institution
higher education "Moscow Aviation Institute (National Research University)"
Scientific director
Sudakov Vladimir Anatolievich Leading Researcher.
Doctor of Technical Sciences, IPM them. M.V. Keldysh RAS
Кутехов Дмитрий Олегович
Студент магистратуры.
Федеральное государственное
бюджетное образовательное учреждение
высшего образования «Московский авиационный институт (национальный
исследовательский университет)»
Арешин Станислав Олегович
Студент магистратуры.
Федеральное государственное
бюджетное образовательное учреждение
высшего образования «Московский авиационный институт (национальный
исследовательский университет)»
Научный руководитель
Судаков Владимир Анатольевич. Ведущий научный сотрудник.
Доктор технических наук, ИПМ им. М.В. Келдыша РАН

Abstract. *The paper presents a solution to the traveling salesman problem using various reinforcement learning algorithms and a comparative analysis of the results obtained when learning algorithms by minimizing the length of the final route. The result of the selected algorithm was compared with a random one.*

Keywords: *reinforcement learning, traveling salesman problem, combinatorial optimization problem, deep learning, heuristic algorithms, route optimization*

Аннотация*. В работе представлено решение задачи коммивояжера с применением различных алгоритмов обучения с подкреплением и проведен сравнительный анализ результатов, полученных при обучении алгоритмов путем минимизации длины конечного маршрута. Результат работы выбранного алгоритма сравнивался со случайным.*

Ключевые слова: *обучение с подкреплением, задача коммивояжера, комбинаторная задача оптимизации, глубокое обучение, эвристические алгоритмы, оптимизация маршрута*

Введение.

Задача коммивояжера - древнейшая задача комбинаторной оптимизации, заключающаяся в поиске Гамильтонова графа — графа, содержащего гамильтонов цикл. При этом гамильтоновым циклом является такой цикл (замкнутый путь), который проходит через каждую вершину данного графа ровно

по одному разу, то есть простой цикл, в который входят все вершины графа. Исследование решения задачи коммивояжера имеет богатую историю и обширный инструментарий различных подходов.

Другими словами, Задача коммивояжера (или TSP) заключается в нахождении кратчайшего возможного пути, соединяющего города с учетом матрицы расстояний между этими городами.

Задача имеет множество практических применений в различных областях, в том числе:

1. Логистика и транспорт: решение задачи коммивояжера можно использовать для оптимизации маршрутов доставки для курьеров, водителей и грузовых компаний, сокращая расход топлива и время, необходимое для доставки.
2. Производство и производство: оптимизация порядка производства в производственном процессе, сокращая время и затраты, необходимые для производства.
3. Дизайн сети: оптимизация развертывания сетей связи, например, для размещения вышек сотовой связи.
4. Секвенирование ДНК: определение порядка секвенирования фрагментов ДНК, что сокращает общее время и стоимость секвенирования.
5. Робототехника и автоматизация: оптимизация пути робота, сокращения времени и энергии, необходимых для выполнения задачи.
6. Туристическая индустрия: оптимизация маршрутов путешествий для туристов, снижения затрат и времени, необходимых для поездки.

Это всего лишь несколько возможных примеров применений задачи коммивояжера из множества возможных задач оптимизаций, стоящих перед современным бизнесом и наукой . Известно, что поиск оптимального решения является NP-трудной задачей и существует (N-1)! различных возможных способов собрать решение с посещением N городов.

В настоящее время существует множество различных методов - как точных, так и эвристических - для эффективного поиска хороших решений. Подходы варьируются от эвристических моделей муравьиных колоний до чрезвычайно эффективных решателей (таких как Concorde), использующих сложные методы целочисленного линейного программирования для поиска точных решений.

Такие решатели, как Concorde, могут находить оптимальные решения для задач с несколькими десятками тысяч городов, и близкие к оптимальным (в пределах 1% или меньше) решения для задач с миллионами городов. Один из самых больших экземпляров задачи был решен с помощью Concorde [1] в 2006 году и включал 85900 взаимосвязей, что в то время заняло 136 CPU-лет [2]. Этот экземпляр задачи был евклидовым - это означает, что все расстояния между городами равны евклидовым расстояниям между ними. Большие неевклидовы

задачи (например, такие как рассмотрение времени ходьбы между городами вместо евклидовых расстояний) также были решены, хотя в целом это гораздо сложнее. Также используются эвристические алгоритмы, которые позволяют получить приближенное решение задачи. Алгоритмы не гарантируют оптимального решения, но дают решения, близкие к оптимальным за разумное время. Сравним эвристические методы с обучением с подкреплением [3].
Эвристики:

- Не гарантируют нахождения точного решения (как и rl).
- Требуют привлечения квалифицированного специалиста для создания эвристики
- Решают каждую задачу с нуля

Обучение с подкреплением:

- Как и эвристические алгоритмы, не гарантирует нахождение точного решения
- Агент обучается алгоритму решения задачи
- Агент обучается решать класс задач

Таким образом видно, что обучение с подкреплением позволяет преодолеть ограничения, которые присущи эвристическим алгоритмам.

Теоретическая часть.

Дан ориентированный граф *G=(V, A)* где,

- V - множество вершин, которые нужно посетить;
- c_{ij}, *(i,j)* ∈ *A* - множество ребер графа, где c_{ij} - дистанция между вершинами
- x_{ij}, *(i,j)* ∈ *{0, 1}* - принадлежность ребра к маршруту

тогда задача заключается в поиске маршрута с наименьшими затратами,

$$min \sum_{i \in A} \sum_{,j \in A} c_{i,j} x_{i,j}$$

который удовлетворяет четырем основным ограничениям задачи:

$$\sum_{i \in V \setminus \{j\}} x_{i,j} = 1, j \in V \quad (1)$$

$$\sum_{i \in V \setminus \{i\}} x_{i,j} = 1, i \in V \quad (2)$$

$$\sum_{i \in S} \sum_{j \notin S} x_{i,j} \geq 1, S \neg \subseteq V, |S| \geq 2 \quad (3)$$

$$x_{i,j} \in \{0,1\}, (i,j) \in A \quad (4)$$

Ограничения (1) и (2) говорят нам, что каждая вершина j/i должна

соединяться/быть соединенной ровно с одной другой вершиной i/j. Однако этих двух ограничений недостаточно, чтобы гарантировать, что результатом работы модели является граф, состоящий из одной компоненты .

Поэтому ограничение (3), говорит, что конечное решение не может быть набором меньших маршрутов (или подциклов) - модель должна вывести единственный маршрут, соединяющий все вершины. Ограничение (4) определяет переменную x_{ij}, устанавливая ее равной 1, если две вершины (i, j) в графе являются частью конечного маршрута, и 0, если нет.

Обучение с подкреплением (Reinforcement Learning) - это область машинного обучения, моделирующая, процесс взаимодействия интеллектуального агента с окружающей его средой в целях максимизации некоторого совокупного вознаграждения. Обучение с подкреплением является одной из трех основных парадигм машинного обучения, наряду с обучением с учителем и обучением без учителя.

Обучение с подкреплением отличается от контролируемого обучения тем, что не требует представления маркированных пар вход/выход и не требует явного исправления неоптимальных действий. Вместо этого основное внимание уделяется поиску баланса между исследованием неизвестного пространства действий и эксплуатацией текущих знаний.

Среда обычно задается в форме марковского процесса принятия решений:

- Множество S состояний среды;
- Множество A действий агента;
- P(s, s') = Pr($s_{t+1} = s' \mid s_t = s, a_t = a$) – вероятность перехода из состояния s в момент времени t при совершении действия a приведет к состоянию s' в момент времени t+1;
- Выбор действия агента моделируется с помощью политики
  - $\pi: A \times S \rightarrow [0, 1]$
  - $\pi(a, s) = Pr(st=s \mid at=a)$
- $R_a(s, s')$ – вознаграждение агента после перехода из состояния s в состояние s';
- Задача агента - максимизация вознаграждения

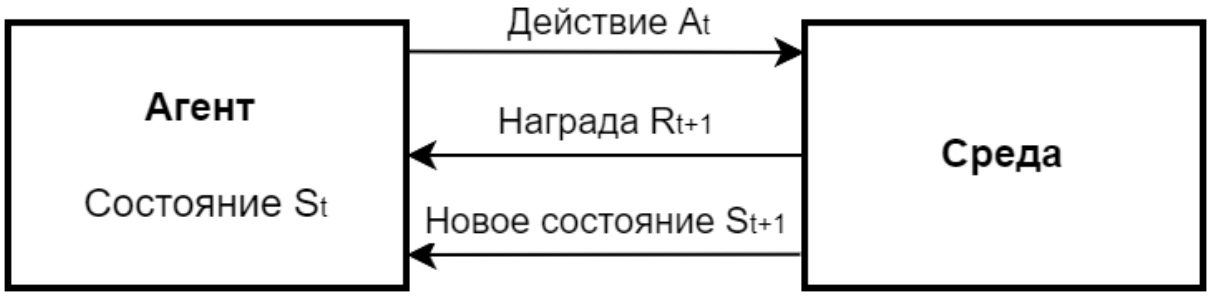

Рисунок 1. Схематичное представление процесса обучения с подкреплением

Многие алгоритмы обучения с подкреплением в этом контексте используют методы динамического программирования. Основное различие между классическими методами динамического программирования и алгоритмами обучения с подкреплением заключается в том, что последние не предполагают знания точной математической модели марковского процесса принятия решений и нацелены на большие MDP, где точные методы становятся невыполнимыми.

Кроме того, у обучения с подкреплением есть еще несколько преимуществ:

1. В эту структуру можно легко вставить достаточно произвольные функции вознаграждения, поэтому можно использовать один и тот же алгоритм для различных вариантов TSP, и особенно для вариантов, которые значительно отличаются от евклидовых. Например, минимизация продолжительности маршрута с учетом некоторого времени, проведенного в каждом городе, или со случайными задержками, прикрепленными к ребрам графа и зависящими от определенных динамических характеристик модели.
2. Скорость вывода и использование в режиме реального времени: как только мы обучили нашу нейронную сеть выдавать следующий лучший город для посещения, это означает, что решение о том, куда ехать дальше, требует только прямого обхода нейронной сети, что обычно происходит быстро. Таким образом, она лучше подходит для динамичных и онлайновых случаев использования, чем некоторые более медленные подходы.
3. Возможность модификации модели для решения других задач комбинаторной оптимизации.

В нашем случае агент заинтересован в поиске кратчайшего пути. Его состояние содержит весь граф, соединяющий все города, координаты городов, а также историю посещений городов. Окружение полностью наблюдаемо и тривиально; оно просто позволяет совершить детерминированное перемещение по ребру графа и наблюдать за пройденным расстоянием.

Чтобы применить обучение с подкреплением к задаче коммивояжера, необходимо создать среду, моделирующую проблему. Среда должна включать в себя представление городов и их расстояний друг от друга, а также функцию вознаграждения, обеспечивающую обучение агента. Цель агента - изучить политику, которая сопоставляет данный штат (набор городов) с действием (следующий город для посещения), максимизирующим ожидаемое вознаграждение. Функция вознаграждения может быть основана на продолжительности пути, при этом более высокое вознаграждения будет предоставляться за более короткие решения.

Множество действий A является набором возможных действий, которые агент может предпринять на каждом этапе решения задачи. Действие соответствует выбору следующего города для посещения. Таким образом, пространство действия - это набор всех возможных городов, которые агент может посетить на следующем шаге.

Множество S состояний среды в задаче TSP относится к информации, которую агент может наблюдать на каждом этапе решения задачи. Область наблюдения обычно включает координаты на плоскости текущего города, который посещает агент, а также набор городов, которые не посещались и евклидово расстояния от текущего города до остальных. Информация используется агентом для принятия решения о том, какой город посетить в следующий раз и как составить оптимальный маршрут.

Политикой π агента в задаче коммивояжера будет являтся сопоставление состояний и действий, а именно текущего и следующего города для посещения.

Политика в задаче коммивояжера может принимать различные формы, в зависимости от конкретного используемого алгоритма обучения с подкреплением. Например, алгоритм Q-обучения может использовать таблицу для представления ожидаемых вознаграждений для каждой пары состояние-действие в TSP. Затем политика выводится из таблицы путем выбора действия с наибольшим ожидаемым вознаграждением для текущего состояния. Другие алгоритмы обучения с подкреплением, такие как методы градиента политики, используют параметрическое представление. Таким представлением могут являться веса нейронной сети.

В обучение с подкреплением награда агента за совершения действия является мерой, отражающей, на сколько совершенное действие приближает агента к правильному решению. Целью агента в задаче коммивояжера является поиск минимального маршрута между всеми городами, соответственно награда агента должна отражать длину решения. Для этого на каждом шаге агента награда $R_a(s, s')$ должна быть равна отрицательному евклидову расстоянию между двумя точками плоскости:

$$R_a(s, s') = -\sqrt{\sum_{k=1}^{m} (x_{ik} - x_{jk})^2}$$, где k=2

Награда на каждом шаге отрицательна для того, чтобы агент, максимизируя награду, выбирал самый короткий маршрут.

Алгоритмы обучения с подкреплением можно разделить на две группы: model-free и model-based. Если агент обучается, делая прогнозы о последствиях своих действий, то это model-based алгоритм, иначе, если он учится только на собственном опыте, тогда это model-free алгоритм.

В свою очередь model-free алгоритмы делятся на две группы: on-policy (Policy Optimization) и off-policy (Q-Learning). Отличие этих групп в том, что on-policy алгоритмы улучшают текущую политику, off-policy алгоритмы моделирую политику, отличную от той, что использовалась для выполнения действий на предыдущем шаге [3]. Этот подход предполагает обучение на поведении "старой" политики, одновременно взаимодействуя с окружающей средой с использованием более новой, улучшенной политики.

Для реализации поставленной задачи рассмотрим следующие алгоритмы обучения с подкреплением: Impala, A2C, A3C, SAC, PPO.

A2C, A3C - это алгоритмы обучения с подкреплением для решения задач последовательного принятия решений, которые используют глубокие нейронные сети для обучения агента. A2C (Advantage Actor-Critic) - это синхронная версия асинхронного алгоритма A3C (Asynchronous Advantage Actor-Critic). A2C более эффективен в вычислительном отношении, чем A3C, за счет разделения параметров нейронной сети между несколькими параллельными экземплярами среды. A2C - это model-free алгоритм, что означает, что он не требует модели среды для обучения. Он использует функцию преимущества для оценки ожидаемой награды от принятия действия в данном состоянии по сравнению со средней ожидаемой наградой от принятия всех возможных действий в этом состоянии.

A3C - это асинхронная версия алгоритма Advantage Actor-Critic (A2C), которая использует несколько параллельных сред для ускорения обучения. Каждая среда работает независимо и взаимодействует со своей копией нейронной сети. Обновления от каждой среды затем объединяются для обновления общих параметров нейронной сети [4].

Критик – это компонент алгоритма обучения, который оценивает функцию ценности для всех агентов в многоагентной среде [5]. Существует централизованная и децентрализованная реализация. В отличие от децентрализованного критика, который оценивает функцию ценности для каждого агента независимо, централизованный критик может учитывать взаимодействия и зависимости между агентами. Критик обычно реализуются в виде нейронной сети, которая принимает состояния и действия всех агентов в качестве входных данных и выводит скалярное значение.

Функция ценности оценивает ожидаемую сумму будущих вознаграждений для данного состояния или пары состояние-действие и используется для управления процессом обучения путем оценки качества действий агента. Централизованный критик оценивает функцию ценности для всех агентов в среде, что может быть особенно полезно в средах, где действия агентов взаимозависимы.Например, в игре в футбол с двумя командами действия каждого игрока зависят от действий других игроков его команды, а также от действий соперников. В этом случае децентрализованный критик, оценивающий функцию ценности для каждого игрока независимо, может не уловить взаимодействие между игроками, тогда как централизованный критик может оценить совместную функцию ценности для обеих команд и лучше отразить динамику игры.

В целом, A2C и A3C - это похожие алгоритмы, которые отличаются друг от друга своей реализацией.

IMPALA (Importance Weighted Actor-Learner Architecture) - еще один асинхронный алгоритм, который использует несколько параллельных сред для ускорения обучения. Он использует centralized critic для оценки функции ценности для всех агентов, что позволяет более эффективно обновлять веса нейронной сети. IMPALA также использует выборку по важности для коррекции смещения,

возникающего при параллельном использовании данных, собранных от нескольких агентов. Это позволяет алгоритму учиться на более разнообразном опыте, что может улучшить качество выработанной сетью политики.IMPALA - это более продвинутый алгоритм, который использует критика и выборку по значимости для повышения эффективности и результативности процесса обучения.

PPO, или Proximal Policy Optimization - алгоритм, позволяющий получать наибольший возможный прирост качества, не ухудшая при этом текущих показателей эффективности политики. Иными словами, алгоритм позволяет обучать агента таким образом, чтобы текущая политика не сильно отличалась от предыдущей.

Алгоритм Soft Actor-Critic (SAC) - это тип алгоритма обучения с подкреплением, который относится к категории алгоритмов off-model и off-policy. Одна из ключевых особенностей алгоритма SAC заключается в том, что он использует энтропийную регуляризацию [6] для поощрения исследования среды. Это помогает предотвратить застревание агента в локальном оптимуме и побуждает его исследовать более широкий диапазон действий. SAC также использует подход мягкого Q-обучения (soft Q-learning) [7], что означает, что Q-значения вычисляются с помощью мягкой версии функции максимума, а не традиционной функции максимума. Это помогает сгладить оценки Q-значений и может улучшить стабильность обучения. В целом, SAC - это мощный и гибкий алгоритм обучения с подкреплением, который показал хорошие результаты при решении различных задач, включая непрерывное управление и робототехнику.

Экспериментальная часть.

Для обучения агента смоделируем среду, отражающую проблематику задачи коммивояжёра. Среда представляет собой полносвязный граф с наградой, основанной на евклидовом расстоянии между парами городов.

Среда позволяет перемещаться между всеми узлами графа. Цель состоит в том, чтобы минимизировать затраты на прохождение всех узлов графа ровно один раз. В обучении с подкреплением эпизодом называется проход агента от начального состояние до терминального. В случае задачи коммивояжера терминальным состоянием является посещение всех вершин в графе. Если агент перемещается в узел более одного раза, то получает большой штраф и завершает эпизод. Если все вершины графа посещены, агент получает награду, равную 100 и завершает эпизод. Граф генерируется случайным образом с N городами.

Обучим модель обучения с подкреплением с помощью вышеперечисленных алгоритмов. Основной метрикой является средняя награда, полученная агентом на каждом эпизоде обучен.

Алгоритм PPO показывает хорошие результаты, на графике видно, что награда стремиться к 100.

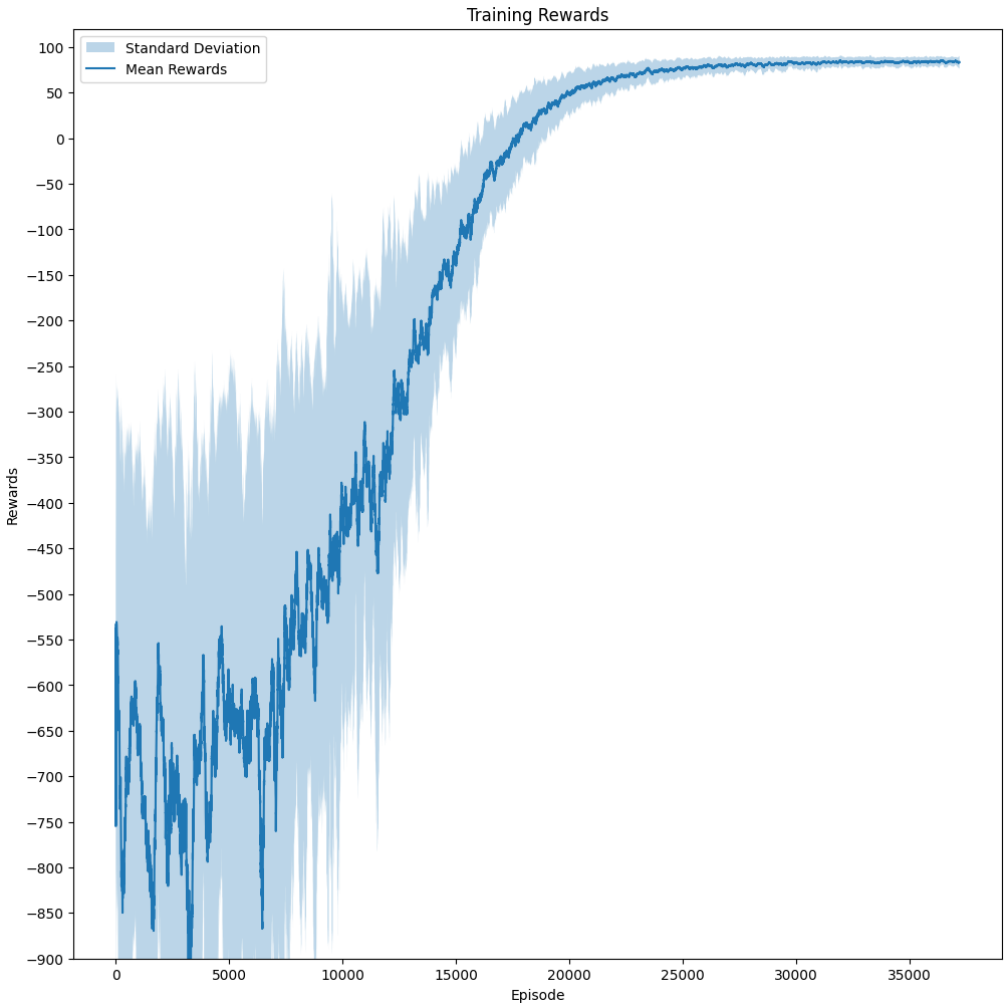

Рисунок 2. Средняя награда алгоритма PPO

Средняя награда алгоритма SAC не достигает 100, а находится примерно на уровне 50, что означает, что алгоритм часто посещает вершины дважды.

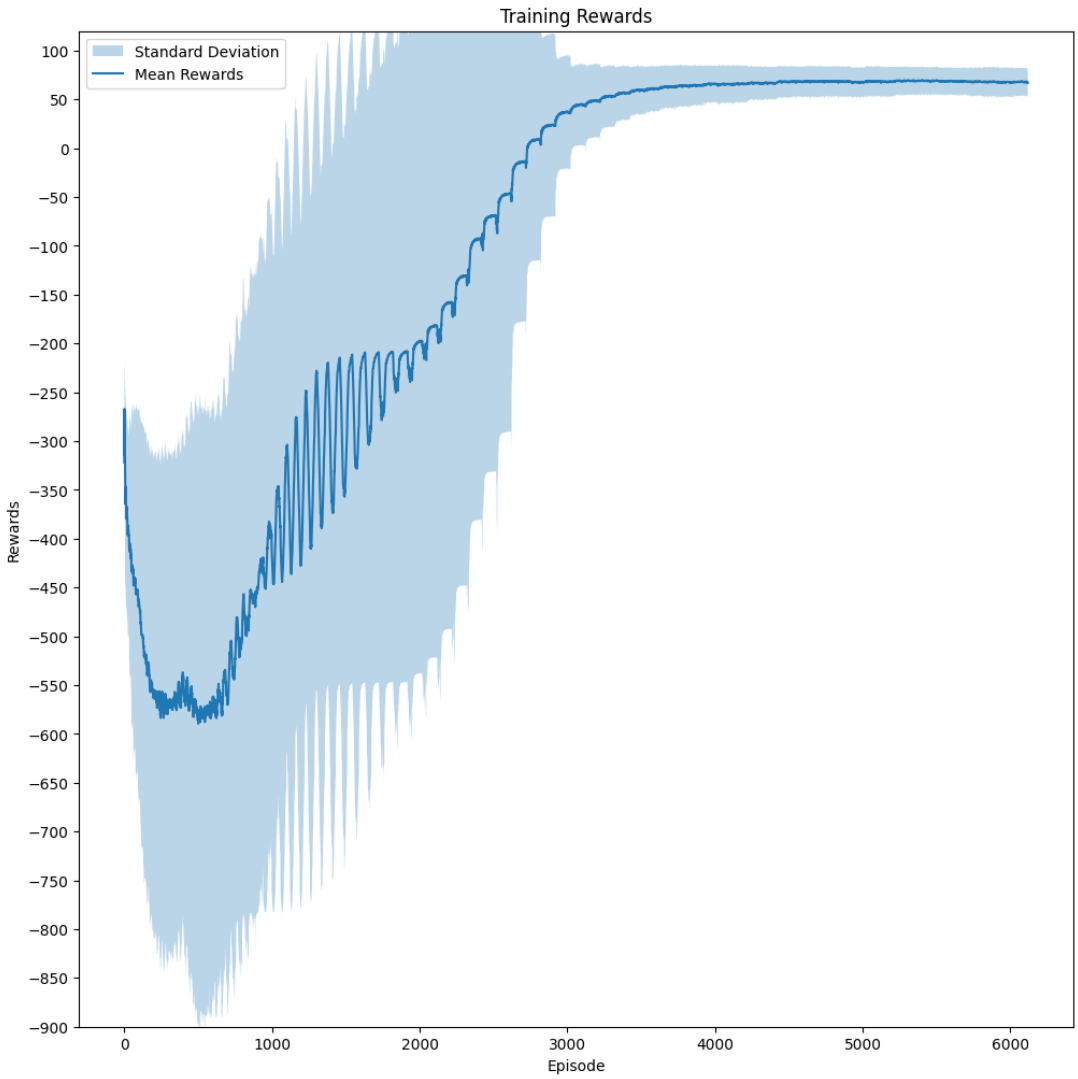

Рисунок 3. Средняя награда алгоритма SAC

При использование алгоритмов A2C, A3C и Impala средняя награда всегда оставалось отрицательной. Это связано с тем, что алгоритм редко заканчивал игры, при этом часто выбирал следующим действием уже посещенные города.

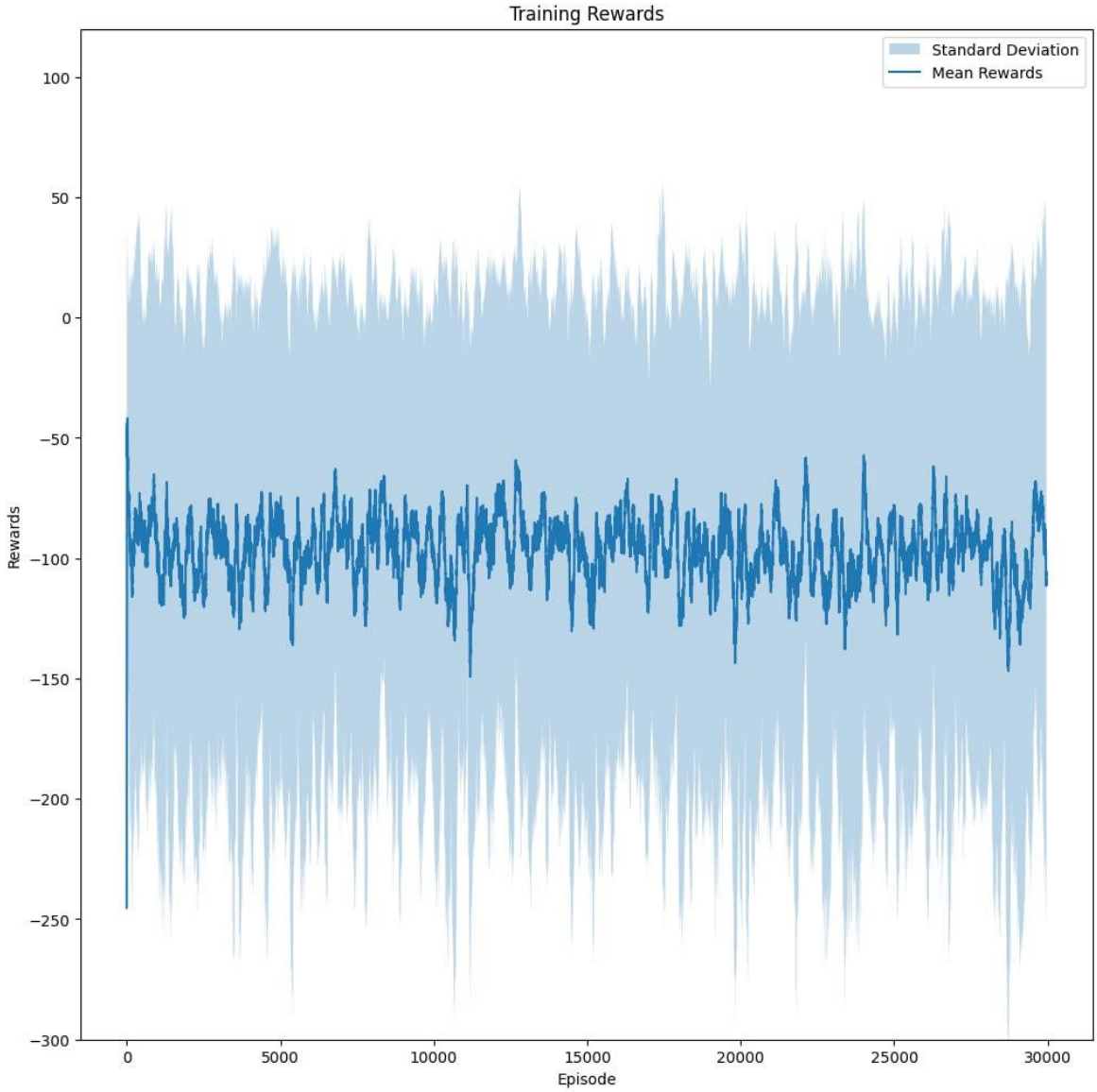

Рис. 4 средняя награда алгоритма A2C

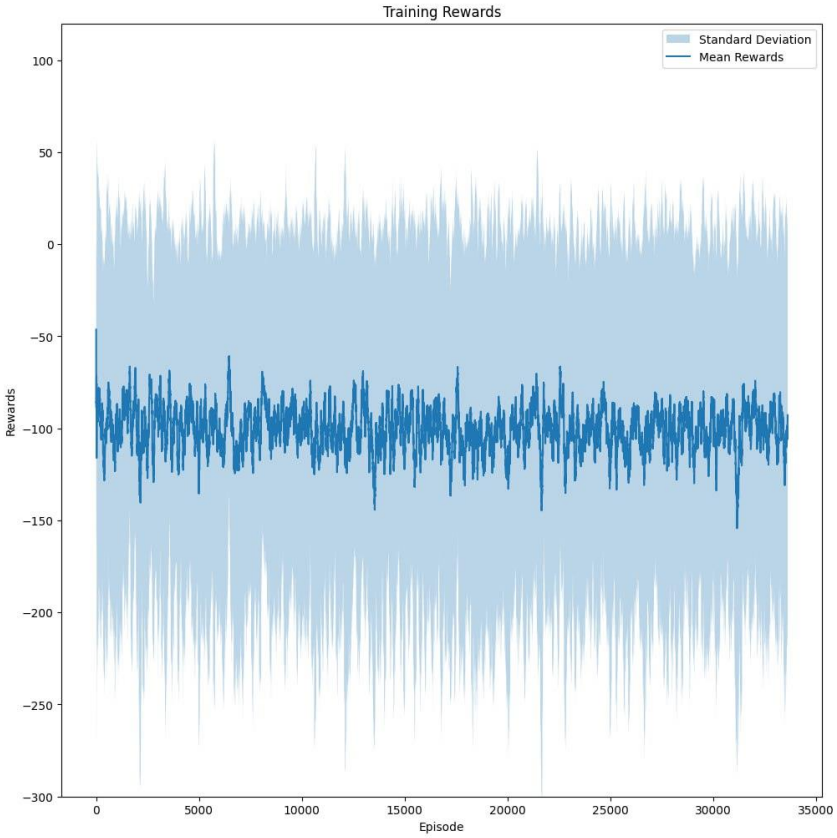

Рисунок 5. Средняя награда алгоритма A3C

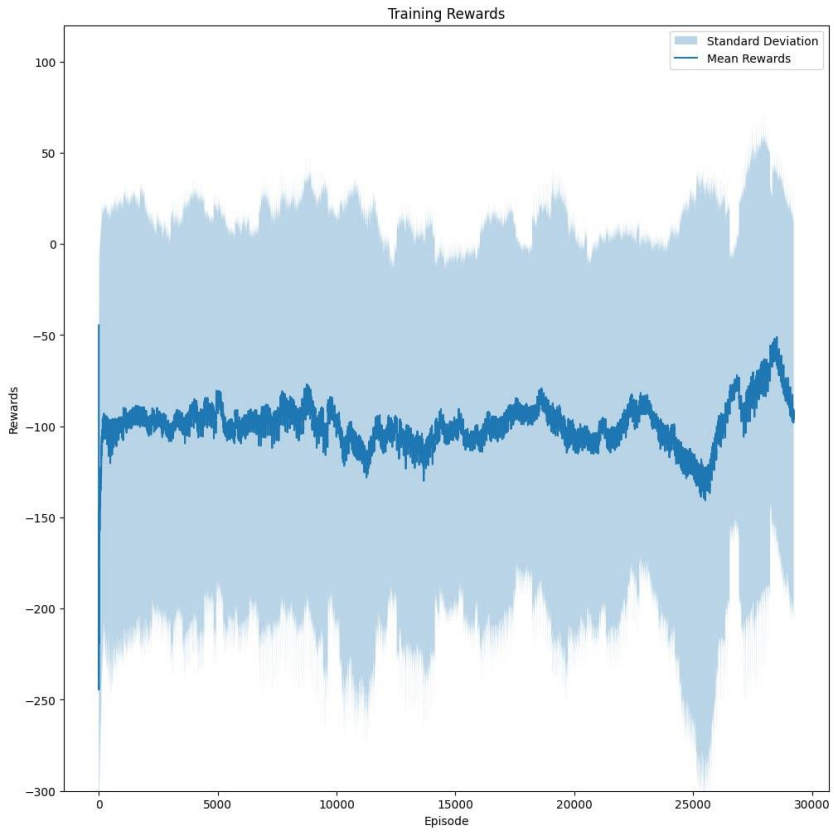

Рисунок 6. Средняя награда алгоритма Impala

Таким образом единственным алгоритмом с приемлемой средней наградой оказался алгоритм PPO, сравним результат работы алгоритма со случайным. Для этого возьмем длину получившегося решения для N городов, где N ∈ [4, 23].

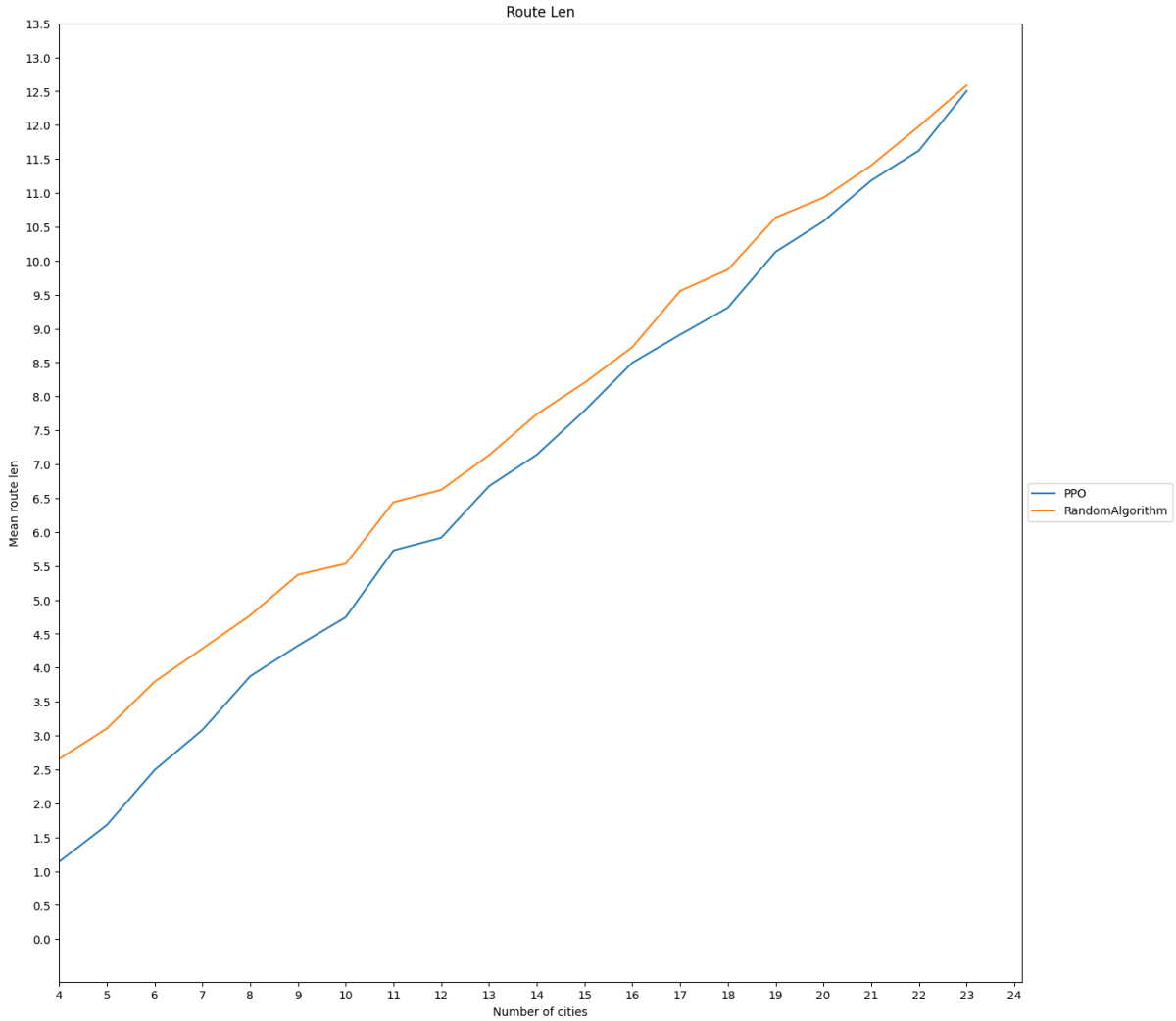

Рисунок 7. Сравнение длин решений алгоритма обучения с подкреплением и случайного алгоритма

Из приведенного выше графика можно сделать вывод, что алгоритм способен находить решение более эффективное, чем случайный. При дальнейшем увеличении количества городов точность алгоритма обучения с подкреплением достигает случайной.

Заключение.

Таким образом, в работе показана применимость обучения с подкреплением в задаче коммивояжера и были протестированы такие алгоритмы, как PPO, SAC, A2C, A3C, Impala. В результате сравнения и анализа средних наград алгоритмов для конечной реализации решения был выбран алгоритм PPO,

точность которого в проведенных экспериментах была выше случайной при $N \in [4, 23]$ количестве городов в исходном графе.

References

1. Электронный ресур https://www.math.uwaterloo.ca/tsp/pla85900/index.html Дата обращения 05.04.2023
2. Электронный ресур https://www.math.uwaterloo.ca/tsp/pla85900/compute/cpu.htm Дата обращения 23.04.2023
3. Пантелеев А.В., Метлицкая Д.В., Алешина Е.А. Методы глобальной оптимизации. Метаэвристические стратегии и алгоритмы. М.: Вузовская книга, 2013.
4. Reinforcement Learning: An Introduction. Second edition, in progress. Richard S. Sutton and Andrew G. Barto c 2014, 2015. A Bradford Book. The MIT Press.
5. Vijay R., Konda John N., Actor-Critic Algorithms – 2018. URL: https://arxiv.org/pdf/1801.01290.pdf
6. Электронный ресур https://spinningup.openai.com/en/latest/algorithms/sac.html Дата обращения 02.05.2023
7. Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, Sergey Levine – 2017. URL: https://arxiv.org/pdf/1702.08165.pdf