Текст работы КТ-2

Смолкина и Горбов

Актуальность и мотивация

Актуальность использования эвристик в навигации неоспорима. Это может быть задача навигации такси, робота-манипулятора или вертолета.

Давайте представим, мы представим, что пилотируем вертолет. И видимость всего лишь 1 км радиусом. Сложность, с которой мы сталкиваемся, это обновление в реальном времени, что также влечёт за собой высокую сложность бортовых вычислений. Также стоит учитывать, что наш вертолет будет летать совершенно над разными ландшафтами. И мы бы хотели получить, такую систему. которая бы работала в любом из ландшафтов. И зачастую это и является ключевой проблемой, так как сложно найти что-то универсальное.

Что будет происходить когда мы запускаем state of the art алгоритм, он не будет контактировать со средой. Мы будем постепенно раскрывать очень много узлов. Изначально, мы не будем фокусироваться на определении полетных или неполётных зон.

Известные планировщики, которые не взаимодействуют со внешней средой, не способны работать в реал-тайме

Обычно,все алгоритмы заточены на рассмотрение Худшего из случаев.

В нашем случае, мы хотим использовать data-driven подходы, мы хотим чтобы планер адаптировался сам по себе при взаимодействии со средой.

Поэтому мы сфокусировались на обучении эвристик на гарфах, чтобы улучшить перфоманс.

Планирование должно быть сосредоточено на ожидаемой перфомансе в настоящей среде с использованием машинного обучения.

Постановка задачи

Проблема - Для обеспечения работы в режиме реального времени алгоритмы должны уделить первоочередное внимание минимизации усилий по поиску, т. е. сохранению объема волнового фронта поиска как можно меньше, пока он растет к цели.

Вообще графы используются в большом числе доменов - например в AlphaGO

Нам нужна эвристика которая приведет к минимуму коллизий

1. Мы хотим вычислить эвристическую policy, которая явно минимизирует expected edge evaluation.
2. На данный момент достаточно найти достижимый путь в реальном времени.
3. Может быть расширен для включения стоимости пути в любое время:

Быстро находите возможный путь и уточняйте его с течением времени

Мы концентрируемся на том - как быстро найти достижимый (feasible) результат

У нас есть старт,цель ,функция саксессор, функция эвалюэшн и селектор

В Open list - лежат все потенциальные кандидаты

В Closed list - те что раскрылись

В Invalid - те что не приведут нас к цели

Пока цель не найдена мы выбираем что-то эвристической функцией, раскрываем чтобы получить всех возможных сакссесеров и все невалидные ребра ( сравнивая с конечной целью).

Обновляем наши листы и проделываем эту процедуру пока цель не попадет в open.

Алгоритм

Key Insight: Search as sequential decision making under uncertainty(over World map) -Основная идея: поиск как последовательное решение

изготовление в условиях неопределенности (по карте мира)

То есть все это время мы жонглируем этими тремя листами.

И мы воспринимаем эристики как classifier. Она выбирает что раскрыть, стараясь минимизировать кол-во раскрытий. Тк наказание = -1, то можно представить, что оптимальная политика явно минимизирует усилия по планированию.

Сразу возникает вопрос, как же мы определяем search state.

Мы сожмем наши три листа, чтобы получить feature vector для каждого узла из open list.

И получим два главных вектора:

1. search based - где мы находимся в нашем графе ( поисковом дереве)
2. world based - Зависит от среды, уже раскрытой

Важно! Расчет признаков не должен требовать дополнительных усилий по поиску!

Задачу уже можно решить в 3ёх измерениях. То есть мы можем построить Оракла, который знает настоящее необходимое кол-о раскрытий.

Oracle является «ясновидящим» с доступом к истинному состоянию лежащего в основе мира, и имеет больше информации чем мы.

Мы будем использовать имитационное обучение cost-to go

Используется функция аппроксимации с oracle регрессором Q value estimator.

Где t~U это наши world’s sample - Равномерно дискретизированный временной шаг

где s-d distribution deals with policy? а именно distribution of state under roll-in policy

Задача в уменьшении RMSE вместе с ораклом.

Мы пытаемся найти параметрический вектор, который минимизирует ошибку.

Планировщик использует жадный алгоритм с усилием по поиску.

Несмотря на все достоинства Оракла, если мы будем использовать только его, то при моменте когда оракл не знает, что делать или было ошибка при обучении все порушится. ПОэтому решением будет комбинация из методов. Это было найдено ( там много успехов и хорошо написано в в Rossel and Bagnel 2014)

Search As Imitation Learning - SaIL

Для каждой итерации проходит m эпизодов

На каждом эпизоде мы sample problem from distribution

Мы Roll-in policy и выбираем действие

И запускаем Оракле для реального усилия по поиску для текущего действия или узла

Агрегируем данные с уже полученными ранее 

Для одного эпизода мы проделываем это K раз 

И обучаем следующие policy на этих агрегированных данных

И выбираем лучшую полиси, которую нашли во время валидации.

Описание Экспериментов

Общее описание экспериментов: Мы будем оценивать алгоритм SAIL в различных задачах 2D-навигации, где робот должен планировать движение от нижнего левого угла к верхнему правому по 8-связной сетке. Сетка встроена в бинарную карту препятствий.

Модель принимает в качестве входных данных 17-мерный признак.

вектор f ∈ F для пары (v, s), который содержит такие значения, как ближайшее недопустимое состояние в I, расстояние до начала

и цель, глубина в дереве и т.д.

Motion Planning Baselines

Мы сравниваем greedy best-first поиск с двумя широко используемыми эвристиками - евклидово расстояние (hEUC) и манхэттенское расстояние (hMAN). Мы также используем алгоритм A\* в качестве основы с эвристикой евклидового расстояния (hEUC). Кроме того, мы сравниваем с алгоритмом MHA\*

который, как было доказано, является эффективным способом объединения нескольких, часто несвязанных, эвристик.

Еще будем использовать упрощенную версию, a round-robin fashion для 3ёх эвристик — [hEUC, hMAN, dOBS], где dOBS — евклидово расстояние до ближайшей, известной ячейки препятствия в среде.

Machine Learning Baselines

Мы рассматриваем два базовых уровня обучения (а) контролируемое обучение (SL) с данными развертывания с πOR и (b) обучения с подкреплением с использованием эволюционных стратегий

(CEM) и Q-Learning (QL) с аппроксимирующей функцией

Материалы

[План работы над проектом](https://docs.google.com/spreadsheets/d/1S4vT303WcSwH8fg_fAF8PBAeVKWHNR55uHqVK7keqNM/edit?usp=sharing)

[Гитхаб репозиторий](https://github.com/SmolkinaJulia/RL_SaIL_project) - Юля

<https://github.com/Gricha1/heuristic_learning> - Гриша

[Наш гугл диск со всеми материалыми](https://drive.google.com/drive/folders/1I6K_ROVSOLrCIqrVXC5dKMKkIZC-3l1Q?usp=sharing)