

# Простейшие методы обучения с подкреплением в задаче синтеза мажоритарных схем

Самбурский Александр, факультет ВМК

2021

# Постановка задачи

Данная работа нацелена на выявление основных свойств и возможностей алгоритмов обучения с подкреплением в случае нестационарной среды работы программы.

В рамках задачи моделирования ФАЛ в базисе функций большинства рассматривается два направления:

- Методы упрощения
- Методы синтеза

## Постановка задачи

Данные направления реализуются с помощью алгоритмов обучения с подкреплением. Для программы выдвигаются следующие требования:

- Упростить задачу перебора
  - Сделать перебор контролируемым
  - Увеличить скорость работы ценой точности
- Обеспечить обучаемость во время решения задачи

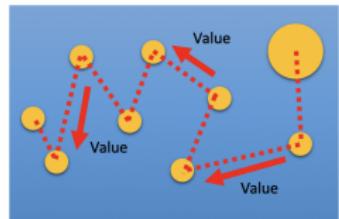
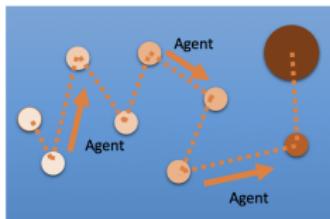
# Общая структура алгоритма

Агенту (исполнителю алгоритма) предоставляется определённый набор действий. При выполнении "правильного" набора действий агент способен решить задачу. Обучение заключается в приведении агента к этой оптимальной функциональности.



# Процесс обучения

Для того, чтобы агент обучался, производится его "запуск" в среду. В ней он совершает определённые действия, после чего производится уточнение таблиц соответствия состояний и оптимальных действий. Это дополнение производится итерационно, в порядке, противоположном исходному направлению переходов агента. Повторение этой процедуры и является обучением.



## Пересчёт таблицы оптимальных действий

После завершения цепочки действий все задействованные позиции в таблице оптимальности обновляются по следующему правилу:

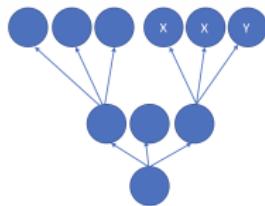
$$E[R|s_t, a_t] = r_t + \gamma E[R|s_{t+1}, a_{t+1}],$$

где  $t$  - дискретный момент времени очередного состояния агента,  $s_t$ ,  $a_t$  и  $r_t$  - состояние агента, совершённое им действие и полученная награда соответственно в момент времени  $t$ .  $R$  - общий "рейтинг" оптимальности пары "состояние - действие",  $\gamma$  - коэффициент дисконтирования.

Цель агента - максимизировать это математическое ожидание.

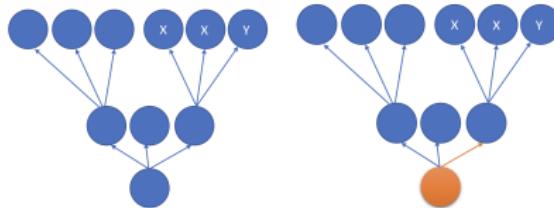
# Упрощение мажоритарных схем

В данной задаче агенту подаётся схема, которую требуется упростить. В качестве действий агенту предоставляется перемещение по ней и локальные упрощения.



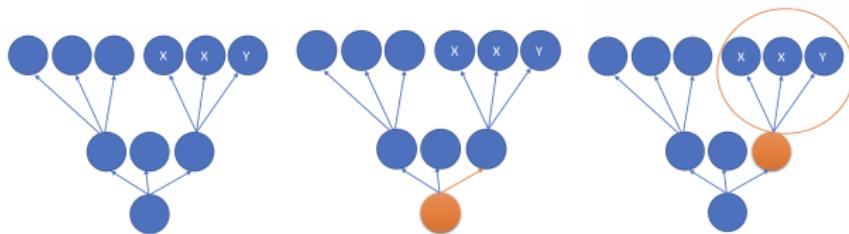
# Упрощение мажоритарных схем

В данной задаче агенту подаётся схема, которую требуется упростить. В качестве действий агенту предоставляется перемещение по ней и локальные упрощения.



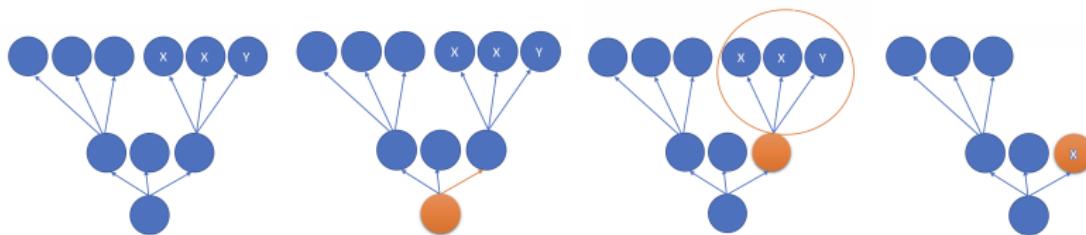
# Упрощение мажоритарных схем

В данной задаче агенту подаётся схема, которую требуется упростить. В качестве действий агенту предоставляется перемещение по ней и локальные упрощения.



# Упрощение мажоритарных схем

В данной задаче агенту подаётся схема, которую требуется упростить. В качестве действий агенту предоставляется перемещение по ней и локальные упрощения.



# Архитектура алгоритма

- Одна итерация обучения - эпизод - выполнение последовательности действий с последующим пересчётом их рейтинга.

# Архитектура алгоритма

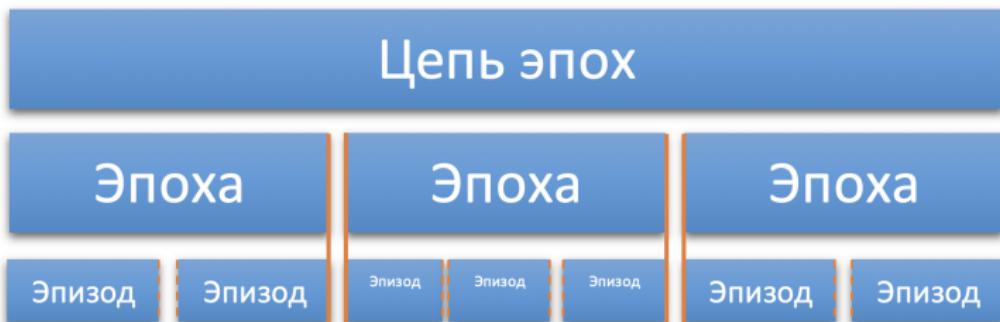
- Одна итерация обучения - эпизод - выполнение последовательности действий с последующим пересчётом их рейтинга.
- Набор эпизодов, последний из которых локально упрощает схему - эпоха. За время эпизодов агент обучается (Exploration), и накопленный "опыт" выливается в последнем эпизоде эпохи (Exploitation).

# Архитектура алгоритма

- Одна итерация обучения - эпизод - выполнение последовательности действий с последующим пересчётом их рейтинга.
- Набор эпизодов, последний из которых локально упрощает схему - эпоха. За время эпизодов агент обучается (Exploration), и накопленный "опыт" выливается в последнем эпизоде эпохи (Exploitation).
- Цепь эпох составляет весь процесс работы программы.

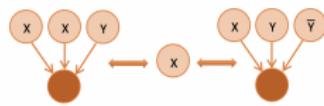
# Архитектура алгоритма

- Одна итерация обучения - эпизод - выполнение последовательности действий с последующим пересчётом их рейтинга.
- Набор эпизодов, последний из которых локально упрощает схему - эпоха. За время эпизодов агент обучается (Exploration), и накопленный "опыт" выливается в последнем эпизоде эпохи (Exploitation).
- Цепь эпох составляет весь процесс работы программы.

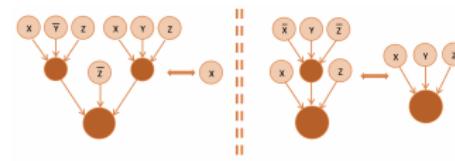


# Применяющиеся упрощения

1) Простейшие фундаментальные действия



2) Регулярные действия с функциональными поддеревьями глубины 2



3) Операции из  $\psi$ -аксиоматики

$$\Psi \left\{ \begin{array}{l} \text{Релевантность } \Psi.R M(x, y, z) = M(x, y, z_{x/y}) \\ \text{Ассоциативность дополнения } \Psi.C \\ M(x, u, M(y, u', z)) = M(x, u, M(y, x, z)) \\ \text{Подстановка } \Psi.S M(x, y, z) = \\ = M(v, M(v', M_{v/u}(x, y, z), u), M(v', M_{v/u'}(x, y, z), u')) \end{array} \right.$$

# Применяющиеся алгоритмы

Рассмотренные ниже алгоритмы отличаются подходом к определению границы эпизодов и эпох.

## Упрощённый

Алгоритм производит фиксированное количество эпизодов в эпохе, производя поиск и отбор наиболее эффективных упрощений.

## Модифицированный

Алгоритм производит ограниченное количество эпизодов в эпохе до тех пор, пока не наткнётся на первый "упрощающий эпизод". Дополнительно к этому он сохраняет контекст неудачных эпизодов.

# Преимущества и недостатки алгоритмов

## Упрощённый

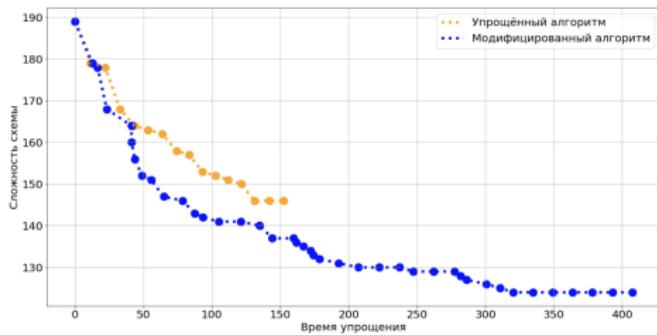
- + Стабильно находит возможности упрощения
- + Имеет возможность выбора лучших видимых упрощений
- + Не подвержен проблеме нестационарности среды
  
- Ликвидирована постепенная настраиваемость на текущую задачу (отсутствие памяти)
- Ошибка выбора гиперпараметра числа действий из архитектуры может сильно замедлить процесс

## Модифицированный

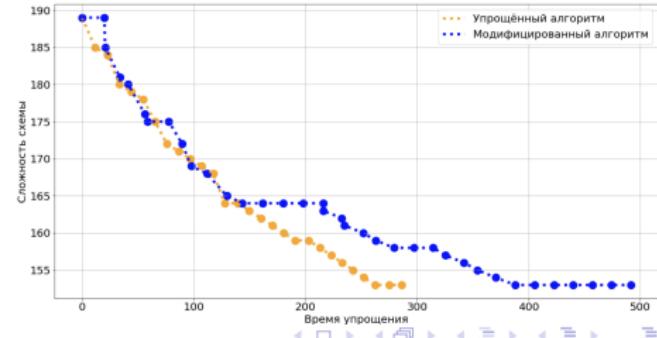
- + При определённых условиях имеет возможность быстро находить возможности упрощения
- + Переводит неудачу эпизода в увеличение потенциала последующих
- + Настраивается на текущую схему
- + Выбор гиперпараметров обучения ограничивает только его перебор, но не замедляет его работу
  
- Небольшое снижение стабильности упрощения (это может проявляться в том, что иногда этот алгоритм из-за "сверх-неудачности" неудачных эпизодов сохраняет в памяти искажённый контекст, что мешает ему в дальнейшем)
- Теряет свою эффективность в случае сверх-быстрого изменения схемы (многократного увеличения показателя нестационарности среды)

# Работа алгоритмов

На верхнем рисунке представлена ситуация выигрыша модифицированного алгоритма. На больших схемах важную роль играет контекст, и при корректной инициализации модифицированный алгоритм будет по всем параметрам обгонять упрощённый. На нём же видны характерные лестничные спады модифицированного алгоритма.



На нижнем рисунке изображено преимущество стабильности обучения упрощённого алгоритма. Здесь же видно, насколько плавнее динамика обучения первого алгоритма. Это является следствием фиксированности числа его итераций. Время обучения задано в секундах.



# Используемые методы для упрощения схем

- Методы Монте-Карло как основа архитектуры
- Свойства вытесняющих штрафов
- $\epsilon$ -жадные сходящиеся алгоритмы
- Внедрение гиперпараметров, отвечающих за контроль перебора
- Разделение агентов по допустимым упрощениям
- "Dropout" частей схемы

# Синтез мажоритарных схем

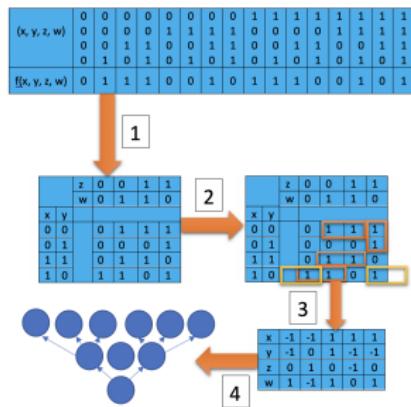
В данной задаче требуется составить по входящему набору значений ФАЛ её модельное представление в виде схемы мажоритарных элементов. В качестве дополнительных требований выдвигается:

- Снижение/контроль времени работы синтезатора
- Снижение сложности и глубины схемы, насколько это позволяет сделать данный перебор

# Архитектура алгоритма

Общие шаги алгоритма синтеза следующие:

- По входящему вектору значений формируется карта Карно целевой функции.
- Затем в ней идёт поиск минимального (по мнению агента) покрытия характерного множества. Поиск происходит итеративно для каждой ещё не покрытой "единицы".
- Далее полученное покрытие кодируется в соответствии с аргументами функции и переходит в матричный вид.
- Программа рекурсивно строит по этой матрице результирующую схему.



# Применимость обучения с подкреплением

Важно отметить, что единственное место где перебор напрямую влияет на вычислительную сложность алгоритма - это поиск покрытия. Остальные части алгоритма представляют точные алгоритмы, не требующие аппроксимации.

В связи с этим обучение с подкреплением применяется на этапе перебора для снижения его сложности (второй этап). Рассмотрим его подробнее.

## Этап поиска минимальной ДНФ

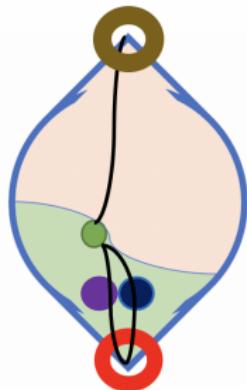
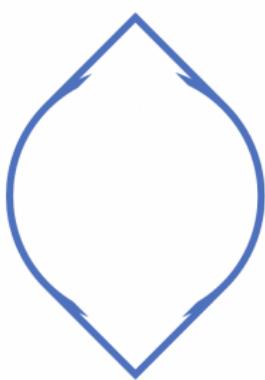
- Для каждой эпохи выбирается непокрытая единица и происходит поиск её максимального покрытия.

## Этап поиска минимальной ДНФ

- Для каждой эпохи выбирается непокрытая единица и происходит поиск её максимального покрытия.
- Среда агента может быть схематично представима в виде двуугольного поля. Каждая её точка - определённая конфигурация размеров грани, покрывающей текущую единицу. Чем выше точка в поле, тем больше грань.

## Этап поиска минимальной ДНФ

- Для каждой эпохи выбирается непокрытая единица и происходит поиск её максимального покрытия.
- Среда агента может быть схематично представима в виде двуугольного поля. Каждая её точка - определённая конфигурация размеров грани, покрывающей текущую единицу. Чем выше точка в поле, тем больше грань.
- Цель агента - найти наиболее "высокую" точку в этом поле.



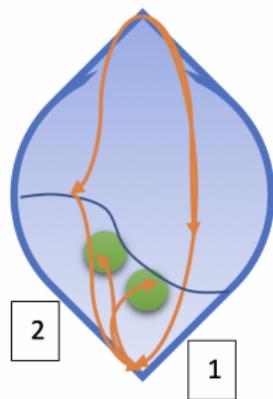
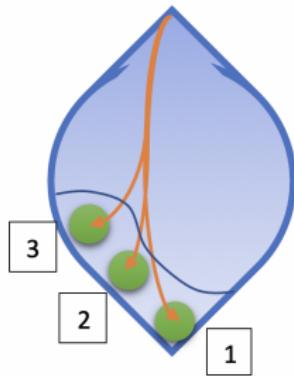
		Z	0	0	1	1
		W	0	1	1	0
X	Y					
0	0		0	1	0	0
0	1		1	1	0	0
1	1		1	1	0	1
1	0		0	0	1	0

# Этап поиска минимальной ДНФ

Реализовано две простые стратегии поиска. В обеих стратегиях эпизод начинается в верхней точке поля (полной Карты Карно).

Первая настраивает покрытие, исходя из лучшей конфигурации на данный момент, с некоторой вероятностью меняя "скамающее" действие.

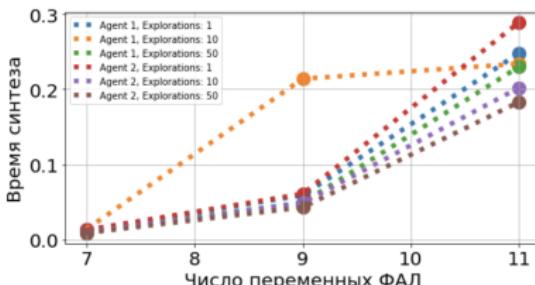
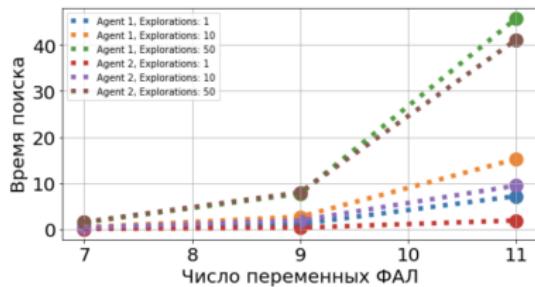
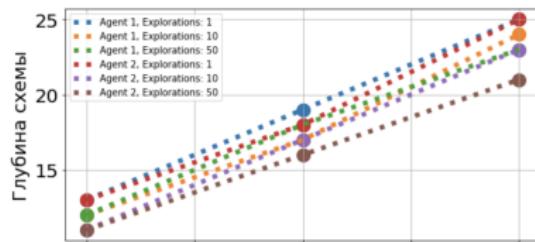
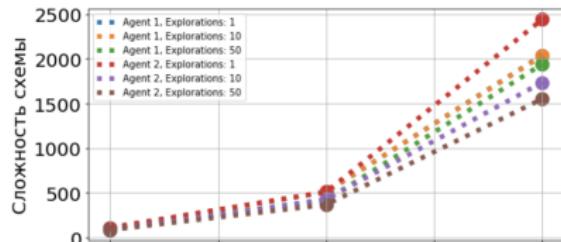
Вторая производит поиск "крюками". Это позволяет повысить разнообразие поиска лучшего покрытия.



В качестве гиперпараметров выступает число эпизодов поиска, а так же вероятность смены действия.

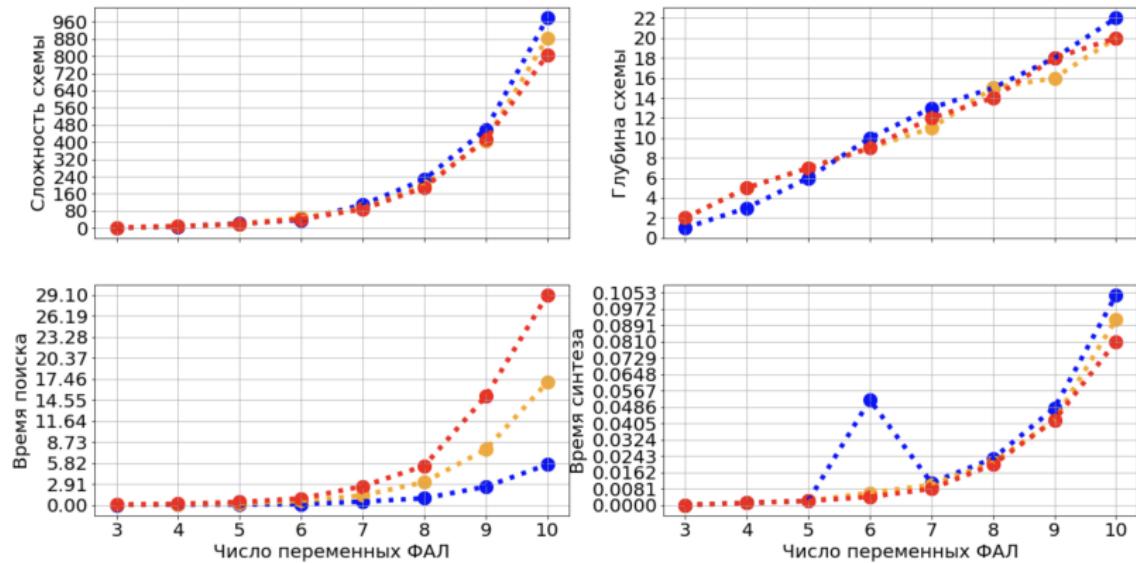
# Сравнение стратегий

На графиках представлены основные характеристики результирующих покрытий на указанных методах (время поиска и синтеза указано в секундах).



# Общая оценка работы синтеза

Ниже изображена статистика работы всей программы синтеза.  
Сложность перебора повышается в порядке  
синего-жёлтого-красного.



# Общие характеристики задачи синтеза

- В отличие от упрощения задача синтеза ТРЕБУЕТ практически полного исследования среды.
- По отношению к RL сложность задач данного типа заключается в обгоне прямых алгоритмов по эффективности.
- Простейшим образом контроль перехода от точности решения к ускорению поиска реализуется гиперпараметрами.
- Гиперпараметры фиксируют объём исследований - это может быть как хорошо, так и пагубно.

# Общие выводы

- Исследована возможность применения алгоритмов RL к задачам упрощения и синтеза схем. Произведено разграничение возможностей обучения с подкреплением по отношению к ним.
- Проанализированы вопросы нестационарности среды работы методов и варианты их преодоления.

Спасибо за внимание!