ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»

Мисник Андрей Сергеевич

ПРИМЕНЕНИЕ СТРУКТУРНОГО МАШИННОГО ОБУЧЕНИЯ ДЛЯ ИГР С НЕПОЛНОЙ ИНФОРМАЦИЕЙ

Специальность 09.04.01 —
«Информатика и вычислительная техника»

Литературный обзор

Научный руководитель:
д.ф.-м.н. Виноградов Дмитрий Вячеславович

Москва 2024

Аннотация

В данном обзоре представлен анализ современных методов, используемых для структурного обучения с подкреплением в контексте игр с неполной информацией. Рассматриваются методы, основанные на Марковских процессах принятия решения и, в частности, на алгоритме Monte Carlo Tree Search. Кроме того, в обзоре указаны теоретические возможности и ограничения Monte Carlo Tree Search; рассмотрены различные оптимизации алгоритма, предложенные для решения проблем, связанных с неполнотой информации - таких, как экспоненциальный рост размера дерева решения.

Содержание

Стр.

Введение 4

Глава 1. Постановка задачи 5

Глава 2. Monte Carlo Tree Search 6

2.1 Описание алгоритма 6

2.2 Политика выбора узлов 8

Глава 3. MCTS для игр с неполной информацией 9

3.1 Детерминация состояний 9

3.2 Информационные наборы 10

3.3 Детерминация переходов 11

Глава 4. Структурное обучение и ВКФ 13

Заключение 15

Список литературы 16

Введение

Обучение с подкреплением представляет собой методику в машинном обучении, в которой агент обучается взаимодействовать с окружающей средой, принимая решения для достижения определенных целей. Агенты получают сигналы вознаграждения или наказания в зависимости от результатов своих действий, и их цель заключается в максимизации итогового вознаграждения.

Структурное обучение с подкрелением распространяет эти концепции на сложные задачи, требующие работы с богатыми структурами данных. Использование структурного обучения способно подтолкнуть агента учитывать контекст и скрытые зависимости, а так же сформировать человекочитаемый результат обучения - некоторый набор правил, которыей определяет оптимальное действие в зависимости от некоторых условий. Это может быть полезно, например, для ручного анализа экспертами в предметной области исходной задачи.

К сожалению, большинство современных моделей машинного обучения хранит переработанную информацию об объектах и результаты обучения отдельно от исходных признаков, теряя структурную взаимосвязь между ними. Кроме того, дистилляция, как единственная общепринятая методика передачи знаний между моделями, не способна передать структурные особенности рассматриваемой среды.

Дополнительными сложностями в классе рассматриваемых задач могут стать недостаток информации о состоянии среды и недетерминированность переходов между ее состояниями. Для решения этих проблем рассмотрены такие подходы, как детерминация состояний, детерминация переходов и метод информационных наборов.

1. Постановка задачи

Задачу обучения с подкреплением можно сформулировать как задачу Марковского процесса принятия решений (МППР). В любой момент времени t среда находится в состоянии из множества S. Для взаимодействия со средой в каждый момент времени агент выбирает действие из множества A, после чего среда переходит из одного состояния в другое. Вероятности перехода из одного состояния среды в другое называются моделью переходов $T(s, a, s')$, где $s \in S$ - состояние, в котором выполнено действие $a \in A$, после чего среда перешла в состояние $s' \in S$. За действия в среде агент получает вознаграждения, которые определяются функцией вознаграждения $R(s, a)$.

В модели структурного обучения с подкреплением помимо пространства состояний вводится набор бинарных признаков состояний и фиксируется отображение F $(\forall s \in S : F(s) = \{f_1, f_2, .., f_n\}, f_i \in \{0, 1\})$.

Целью агента в структурном обучении является порождение множества гипотез H $(\forall h_i \in H : h_i = \{F_i, A_i\}, F_i \subset \{0, 1\}^n)$, где каждая гипотеза имеет свойство $\forall s_t \in S : F(s_t) \in F_h \Rightarrow A_h \sim \pi(\cdot|s_t)$.

Для формализации задачи обучения в играх с неполной информацией используется модель частично наблюдаемых Марковских процессов принятия решений: дополнительно вводится понятие наблюдений: O — множество возможных наблюдений, которые агент может получить, $\Omega : S' \times A \times O \to [0,1]$ — функция наблюдений, определяющая вероятность получения наблюдения o после выполнения действия a и перехода в состояние s'.

2. Monte Carlo Tree Search

Monte Carlo Tree Search (MCTS) - это алгоритм принятия решений, который заключается в поиске в комбинаторных пространствах, представленных деревьями [1]. В контексте задач обучения с подкреплением, состояния являются узлами деревьев, а переходы между ними обозначаются ребрами.

MCTS был предложен в 2006 году для создания ИИ по игре в Го [2]. В 2016 году была представлена программа AlphaGo, основанная на этом алгоритме.

В практических приложениях задача поиска зачастую оказывается слишком сложной для базовой версии алгоритма, в связи с чем в современных работах исследователи демонстрируют различные оптимизации, позволяющие сократить вычислительную сложность, чтобы алгоритм выдавал качественные результаты при практических вычислительных ограничениях. Некоторые из этих оптимизаций будут рассмотрены в этом обзоре.

2.1 Описание алгоритма

MCTS - это итеративный алгоритм, который выполняет поиск в пространстве состояний и собирает статистические данные о решениях, доступных в конкретных состояниях. MCTS, как правило, применим к Марковским процессам принятия решений. Модель МППР позволяет моделировать исследуемую среду с последовательными решениями, которые необходимо предпринять, чтобы получить искомое вознаграждение.

В нетривиальных задачах невозможно выполнить полный поиск в пространстве состояний. В практических приложениях для работы MCTS выделяется некоторый вычислительный бюджет, который может быть определен либо количеством итераций, либо временем, доступным для

принятия решения. MCTS - это алгоритм, который можно остановить в любое время и принять наилучшее на данный момент действие.

Каждая итерация алгоритма состоит из 4 действий:

1. Выбор - алгоритм выполняет поиск в той части дерева, которая уже была представлена в памяти. Выбор всегда начинается с корневого узла и на каждом уровне выбирает следующий узел, пока не попадет в узел, соответствующий конечному состоянию игры.
2. Расширение - если выбор не достиг конечного состояния, на этом шагу добавляется по крайней мере один новый дочерний узел к дереву, представленному в памяти. Новый узел соответствует состоянию, которое достигается при выполнении последнего действия на этапе выбора.
3. Симуляция - выполняет полную случайную симуляцию игры/задачи из конкретного состояния в узле, т.е. достигает конечного состояния и вычисляет награду.
4. Обратное распространение результата - возвращает данные о награде (результат симуляции) ко всем узлам на пути от последнего симулируемого узла в дереве к корневому.

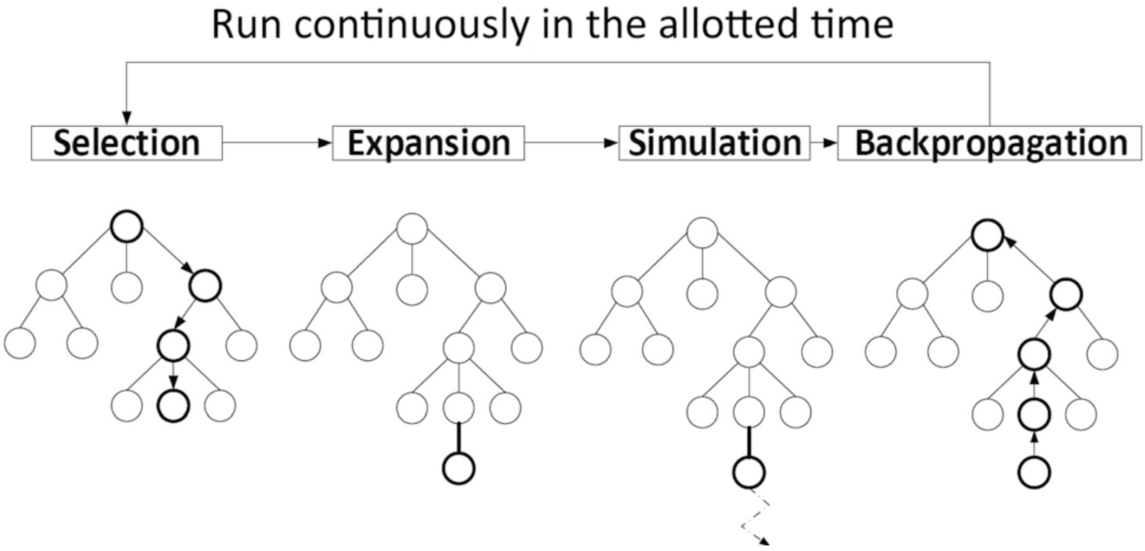

Рис. 2.1 — Графическое описание фаз алгоритма MCTS

2.2 Политика выбора узлов

Качество работы алгоритма зависит от политики, по которой выбирается следующий узел на первой фазе алгоритма (выборе). Выбор узлов для симуляции должен искать наилучший баланс между исследованием новых опций (exploration) и использованием уже найденных ранее опций (exploitation). Основной используемый метод - UCT, описан в статье [3]. Для каждого действия из некотого состояния предлагается вычислить сумму средней награды и функции, основанной на количестве раз, которое это действие было выбрано:

$$a^* = \arg\max_{a \in A(s)} \left\{ Q(s,a) + C\sqrt{\frac{\ln[N(s)]}{N(s,a)}} \right\},$$

где $A(s)$ — множество действий, доступных в состоянии s, $Q(s,a)$ обозначает средний результат выполнения действия a в состоянии s в симуляциях, проведенных ранее, $N(s)$ — число посещений состояния s в предыдущих итерациях, а $N(s,a)$ — число выборов действия a в состоянии s. Константа C контролирует баланс между exploration и exploitation.

Есть и другие вариации предложенной формуле - например, в работе [4] дополнительно рассматривается дисперсия средней награды при выборе действия a.

В статье [5] предлагается альтернативный вариант стратегии exploration-exploitation, в котором для оценки среднего и дисперсии Q-values используется отдельная модель. Вместо выборки в пространстве состояний, рассматриваются переходы из буфера воспроизведения.

3. MCTS для игр с неполной информацией

Как было сказано ранее, основной проблемой алгоритма MCTS на деревьях игры является экспоненциальный рост количества узлов, что также известно как "высокий фактор ветвления". Кроме того, переходы в некоторых играх могут быть недетерминированными - например, бросок шестигранного кубика в конкретном состоянии может породить 6 новых состояний в зависимости от результата броска. В последние годы было разработано несколько модификаций MCTS, направленных на решение этой проблемы, некоторые из них будут разобраны далее.

Кроме того, стоит отметить и некоторые альтернативные оптимизации, не рассмотренные в этом обзоре:

- Ограничение количества доступных действий.
- Преждевременное завершение этапа моделирования.
- Альтернативы UCT — другие подходы к этапу выбора MCTS.

3.1 Детерминация состояний

Одним из методов решения проблемы случайности в играх является детерминация состояний, которая из модели недетерминированной игры создает игру с полной информацией и добавляет в выборку несколько сформированных по такому принципу игр. В статье [6] используется понятие "дерева Монте-Карло с полной информацией" и указываются следующие проблемы такого подхода:

1. Объединение стратегий невозможно, потому что процесс поиска может принимать разные решения в разных моделях игр в зависимости от детерминации поддерева.
2. Отдельное использование вычислительного бюджета для каждой из созданных моделей.
3. Наилучший выигрыш нельзя определить рекурсивно, поскольку в результате детерминации могли потеряться некоторые состояния.

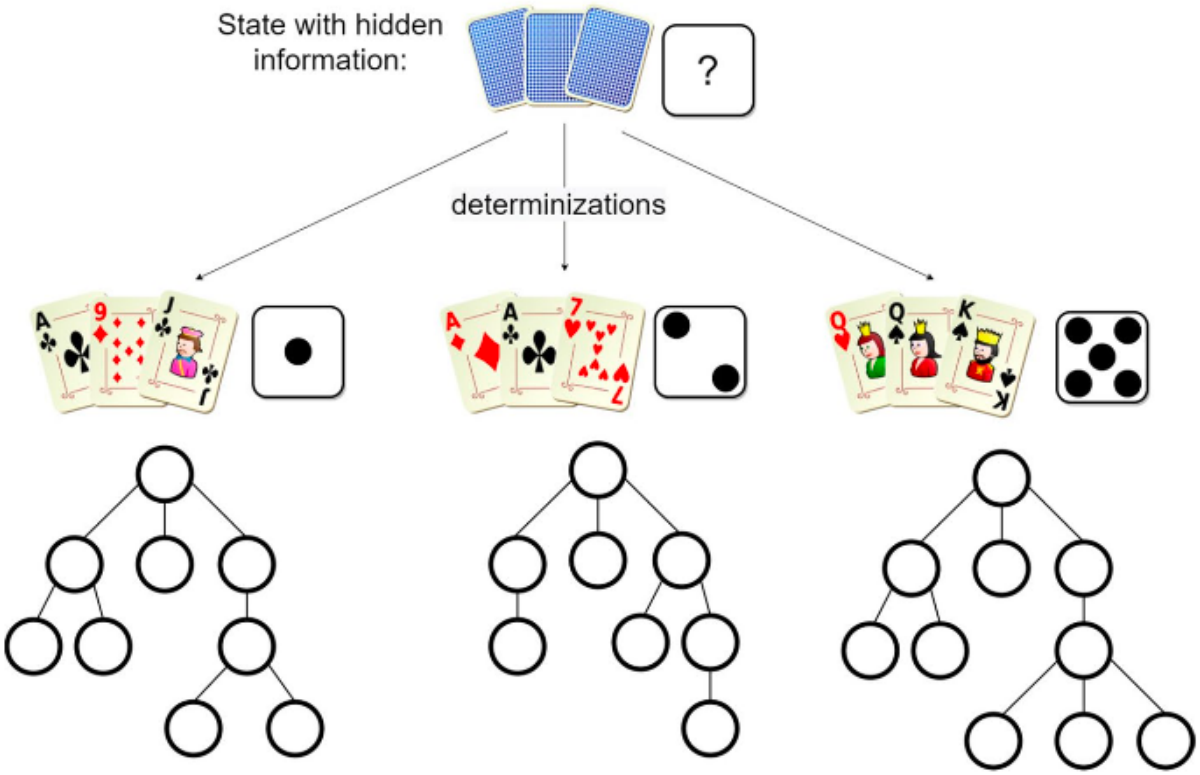

Рис. 3.1 — Пример детерминации состояний

3.2 Информационные наборы

Этот метод подразумевает объединение состояний игры, соответствующих разным узлам дерева, но неотличимых с точки зрения игрока. В алгоритме MCTS, основанном на информационных наборах, каждый узел содержит информационный набор, который позволяет игроку действовать на основе своих собственных наблюдений.

В статье [7] вводится понятие онлайн выборки результатов (OOS). Дерево строится поэтапно, и на каждом этапе учитываются наиболее значимые для итогового результата узлы.

Эксперименты в приведенных работах показали, что предложенные алгоритмы стабильно приближаются к равновесию Нэша, это достигается с помощью информационного набора о текущем состоянии игры.

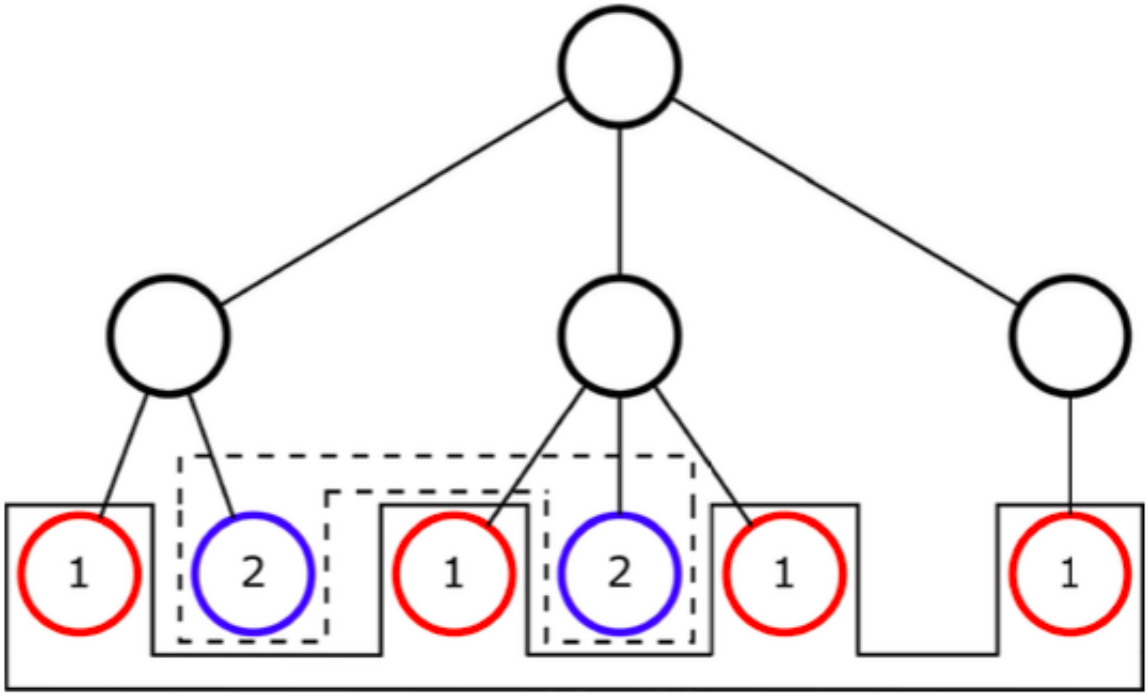

Рис. 3.2 — Пример объединения состояний в группы неотличимых

3.3 Детерминация переходов

Этот метод подразумевает замену случайных переходов на детерминированные путем ограничения игровых правил и обучение нескольких агентов по отдельности в условиях локальной детерминированности.

Пример реализации для игры Magic: The Gathering описан в статье [8] и назван "ансамблевой детерминацией".

В каждой из игровых моделей некоторым образом ограничиваются игровые правила, что порождает отдельные деревья игры с разными ограничениями - например, некоторые деревья ограничены в глубину, а некоторые - количеством потомков. Результаты, приведенные в статье, показали, что обучение на модели бинарного дерева выдало приемлемые результаты с наилучшим временем отклика среди всех моделей, и каждая отдельно взятая модель улучшила время отклика исходного дерева в $10-20$ раз.

Short name	Description	Trees	Simulations per tree	UCT constant	Win/loss reward	Reward discount	Move pruning	Simulation strategy	Tree structure
AP	All Possible Deals / Uniform Random Rollouts	40	250	1.5	1 / 0	1	None	Uniform Random	All Possible Deals
BA	Baseline	40	250	1.5	1 / 0	0.99	Land	Reduced Rules	Unlimited Degree
IL	Interesting Simulations (Leaf. 1% of sim budget)	40	250	1.5	1 / 0	0.99	Land	**Reduced Rules / Interesting (Leaf)**	Unlimited Degree
IR	Interesting Simulations (Root. 5% of sim budget)	40	250	1.5	1 / 0	0.99	Land	**Reduced Rules / Interesting (Root)**	Unlimited Degree
NL	Negative Reward for Loss	40	250	1.5	**1 / -1**	0.99	Land	Reduced Rules	Unlimited Degree
MP	Dominated Move Pruning	40	250	1.5	1 / 0	0.99	**Dominated**	Reduced Rules	Unlimited Degree
BT	Binary Tree (Descending Mana Cost)	40	250	1.5	1 / 0	0.99	Land	Reduced Rules	**Binary**

Рис. 3.3 — Модели игр, используемые для детерминации переходов в игре MTG

4. Структурное обучение и ВКФ

Все приведенные ранее в этом обзоре подходы объединяет одна концептуальная проблема, упомянутая во введении - в результате обучения теряется структурная взаимосвязь между состояниями игры, что не позволяет интерпретировать сгенерированные политики на человеческом языке.

В статье [9] предлагается инновационный подход к структурному обучению с подкреплением, результатом обучения которого являются правила, которые при наличии у состояния определенного свойства утверждают наилучшее действие для этого состояния.

Между состояниями вводится функция сходства, основанная на их признаковом представлении: $sim(S_1, S_2) = F(S_1)\&F(S_2)$. Гипотезы формируются для наборов сходных параметров между похожими состояниями, в которых предпринимается одно и то же действие. Кроме того, гипотезы должны быть сформированы так, чтобы у них не было "отрицательных примеров" - состояний, обладающих этими же свойствами, но имеющих другое оптимальное действие.

Алгоритм порождения гипотез устроен следующим образом:

1. Каждая гипотеза-кандитат порождается независимо с использованием цепи Маркова
2. Для каждого из затрагиваемых в гипотезе состояний проверяется оптимальность предпринятого действия (используя результат симуляции MCTS для этого состояния)
3. Гипотеза подтверждается, если действие оптимально для всех затрагиваемых состояний

Data: множество обучающих (+)- и (-)-примеров; число N порождаемых ВКФ-гипотез
Result: случайная выборка S ВКФ-гипотез
$O :=$ (+)-примеры, $F :=$ признаки; $I \subseteq O \times F$ формальный контекст для (+)-примеров;
$C :=$ (-)-примеры; $S := \emptyset$; $i := 0$;
while $(i < N)$ **do**
  породить кандидата $\langle A, B \rangle$ с помощью цепи Маркова;
  $hasObstacle :=$ **false**;
  for $(c \in C)$ **do**
    if $(B \subseteq c')$ **then**
      $hasObstacle :=$ **true**;
    end
  end
  if $(hasObstacle =$ ***false***$)$ **then**
    $S := S \cup \{\langle A, B \rangle\}$;
    $i := i + 1$;
  end
end

Рис. 4.1 — Процедура индуктивного обобщения ВКФ для порождения гипотез

После порождения можно симулировать игру, выбрав в качестве политики следование порожденным правилам. После применения порожденных гипотез может быть вычислен эмпирический риск (средняя ошибка алгоритма на обучающей выборке). Далее порождается следующий набор гипотез другого размера, и производится сравнение эмпирического риска для полученных наборов гипотез. Алгоритм ВКФ завершается, когда эмпирический риск прекращает уменьшаться.

В работе Д.В. Виноградова предлагается запуск алгоритма порождения с последовательным увеличением количества гипотез вдвое, а также доказана теоретическая и практическая сходимоть алгоритма за линейное от количества состояний время.

Кроме того, демонстрируется использование приведенного метода для игры ”Крестики-Нолики” и для публичного набора данных "Mushrooms". Эти модели - примеры игр с полной информацией, но данный подход может быть применен и для игр с неполной информацией - это станет предметом дальнейших исследований.

Заключение

В данном обзоре представлены подходы к решению задачи обучения с подкреплением с использованием алгоритма Monte Carlo Tree Search и его модификаций. Потенциально, каждый из рассмотренных подходов может быть адаптирован для игр с неполной информацией. Подходы с объединением состояний среды в группы (методом "информационных наборов") показывают наилучшие сравнительные результаты и позволяют решать задачи с наиболее богатой структурой состояний.

Одним их немногих алгоритмов, который можно отнести именно к структурному обучению, является алгоритм ВКФ - он станет предметом дальнейшего изучения и будет адаптирован для игр с неполной информацией в рамках научной работы.

Список литературы

1. Maciej Świechowski et. al. Monte Carlo Tree Search: a review of recent modifications and applications // Artificial Intelligence Review. 2021.

2. Coulom Rémi. Efficient selectivity and backup operators in Monte-Carlo tree search // International conference on computers and games / Springer. 2006.

3. Kocsis Levente, Szepesvári Csaba. Bandit based monte-carlo planning // European conference on machine learning / Springer. 2006.

4. Gelly Sylvain, Wang Yizao. Exploration exploitation in go: UCT for Monte-Carlo go // NIPS: Neural Information Processing Systems Conference On-line trading of Exploration and Exploitation Workshop. NIPS, 2006.

5. Ott Julius, Servadei Lorenzo. MEET: A Monte Carlo Exploration-Exploitation Trade-Off for Buffer Sampling // ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2023.

6. Cowling Peter, Powley Edward, Whitehouse Daniel. Information set monte carlo tree search // IEEE Transactions on Computational Intelligence and AI in Games. T. 4. IEEE, 2012.

7. Lisỳ Viliam, Lanctot Marc, Bowling Michael H. Online Monte Carlo Counterfactual Regret Minimization for Search in Imperfect Information Games. // AAMAS. 2015.

8. Cowling Peter I, Ward Colin D, Powley Edward J. Ensemble determinization in monte carlo tree search for the imperfect information card game magic: The gathering // IEEE Transactions on Computational Intelligence and AI in Games. T. 4. IEEE, 2012.

9. Виноградов Дмитрий. Вероятностно-комбинаторный формальный метод обучения, основанный на теории решеток // ФИЦ ИУ РАН. 2018.