ПЛАНИРОВАНИЕ В RL

Сергей Николенко

СПбГУ — Санкт-Петербург

06 мая 2022 г.

Random facts:

- 6 мая 319 г. святая равноапостольная Нино крестила царя Грузии Мириана и его сановников
- 6 мая 1626 г. Петер Минёйт (Peter Minuit) за вещи стоимостью 60 гульденов (500-700 современных долларов) приобрёл у индейцев Манхэттен
- 6 мая 1840 г. был выпущен в обращение «Чёрный пенни»; их было напечатано более 68 миллионов, и сохранились не менее нескольких процентов, так что сейчас гашеный «Чёрный пенни» можно купить за несколько десятков долларов
- 6 мая 1872 г. был учреждён Кубок Англии по футболу (FA Cup), старейшее футбольное соревнование в мире; а 6 мая 1942 г. прошёл футбольный матч на стадионе «Динамо» в осаждённом Ленинграде («Динамо» — ЛМЗ, 6:0)
- 6 мая 1994 г. Елизавета II и Франсуа Миттеран открыли тоннель под Ла-Маншем
- 6 мая 2012 г. в России от Калужской до Болотной площади прошёл «Марш миллионов», повлёкший за собой «Болотное дело»

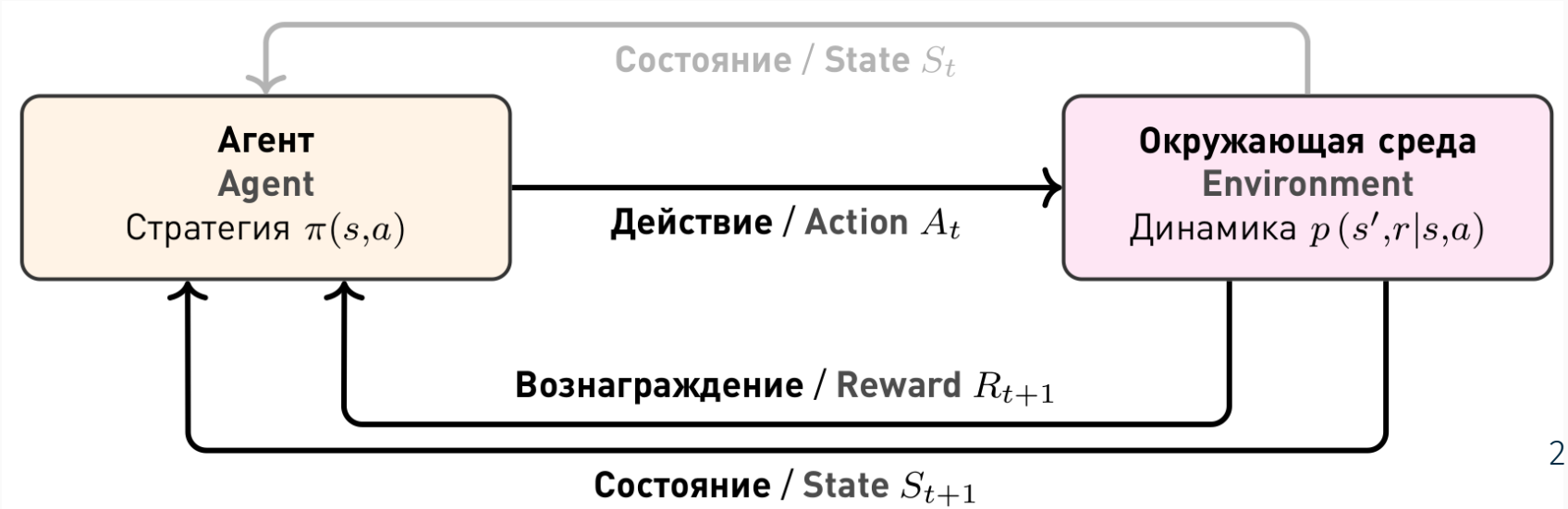
Обучение с подкреплением

- В прошлый раз мы ввели основные понятия динамики марковских процессов принятия решений:
  - собственно динамику процесса:

$$p\left(s', r \mid s, a\right) = p\left(S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a \mid;\right)$$

  - награды за каждый эпизод, начиная со времени t:

$$G_t = R_{t+1} + \gamma G_{t+1} = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1};$$



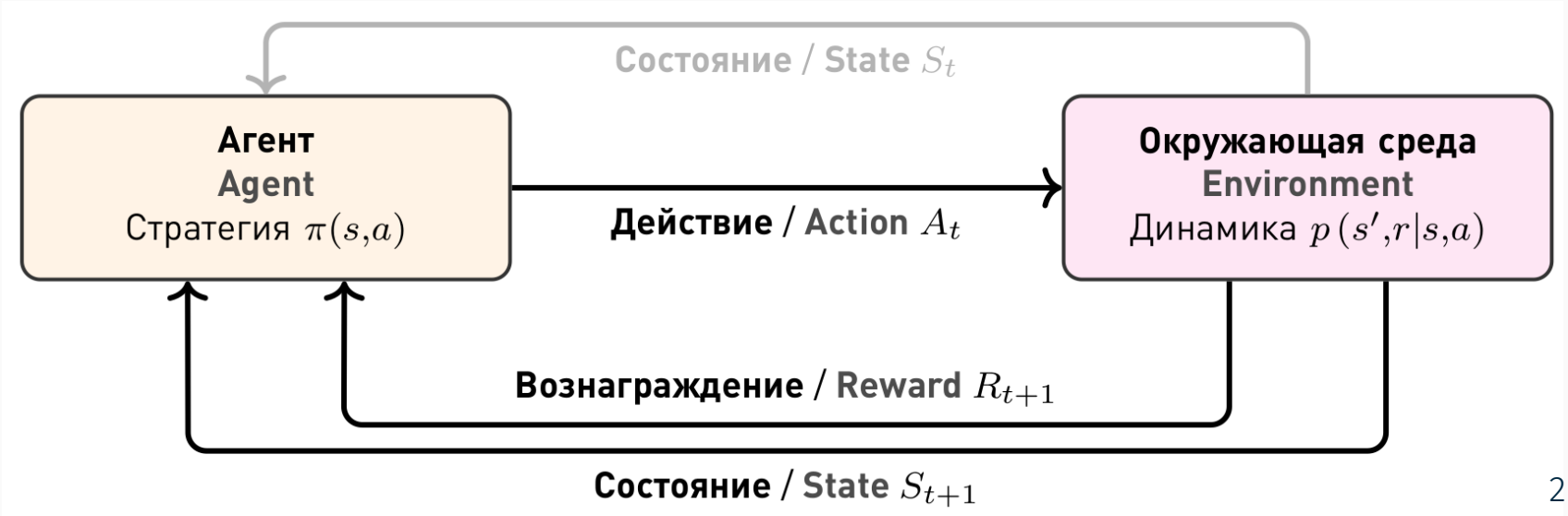
Обучение с подкреплением

- Определили функции значений V и Q

$$V_\pi(s) = \mathbb{E}_\pi\left[G_t \mid S_t = s\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right],$$

$$Q_\pi(s,a) = \mathbb{E}_\pi\left[G_t \mid S_t = s, A_t = a\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right]$$

- Выписали уравнения Беллмана и научились их решать.



Основные задачи

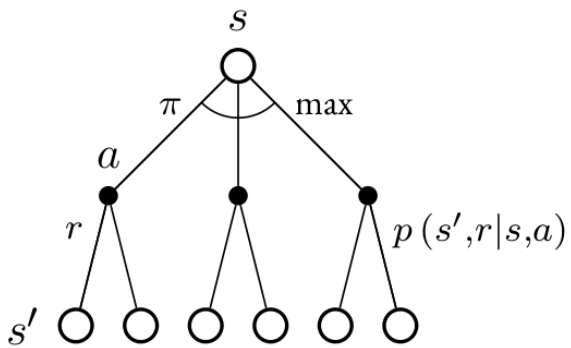
- Теоретически всё готово, но у нас много проблем:
  - уравнения знаем, но пока не знаем, как их решать, то есть как найти V^π для данного π?
  - разных стратегий очень, очень много — как найти оптимальную стратегию поведения агента в данной модели и соответствующие V^*?
  - но уравнений тоже не знаем — в реальности обычно P и R не даны, их тоже нужно обучить; как?
  - более того, их обычно даже записать не получится, слишком уж много состояний в любой реальной задаче… что делать?

- Давайте есть слона по частям…

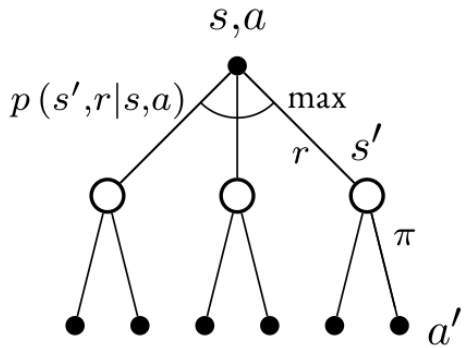
Диаграммы обхода

- Вспомним, о чём мы говорили, и заодно введём новый вид картинок: *диаграммы обхода* (backup diagrams)

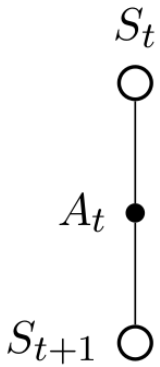
Уравнение Беллмана для V_*



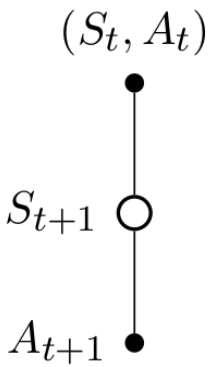
Уравнение Беллмана для Q_*



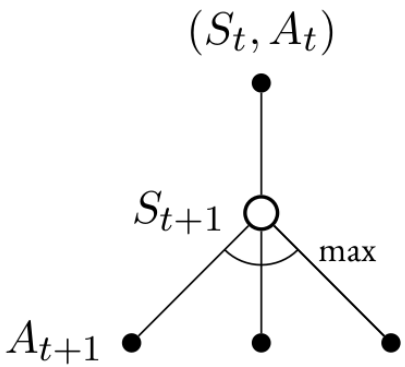
$TD(0)$



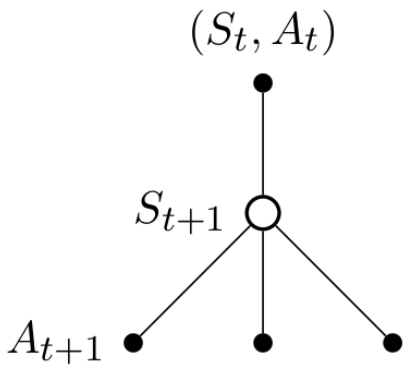
Sarsa



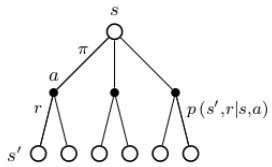
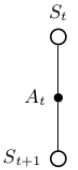
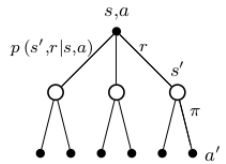
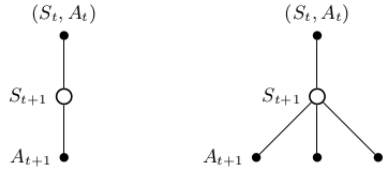
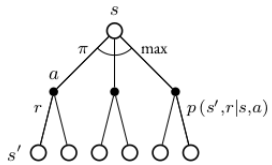
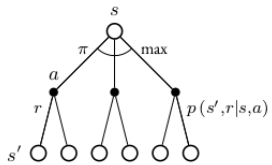
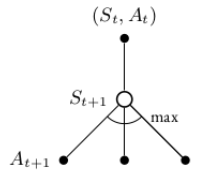
Q-обучение



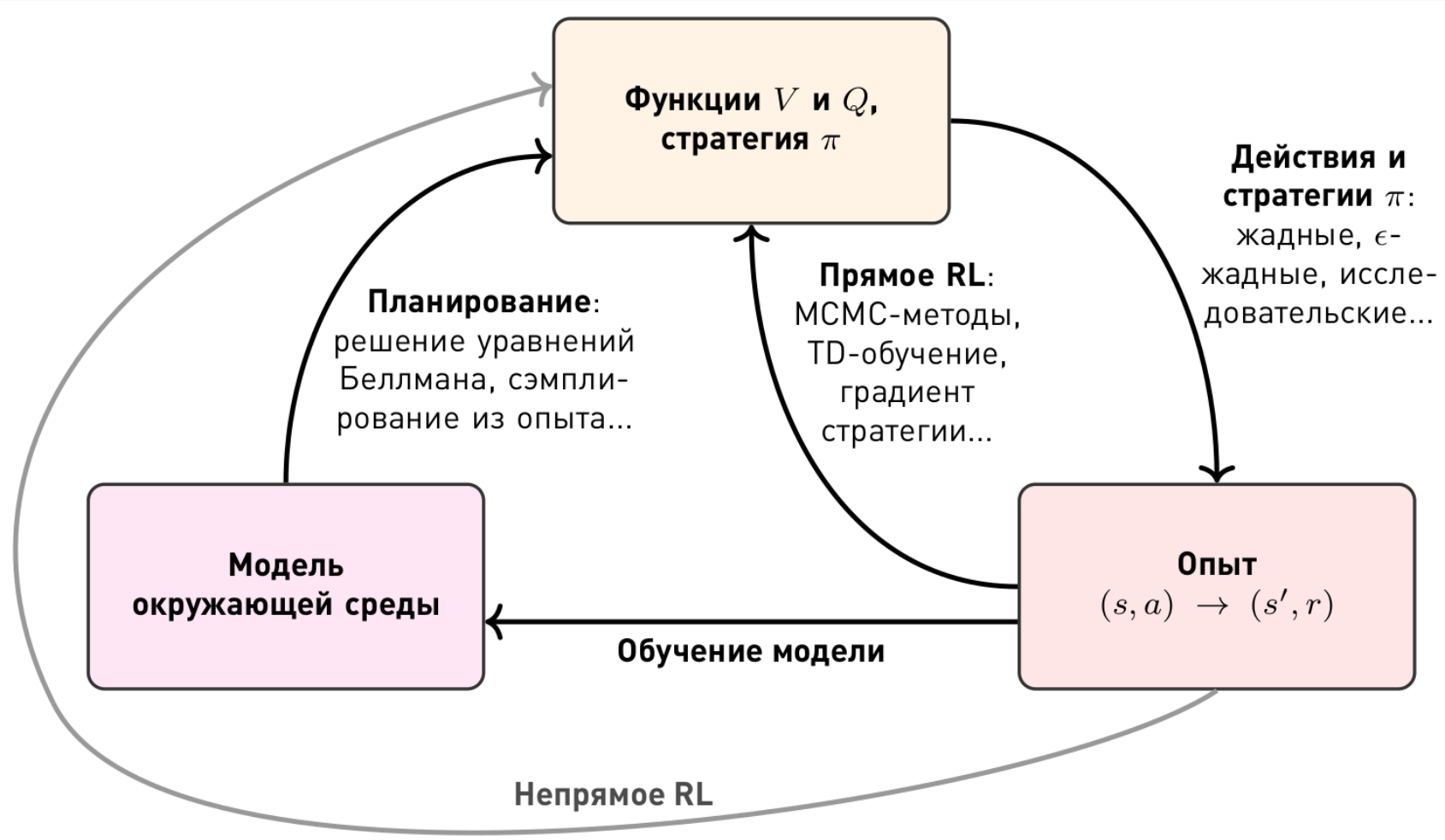
Sarsa с ожиданием



Таблица

	Функция	Обновление в ожидании	Обновление по выборке
Оценка стратегии π	$V_\pi(S_t) :=$	**Уравнение Беллмана** $\sum_{a\in\mathcal{A}} \pi(a\|S_t) \sum_{s'\in\mathcal{S}} \sum_{r\in\mathcal{R}} p(s',r\|s,a)\,(r+\gamma V_\pi(s'))$ 	**TD**(0) $V(S_t) + \alpha\,(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$ 
	$Q_\pi(S_t,A_t) :=$	**Уравнение Беллмана** $\sum_{s'\in\mathcal{S}} \sum_{r\in\mathcal{R}} p(s',r\|S_t,A_t)\,(r+\gamma \sum_{a'\in\mathcal{A}} \pi(a'\|s')\,Q_\pi(s',a'))$ 	**Sarsa, Sarsa с ожиданием** $Q(S_t,A_t) + \alpha\,(R_{t+1} + \gamma Q(S_{t+1},A_{t+1}) - Q(S_t,A_t))$ $Q(S_t,A_t) + \alpha\,\big(R_{t+1} + \gamma \sum_a \pi(a\|S_{t+1})\,Q(S_{t+1},a) - Q(S_t,A_t)\big)$ 
Управление, поиск π_*	$V_*(S_t) :=$	**Уравнение Беллмана** $\max_{a\in\mathcal{A}} \sum_{s'\in\mathcal{S}} \sum_{r\in\mathcal{R}} p(s',r\|S_t,a)\,(r+\gamma V_*(s'))$ 	
	$Q_*(S_t,A_t) :=$	**Уравнение Беллмана** $\sum_{s'\in\mathcal{S}} \sum_{r\in\mathcal{R}} p(s',r\|S_t,A_t)\left(r+\gamma \max_{a'} Q_*(s',a')\right)$ 	**Q-обучение** $Q(S_t,A_t) + \alpha\,(R_{t+1} + \gamma \max_a Q(S_{t+1},a) - Q(S_t,A_t))$ 

Обучение с подкреплением



Планирование

Планирование

- В прошлый раз мы ввели основные понятия динамики марковских процессов принятия решений:
  - собственно динамику процесса:

$$p\left(s', r \mid s, a\right) = p\left(S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\right);$$

  - награды за каждый эпизод, начиная со времени t:

$$G_t = R_{t+1} + \gamma G_{t+1} = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1};$$

  - функцию значения для состояний и пар состояние-действие:

$$V_\pi(s) = \mathbb{E}_\pi\left[G_t \mid S_t = s\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right],$$

$$Q_\pi(s, a) = \mathbb{E}_\pi\left[G_t \mid S_t = s, A_t = a\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right]$$

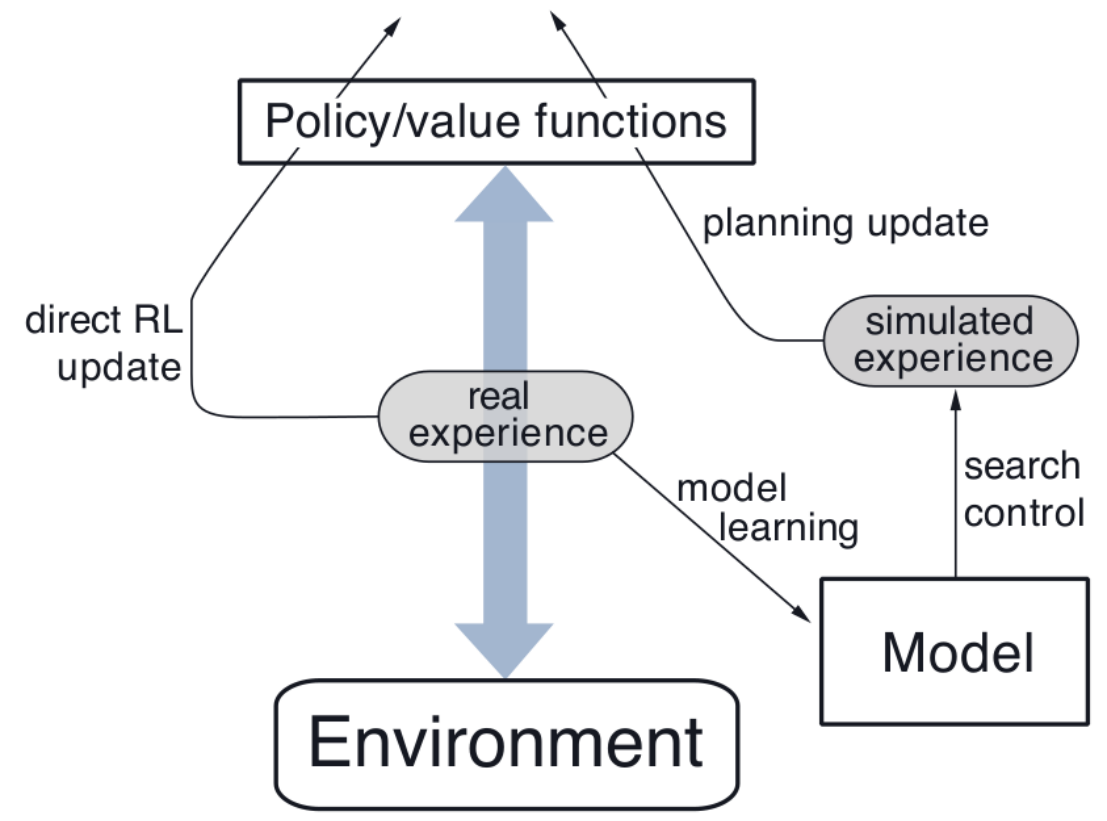
  и их оптимальные варианты V_*, Q_*;

Планирование

- Мы выписывали уравнения Беллмана на V и Q и научились их решать.
- Кроме того, методами Монте-Карло мы умеем обучать модель окружающей среды.
- А TD-обучением мы можем обучать одновременно и модель, и оптимальную стратегию.
- *Планирование* (planning) — это то, как использовать модель, чтобы лучше обучать V и Q.
- Как нам это сделать?..

Планирование

- Базовая идея: давайте сэмплировать новый опыт из модели окружающей среды, использовать симуляции.
- Пример – архитектура Dyna:



Планирование

- В простейшем случае можно делать ещё несколько обновлений, скажем, Q-обучения, исходя из модели.

Алгоритм Dyna-Q:

- инициализировать случайно π, $Q(s,a)$, $M(s,a)$ (модель);
- повторять (цикл внутри эпизода, эпизоды тоже повторять):
  - для состояния S выбрать A по ϵ-жадной стратегии из Q;
  - сделать действие A, пронаблюдать R и S';
  - $Q(S,A) := Q(S,A) + \alpha\left(R + \gamma \max_a Q(S',a) - Q(S,A)\right)$;
  - добавить R, S' в $M(S,A)$;
  - повторить k раз:
    - (S,A) — случайная пара, которую уже наблюдали раньше (т.е. для неё есть $M(S,A)$);
    - породить R, S' по $M(S,A)$;
    - $Q(S,A) := Q(S,A) + \alpha\left(R + \gamma \max_a Q(S',a) - Q(S,A)\right)$.
- Dyna-Q случайно обновляет k значений $Q(s,a)$ по имеющейся модели; k можно выбирать из соображений имеющихся ресурсов.

Планирование

- Конечно, ещё лучше будет обновлять не случайно.

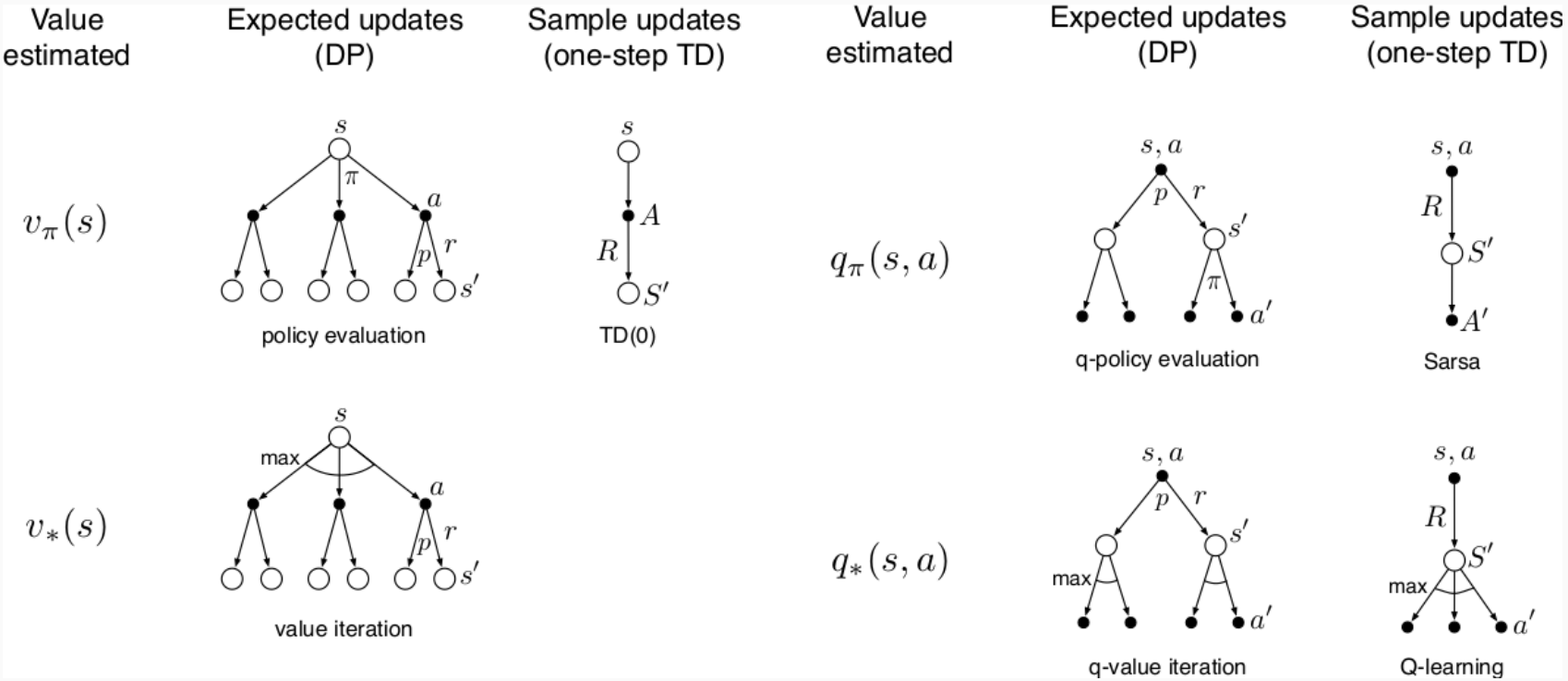
Алгоритм обхода по приоритетам (prioritized sweeping):

- инициализировать π, $Q(s,a)$, $M(s,a)$ (модель), PQ (очередь);
- повторять (цикл внутри эпизода, эпизоды тоже повторять):
  - для состояния S выбрать A по ϵ-жадной стратегии из Q;
  - сделать A, пронаблюдать R и S', добавить R, S' в $M(S,A)$;
  - $P := |R + \gamma \max_a Q(S',a) - Q(S,A)|$; если $P > \theta$ (порог), добавить (S,A) в PQ с приоритетом P;
  - повторить k раз, пока $PQ \neq \emptyset$:
    - взять $(S,A) := \mathrm{Head}(PQ)$, породить R, S' по $M(S,A)$;
    - $Q(S,A) := Q(S,A) + \alpha\left(R + \gamma \max_a Q(S',a) - Q(S,A)\right)$.
    - для каждой пары $(\tilde{S}, \tilde{A})$, из которой модель попадает в S:
      * $\tilde{R}$ – предсказанная моделью награда для $(\tilde{S}, \tilde{A}, S)$;
      * $P := |\tilde{R} + \gamma \max_a Q(S,a) - Q(\tilde{S}, \tilde{A})|$ (приоритет);
      * если $P > \theta$, то добавить $(\tilde{S}, \tilde{A})$ в PQ с приоритетом P.

- Это для детерминированного окружения, для случайного можно взвесить оценками вероятностей попасть в S.

Expected vs. sample updates

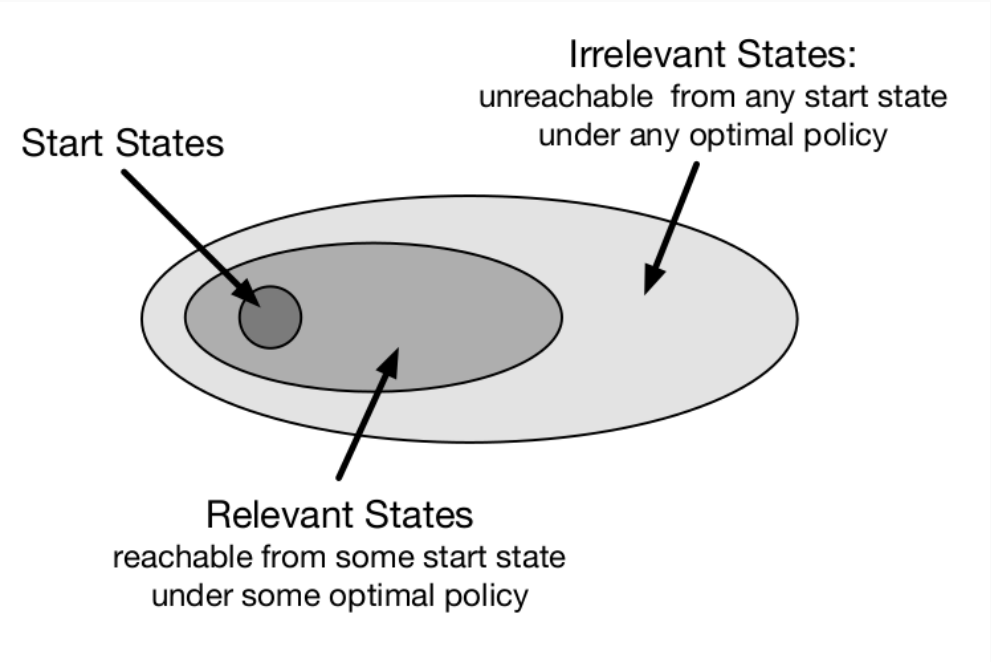
- Вот ещё один взгляд на то, чем мы занимались – expected updates vs. sample updates:



- Expected update проходит по всем следующим состояниям.
- Sample update выбирает одно (случайно или из опыта).
- Expected, конечно, лучше, но и дороже, в целом sample даже выгоднее может получиться.

Expected vs. sample updates

- Как лучше распределить апдейты (т.е. фактически имеющийся вычислительный ресурс)?
- Trajectory sampling: лучше сэмплировать сразу траекториями по реальной стратегии. Так мы не будем тратить время на недостижимые или очень-редко-достижимые состояния.
- Например, RTDP (real-time dynamic programming): обновляем по уравнениям Беллмана, но только состояния, которые реально встречались в траекториях.

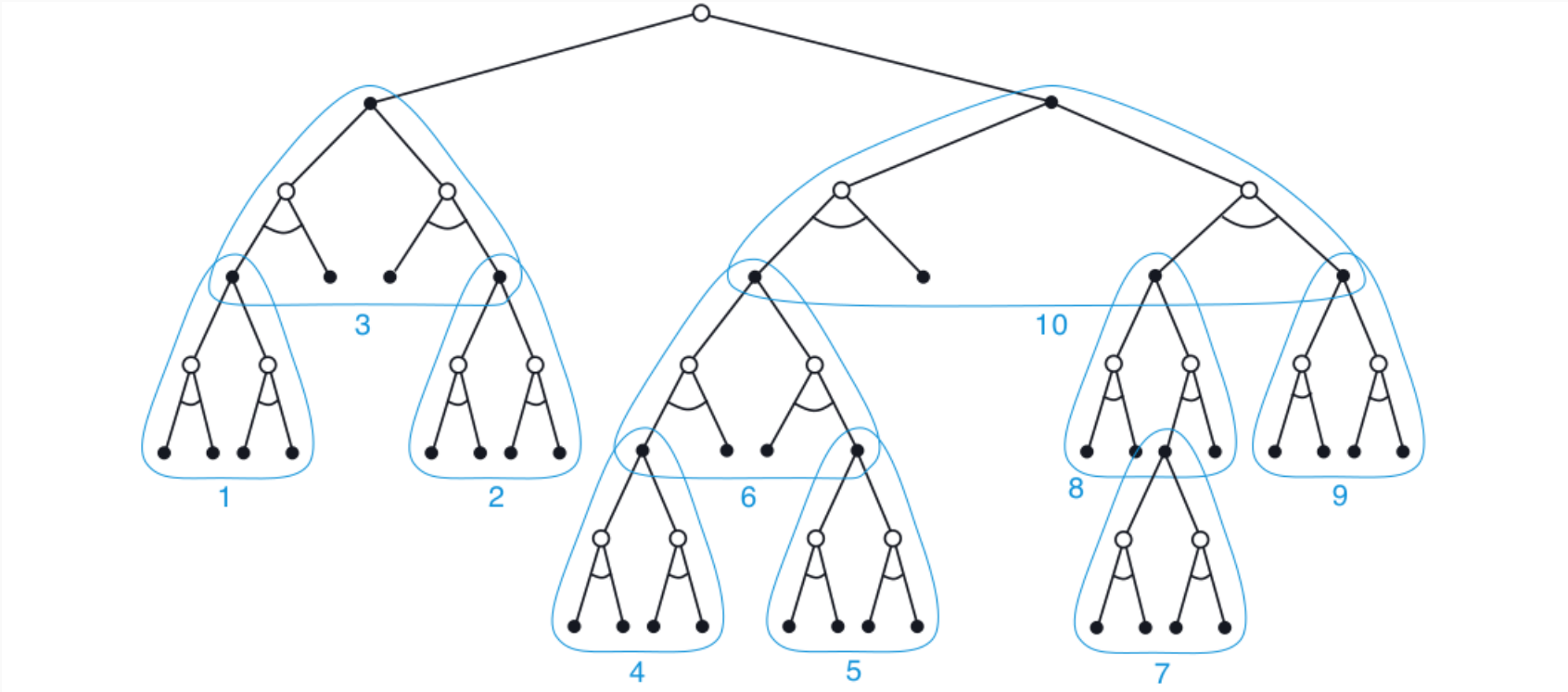


Планирование в текущем моменте

- А можно использовать планирование прямо при выборе действия, т.е. фактически как часть стратегии (decision-time planning).
- Простейшее такое планирование мы знаем: когда мы обучали $V(s)$, а не $Q(s, a)$, потом для получения стратегии π по функции V нужно было перебрать возможные следующие состояния и выбрать лучшее.
- Как это обобщить и улучшить?..

Планирование в текущем моменте

- Правильно, это поиск на несколько «ходов» в глубину! Точнее говоря, поиск получается в ширину. :)



- Heuristic search: делаем поиск в ширину на несколько ходов вперёд, выбирая наилучшее действие (кстати, почему это называется «минимакс», если тут только $\max$?)

Планирование в текущем моменте

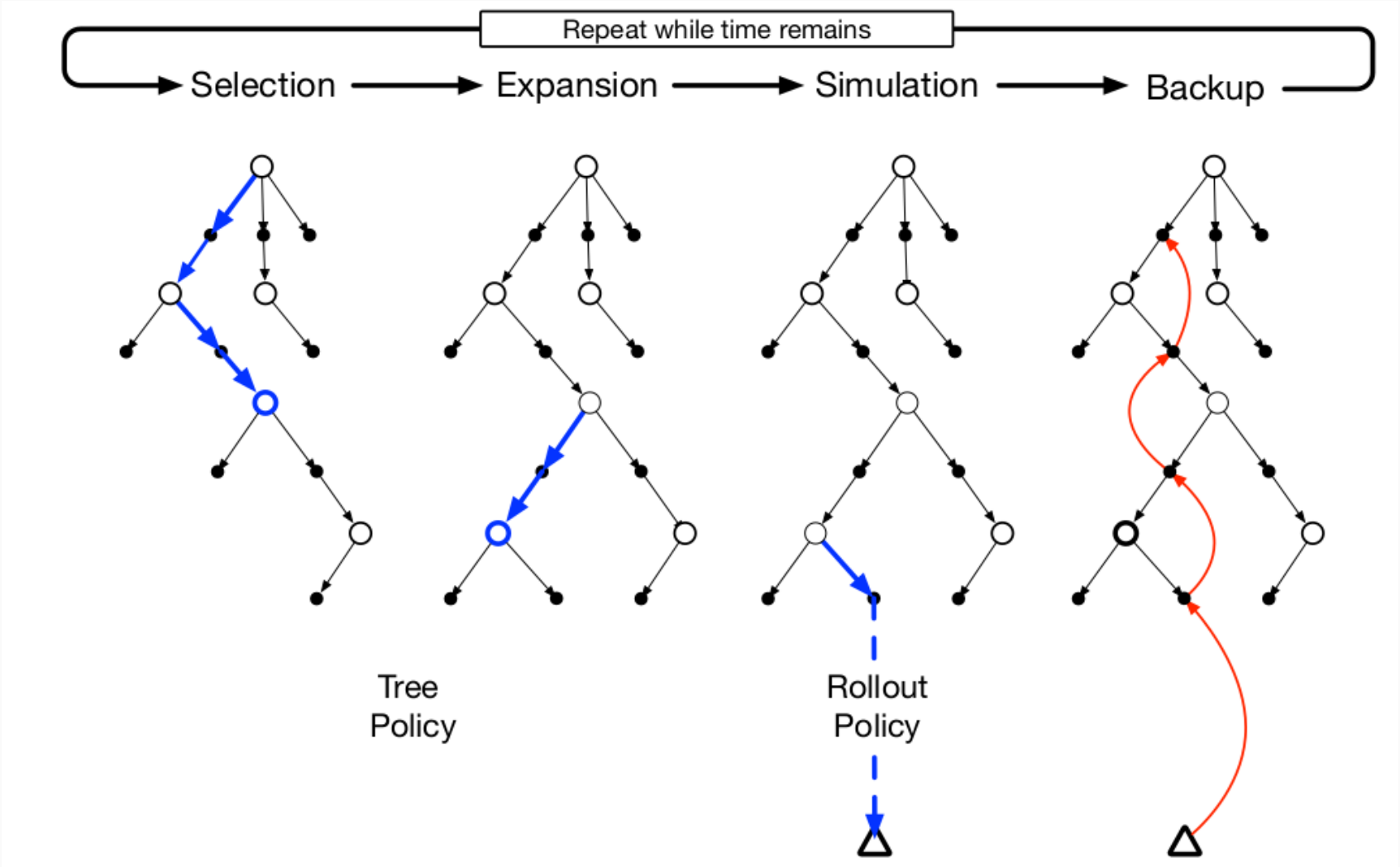
- Другой способ реализовать такое планирование: rollouts.
- Начиная с текущего состояния, симулируем несколько траекторий по текущей стратегии, а потом оцениваем действия, усредняя результаты этих симуляций.
- Когда оценки становятся достаточно точными, можно сделать ход, а потом опять строить rollouts из следующего состояния.
- Это пробовал ещё Тезауро для нард, и получалось очень хорошо, прямо неожиданно хорошо.
- Но ещё лучше совместить одно и другое и получить...

Планирование в текущем моменте

- ...Monte Carlo tree search (MCTS)! Это один из ключевых компонентов, например, в прогрессе алгоритмов для го между 2005 (слабый любитель) до 2015 (6 дан), а потом к AlphaGo, а потом и к AlphaZero — там везде есть MCTS.
- Мы строим дерево, в котором оцениваем листья через rollouts и выбираем, когда раскрыть лист дальше, т.е. итеративно:
  - выбираем лист дерева, действие в нём и следующее за ним состояние;
  - раскрываем всевозможные действия в этом состоянии;
  - делаем rollouts исходя из этих действий, используя их результаты для обновления оценок действий на пути к корню дерева.

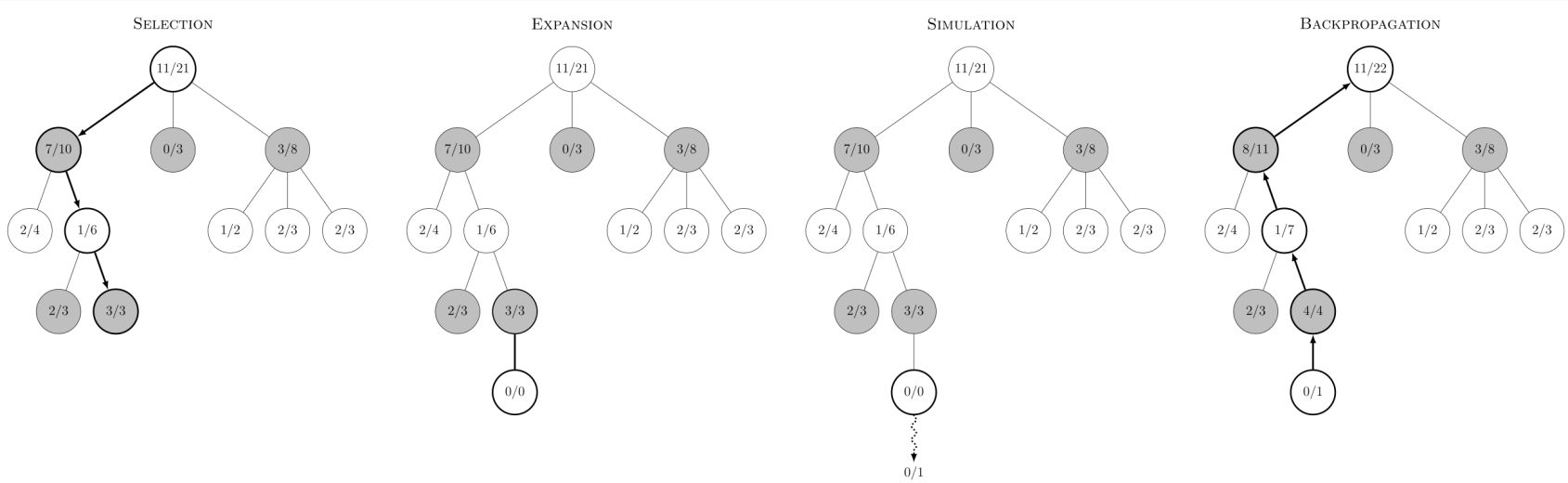
Планирование в текущем моменте

- Вот такая картинка получается:



Планирование в текущем моменте

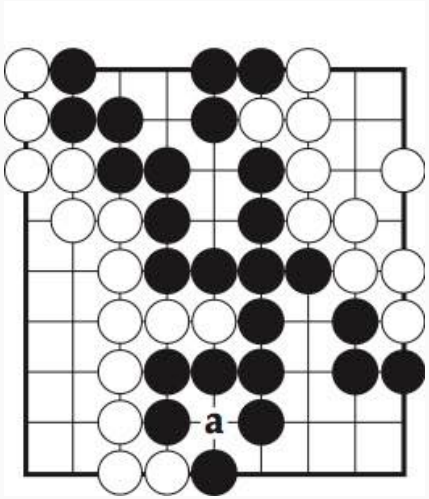
- Выбирать листья и действия нужно по статистике побед/поражений:



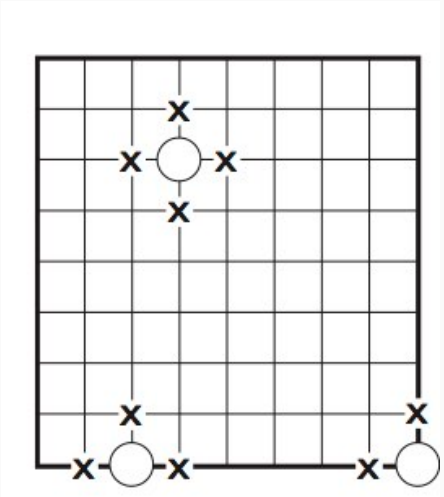
- Например, по методу UCT (upper confidence bounds applied to trees): выбираем для rollout действие, максимизирующее

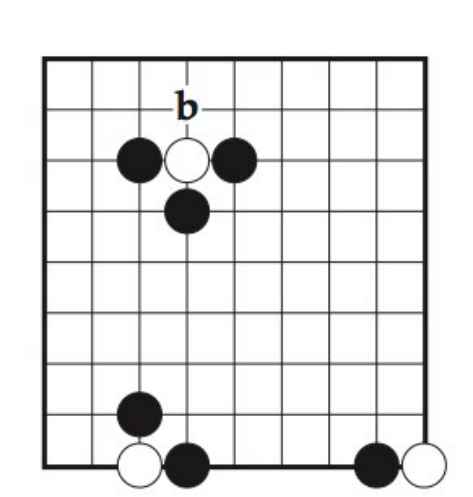
$$\frac{w_i}{n_i} + c\sqrt{\frac{\ln N_i}{n_i}}.$$

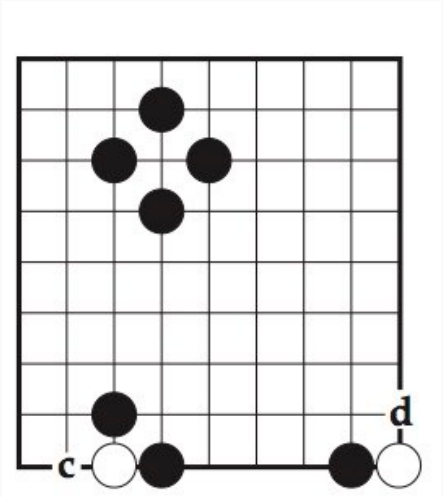
Правила го



- Вкратце о правилах: камни ставят на доску 19×19 (9×9, 13×13).
- Камни снимаются, когда их окружают.
- Побеждает тот, кто занимает больше территории (плюс съеденные камни).





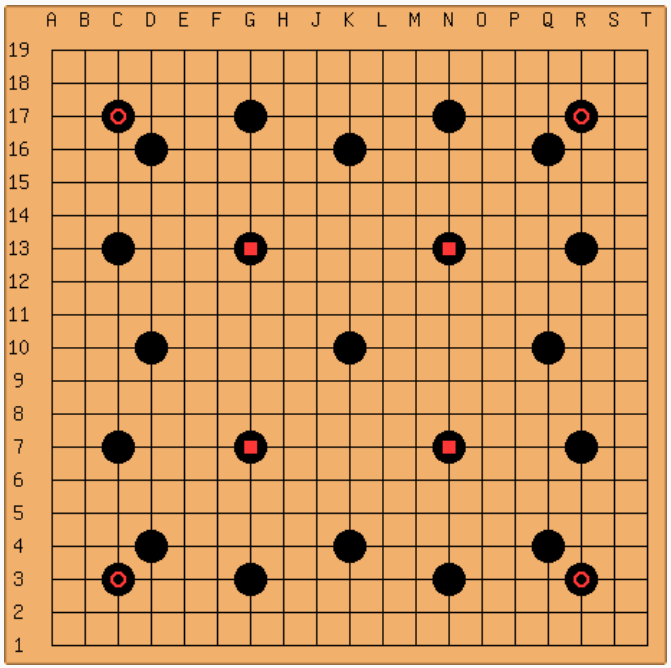


История компьютерного го

- 1968: Albert Zobrist – первая программа, влияние камней, Zobrist hashing.
- Середина 1980х: появились PC, интерес к этому.
- Ing Prize: 40M тайваньских долларов, если компьютер сможет до 2000 года обыграть профессионала 1 дана.
- Программы 1990х: Goliath (1989-1991), Go Intellect, Handtalk, Goemate.
- (David Fotland, 1993): Many Faces of Go, долгое время одна из ведущих программ.
- Они работали на pattern matching и базах знаний.
- И насколько же хорошо они играли?..

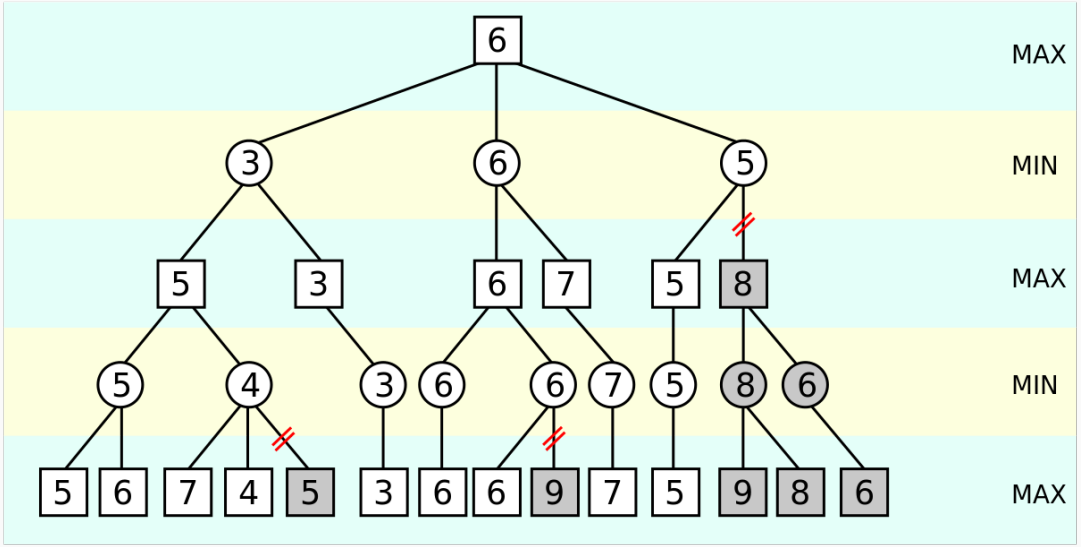
О силе игры

- ...да совсем никак.
- 1997: Janice Kim 1p победила HandTalk *на 25 камнях форы*.
- В том же году HandTalk выиграл часть Ing Prize, победив трёх 11-13-летних любителей 2-6d на 11 камнях форы.
- 2001: Many Faces выиграл у insei (1kyu pro) *на 15 камнях*.
- Это те годы, когда DeepBlue уже давно победил Каспарова.
- В чём же дело? Почему так сложно?



Почему го -- это сложно

- В других играх (шахматы) отлично работает alpha-beta search: строим минимакс-дерево ходов, выкидываем ходы, которые заведомо хуже других, ищем в глубину.



- Почему не в го?

Почему го -- это сложно

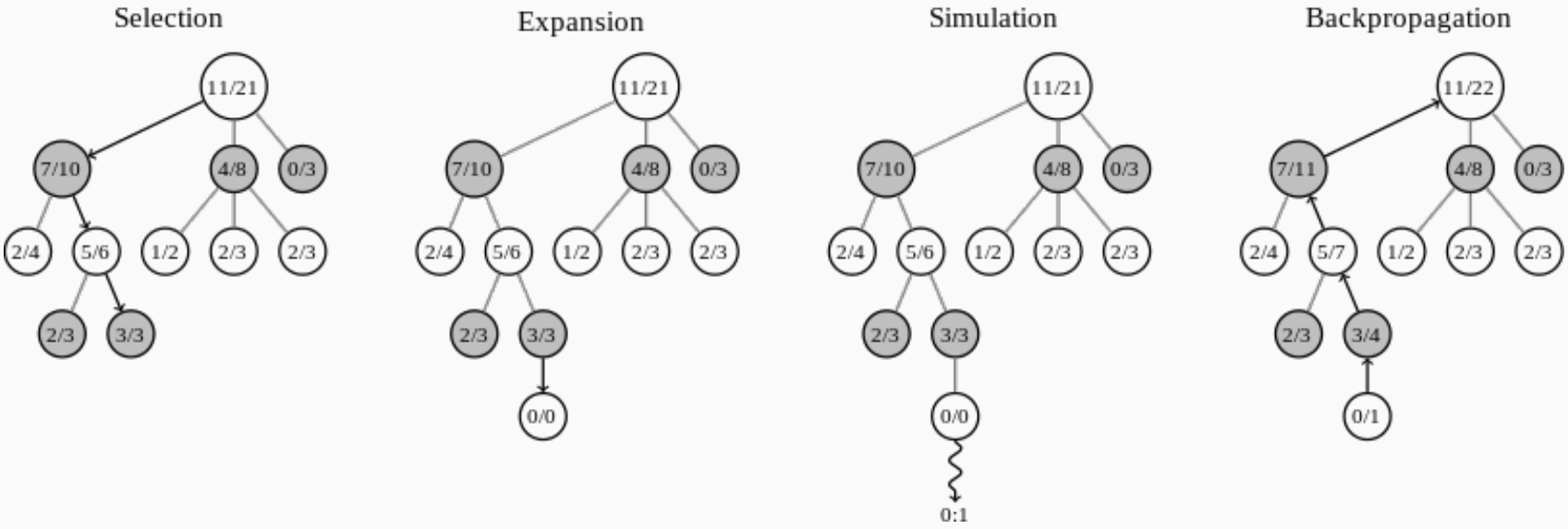
- Почему го сложнее, чем шахматы?
- Очень большая ширина дерева (branching factor):
  - в шахматах 30-40 возможных ходов, из них многие сразу приводят к потерям и их можно обрезать;
  - в го около 250 возможных ходов, и из них «вполне разумных» тоже около сотни.
- Оценка позиции:
  - в шахматах можно посчитать материал и какие-то простые эвристики, и если разрыв большой, скорее всего оценка очевидна;
  - в го тоже может упасть большая группа, но это редкость, обычно потери территориальные, труднооцениваемые;
  - очень сложная для автоматизации задача – даже просто оценить конечную позицию в го (когда люди уже согласились, кто выиграл).

Почему го -- это сложно

- Тактика – life and death problems:
  - казалось бы, компьютеры считают варианты быстро;
  - но в го на решение локальных проблем влияют далеко расположенные камни и вся позиция; редко можно обрезать часть доски и сделать полный перебор;
  - а статус групп может измениться под влиянием глобальных эффектов и взаимодействовать с ними;
  - кроме того, человеку в го считать проще, чем в шахматах.
- Стратегия: нужно понимать взаимодействие камней на всей доске и очень тонко оценивать возникающую позицию, досчитать большинство веток до «явного преимущества», как в шахматах, невозможно.
- Ко-борьба: программы всегда отвратительно играли ко, бездарно тратили ко-угрозы, не замечали, что ко может возникнуть.

Monte-Carlo tree search

- 2006 (Remi Coulom, затем MoGo, затем все): революция компьютерного го – Monte-Carlo tree search (MCTS).
- Запускаем *случайные симуляции* с текущей позиции, смотрим, в каких ветках больше выигрышей, повторяем.



Monte-Carlo tree search

- Основная часть MCTS – формула, по которой выбирают узел для дальнейшего анализа:
  - это всё основано на тех же многоруких бандитах;
  - UCT (upper confidence bound UCB1 applied to trees) – выбираем узел с максимальным

$$\frac{w_i}{n_i} + c\sqrt{\frac{\ln t}{n_i}},$$

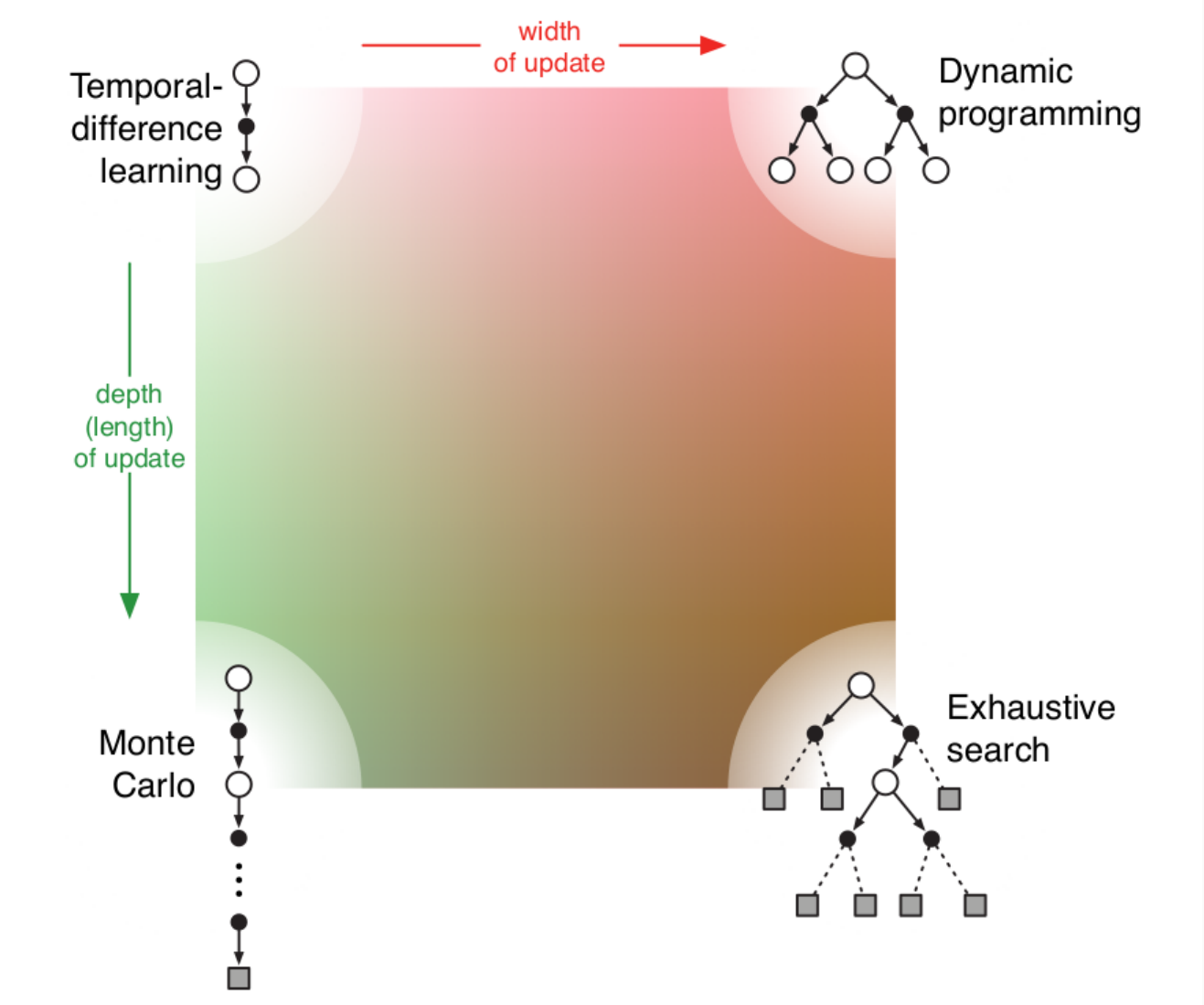
где w_i – число побед, n_i – число симуляций, c – параметр (часто $\sqrt{2}$), $t = \sum_i n_i$ – общее число симуляций.

Monte-Carlo tree search

- И пошли более разумные результаты:
  - 2006: CrazyStone выиграл ICGA Computer Olympiad на доске 9×9;
  - 2007: CadiaPlayer – чемпион по General Game Playing;
  - 2008: MoGo достиг уровня дана на доске 9×9;
  - 2009: Fuego побеждает одного из топ-игроков на доске 9×9;
  - 2009: Zen достигает уровня 1d на сервере KGS на доске 19×19;
  - 2012: Zen выиграл 3:1 матч у 2d любителя на доске 19×19;
  - 2013: CrazyStone выиграл у Yoshio Ishida 9p на четырёх камнях;
  - около 2015-2016 CrazyStone и Zen, основанные на MCTS, достигали уровня 6-7d на KGS (но это совсем не то же, что профессиональные даны).
- MCTS работал и в других играх (Hex, Arimaa, MtG, Settlers).

Наши методы

- И ещё один взгляд-иллюстрация:



Спасибо!

Спасибо за внимание!