Планирование в задачах обучения с подкреплением

Модели

- Модель – то, что может использоваться агентом для прогнозирования реакции среды на действия.

- Модель распределений

$$P^a_{ss'} \quad R^a_{ss'}$$

- Модель образцов взаимодействия

Планирование

- Планирование – вычислительный процесс, получающий на вход модель и получающий (улучшающий) стратегию

- Планирование в пространстве состояний
- Планирование в пространстве планов

Планирование и обучение с подкреплением

- Методы планирования в пространстве состояний вычисляют функции ценности как основной шаг для улучшения стратегии.
- Они вычисляют функции ценности путем выполнения обновлений на основе опыта взаимодействия с моделью

Одношаговое Q-планирование со случайным выбором

Инициализация: Q(s,a) – произвольно

Повторять бесконечно:

Случайно выбрать состояние $s \in S$ и действие $a \in A(s)$

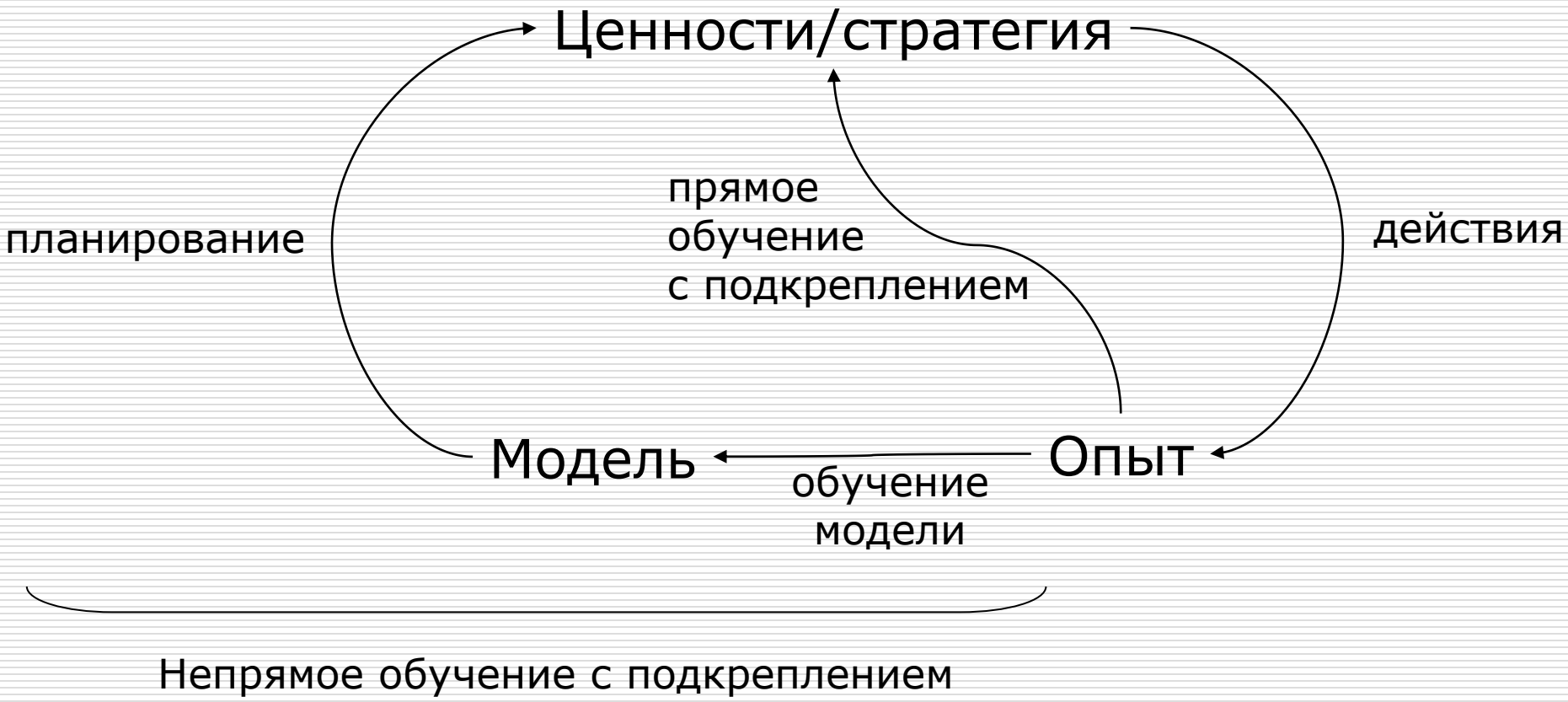
Передать s и a модели, получить следующее состояние s’ и подкрепление r.

Выполнить одношаговое Q-обучение

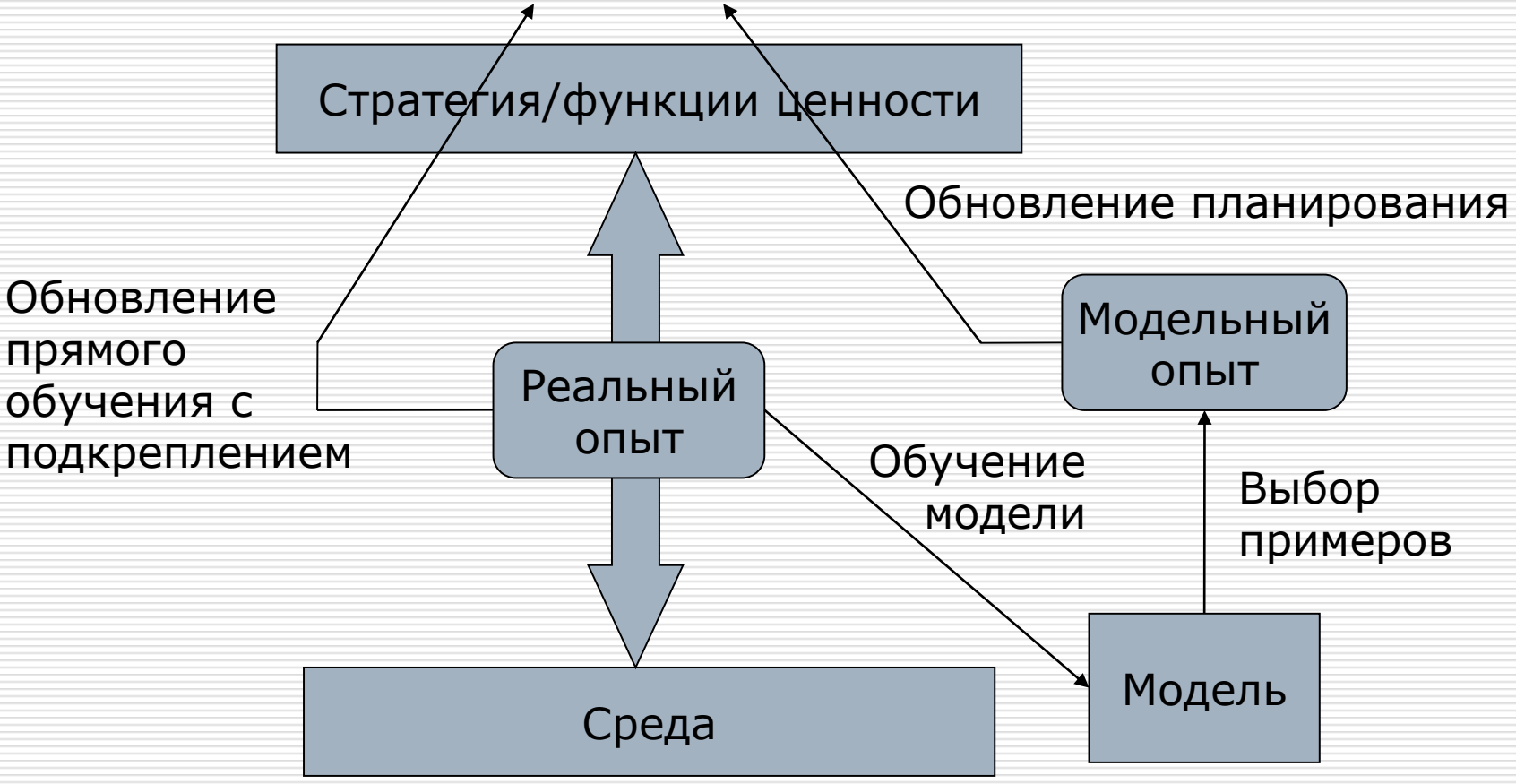
$Q(s,a) \leftarrow Q(s,a)+\alpha(r+\gamma \max_{a'} Q(s',a')-Q(s,a))$

- Сходится при условии выбора всех пар (s,a) и уменьшения α.

Планирование, обучение и действие



Архитектура Dyna-Q



Алгоритм Dyna-Q

- Алгоритм планирования:
  - Одношаговое Q-планирование со случайным выбором
- Алгоритм обучения:
  - Одношаговое Q-обучение
- Обучение модели:
  - Детерминированный случай
  - Для всех посещённых пар (s,a) запоминаем (s’,r)
- Алгоритм выбора:
  - Случайно выбираем одну из ранее посещённых пар (s,a)

Алгоритм Dyna-Q

Инициализация:

Q(s,a) – произвольно

Model(s,a) – пусто

Повторять вечно

$s \leftarrow$ начальное состояние

$a \leftarrow \varepsilon$-жадное(Q,s)

Выполнить a, получить s’ и r.

$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a))$

$Model(s,a) \leftarrow (s',r)$

Повторять N раз

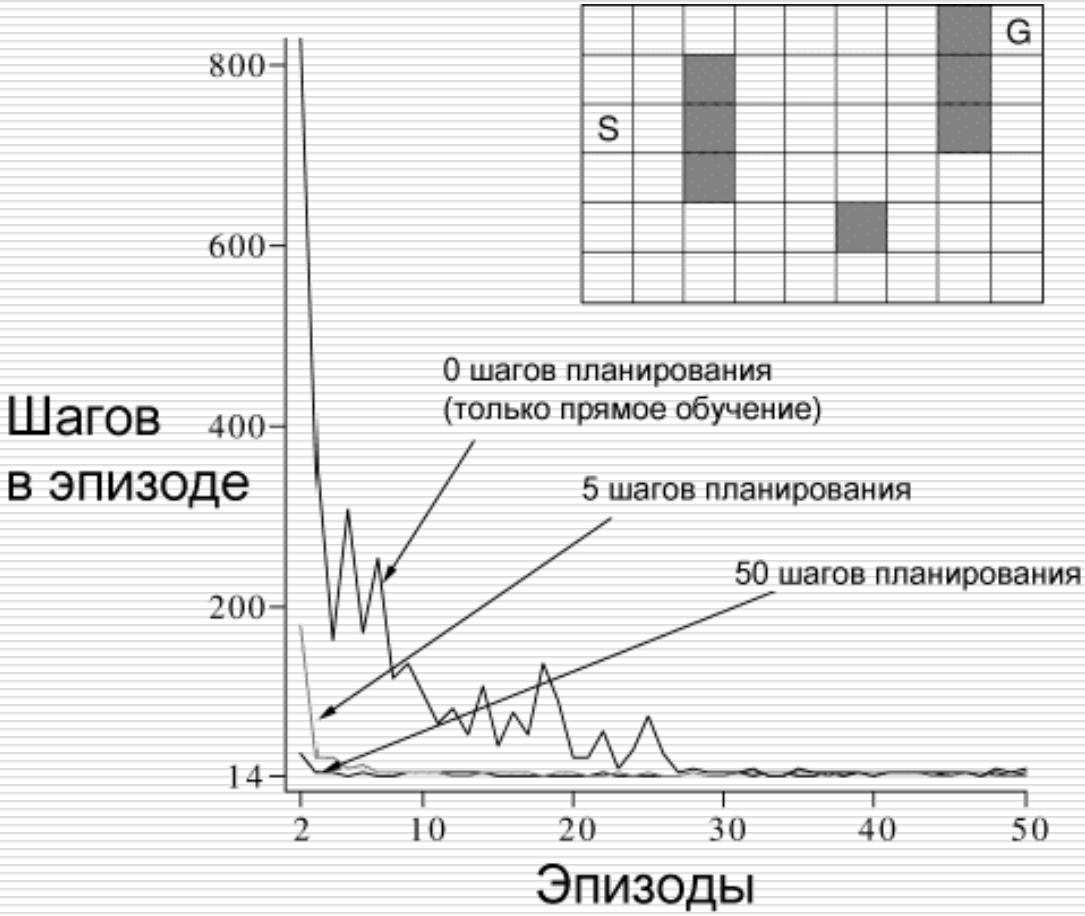
$s \leftarrow$ случайное ранее посещённое состояние

$a \leftarrow$ случайное действие, выполненное в s.

$(s',r) \leftarrow Model(s,a)$

$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a))$

Алгоритм Dyna-Q. Пример.



α=0.1
ε=0.1

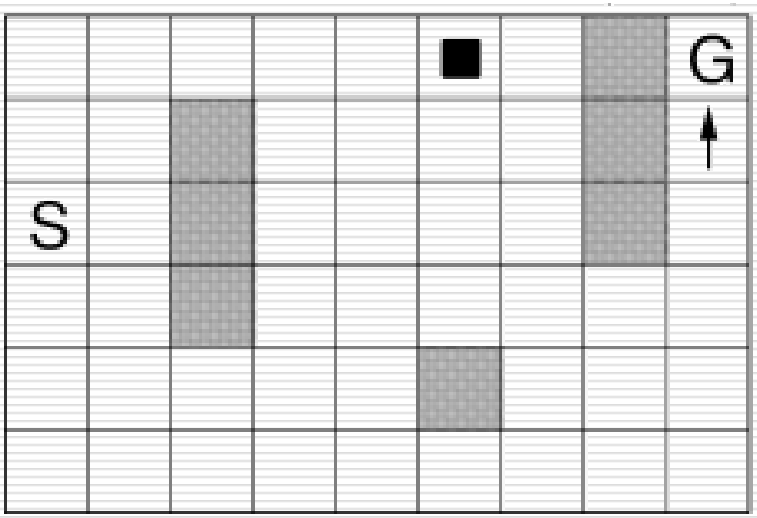
ε-оптимальная стратегия найдена за:

- 25 эпизодов при N=0
- 5 эпизодов при N=5
- 3 эпизода при N=50

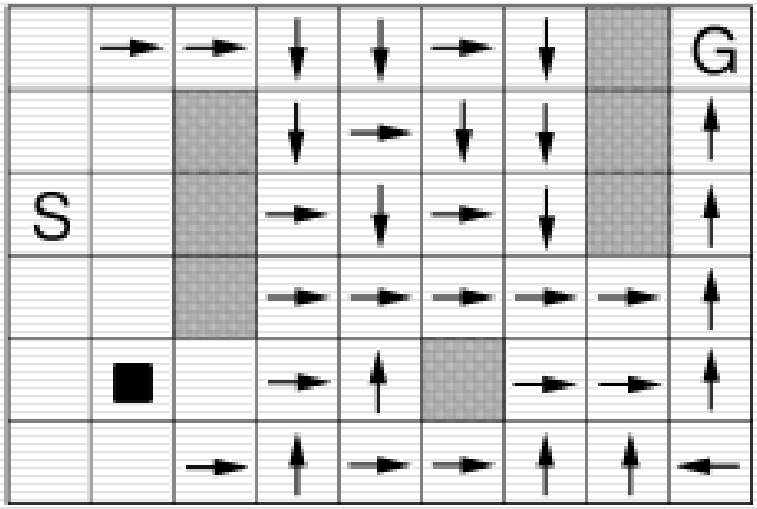
Алгоритм Dyna-Q. Пример.

- Стратегии агента во время второго эпизода

Без планирования (N=0)



С планированием (N=50)

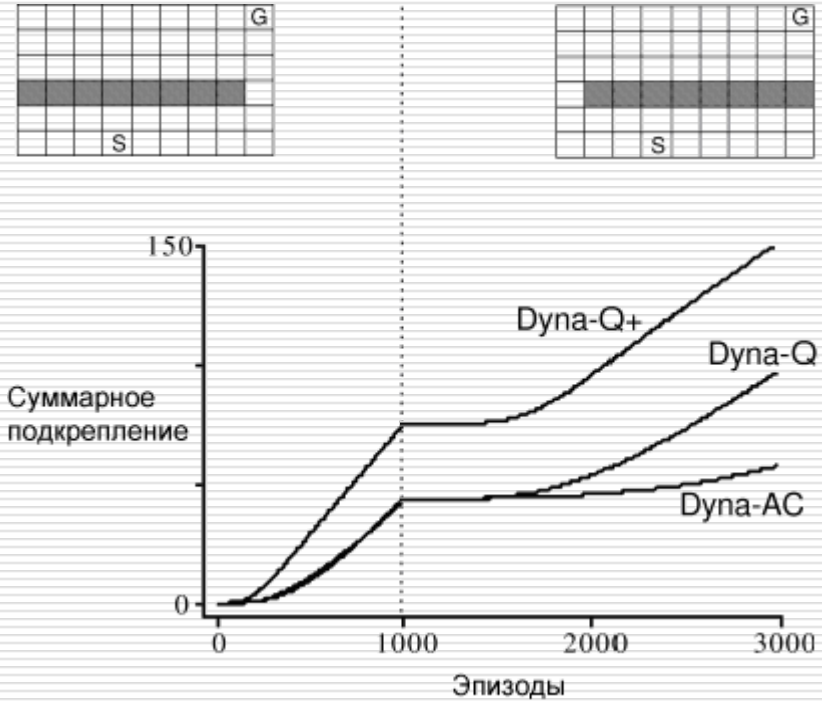


Причины неточностей модели

- Среда является стохастической, и мы имеем лишь ограниченный опыт взаимодействия
- Использовалась аппроксимация
- Изменения среды

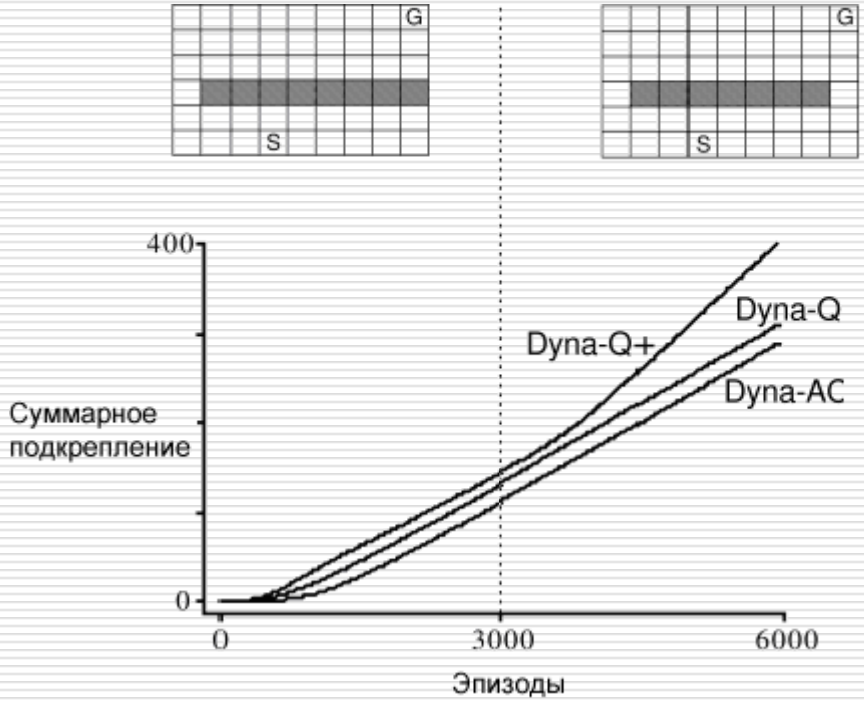
Неточности модели

- Ухудшение среды



Неточности модели

- Улучшение среды



Агент Dyna-Q+

- Запоминаем время последнего посещения для каждой пары (s,a)

- При использовании в планировании пары (s,a), посещённой n шагов назад с подкреплением r, используем подкрепление

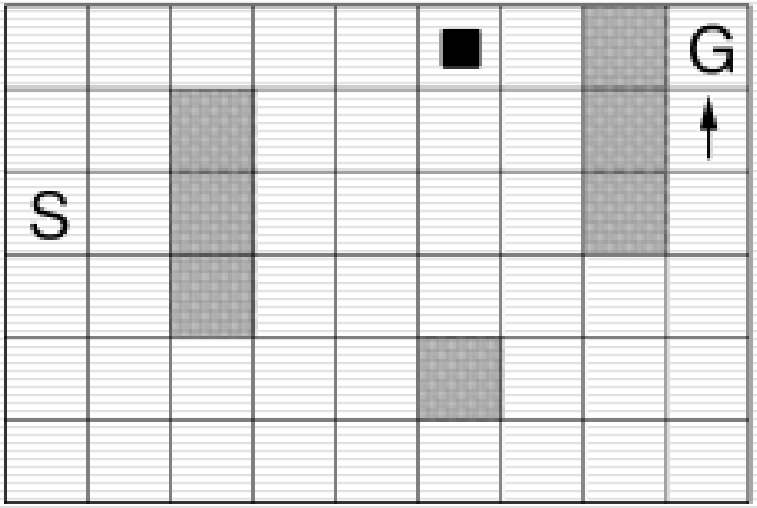
$$r + \kappa\sqrt{n}$$

где k – положительная константа.

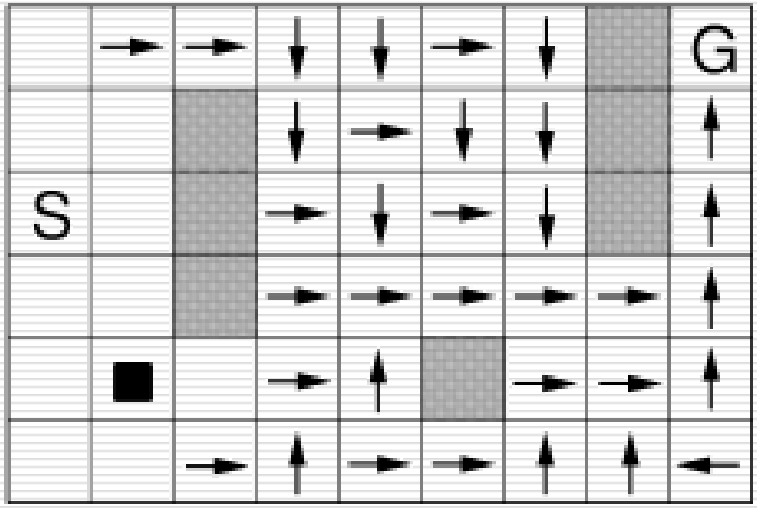
Приоритетная выборка примеров

- Вспомним пример с лабиринтом

Без планирования (N=0)



С планированием (N=50)



Приоритетная выборка примеров

- Содержательные обновления получаются только при рассмотрении переходов, которые приходят в состояния, значение функции ценности которых было изменено.
- Чем сильнее изменение функции ценности для состояния, тем большие изменения функции ценности оно может вызвать для других состояний.
- Можно организовать очередь изменившихся состояний с приоритетами, зависящими от абсолютного значения изменения функции ценности.

Алгоритм Dyna-Q с приоритетной выборкой

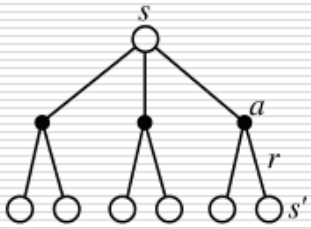
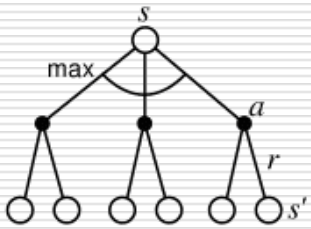
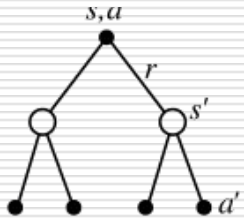
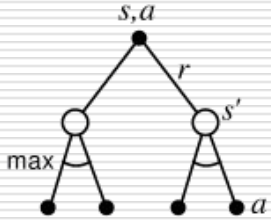
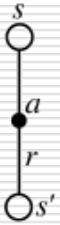
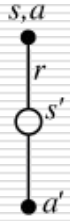
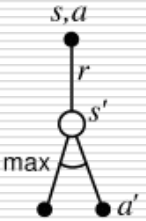
Инициализация:

Q(s,a) – произвольно
Model(s,a) – пусто
PQueue - пусто

Повторять вечно

s ← начальное состояние
a ← стратегия(Q,s)
Выполнить a, получить s’ и r.
Model(s,a) ← (s’,r)
p ← $|r+\gamma \max_{a'} Q(s',a')-Q(s,a)|$
Если p>0 вставить (s,a) в PQueue с приоритетом p
Повторять N раз и пока PQueue не пуста
(s,a) ← наиболее приоритетная из PQueue
(s’,r) ← Model(s,a)
Q(s,a) ← $Q(s,a)+ \alpha(r+\gamma \max_{a'} Q(s',a')-Q(s,a))$
Повторять для всех s’,a’ что Model(s’,a’)=(s,r)
p ← $|r+\gamma \max_{a'} Q(s',a')-Q(s,a)|$
Если p>0 вставить (s’,a’) в PQueue с приоритетом p

Одношаговые обновления

Оцениваемая функция	$V^{\pi}(s)$	$V^{*}(s)$	$Q^{\pi}(a,s)$	$Q^{*}(a,s)$
Полные обновления (DP)	 Оценка стратегии	 Итерация ценности	 Q-оценка стратегии	 Q-итерация ценности
Обновления по образцу (одношаговый TD)	 TD(0)		 Sarsa	 Q-learning

Полные обновления и обновления по образцу

- Рассмотрим вычисление Q^* при условиях:
  - Дискретных множеств S и A(s)
  - Табличное представление функций
  - Модель распределений: $\hat{P}^a_{ss'}$ $\hat{R}^a_{ss'}$

- Полное обновление

$$Q(s,a) \leftarrow \sum_{s'} \hat{P}^a_{ss'} \left[\hat{R}^a_{ss'} + \gamma \max_{a'} Q(s',a') \right].$$

- Обновление по образцу

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[\hat{R}^a_{ss'} + \gamma \max_{a'} Q(s',a') - Q(s,a) \right],$$

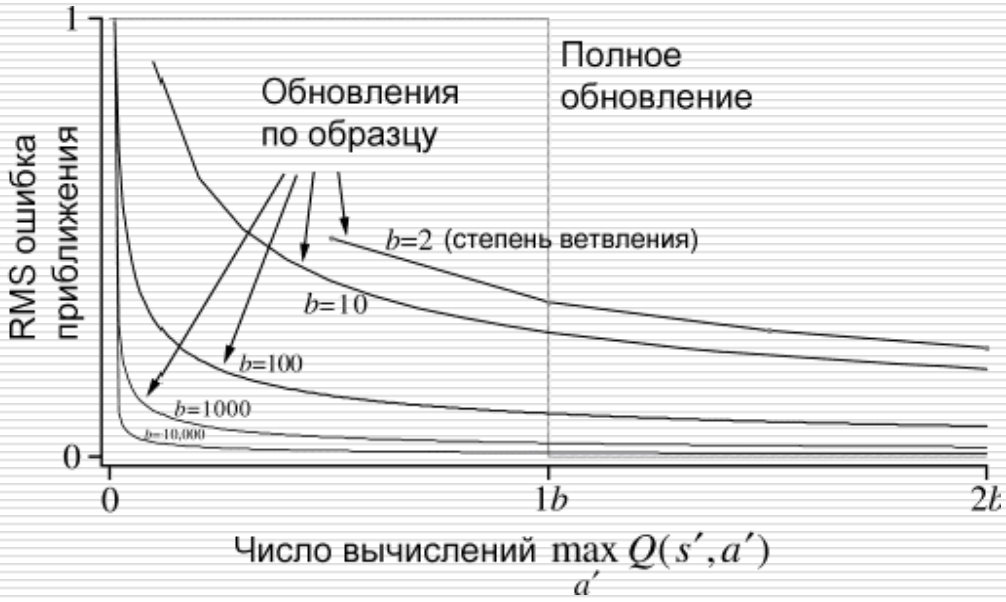
Полные обновления и обновления по образцу

- Если возможно только одно следующее состояние, то обновления идентичны
- Полное обновление
  - Точность вычисления полного обновления зависит только от точности значений функции Q(s’,a’).
- Обновление по образцу
  - Точность зависит от точности Q(s’,a’) и использованного образца
  - Требует меньше вычислительных ресурсов

Полные обновления и обновления по образцу

- Степень ветвления – число последующих состояний, для которых $\hat{P}^a_{ss'} > 0$.
- Пусть имеется b равновероятных последующих состояний и ошибка в начальном состоянии =1.
- Обновления по образцу уменьшают ошибку пропорционально

$$\frac{1}{\sqrt{t}}\,\frac{b-1}{b}$$



Распределение обновлений

- Полный проход по пространству состояний
  - Для больших задач может быть недостаточно времени даже для одного прохода
  - Не позволяет акцентировать внимание на отдельных регионах
- Выборка согласно некоторому распределению
  - Равномерное распределение
    - Имеет те же проблемы, что и полный проход
  - Распределение соответствующее текущей стратегии
    - Легко генерировать
    - Акцентирует внимание на нужных агенту состояниях
    - Гарантирована сходимость при использовании линейной аппроксимации

Распределение обновлений согласно траектории

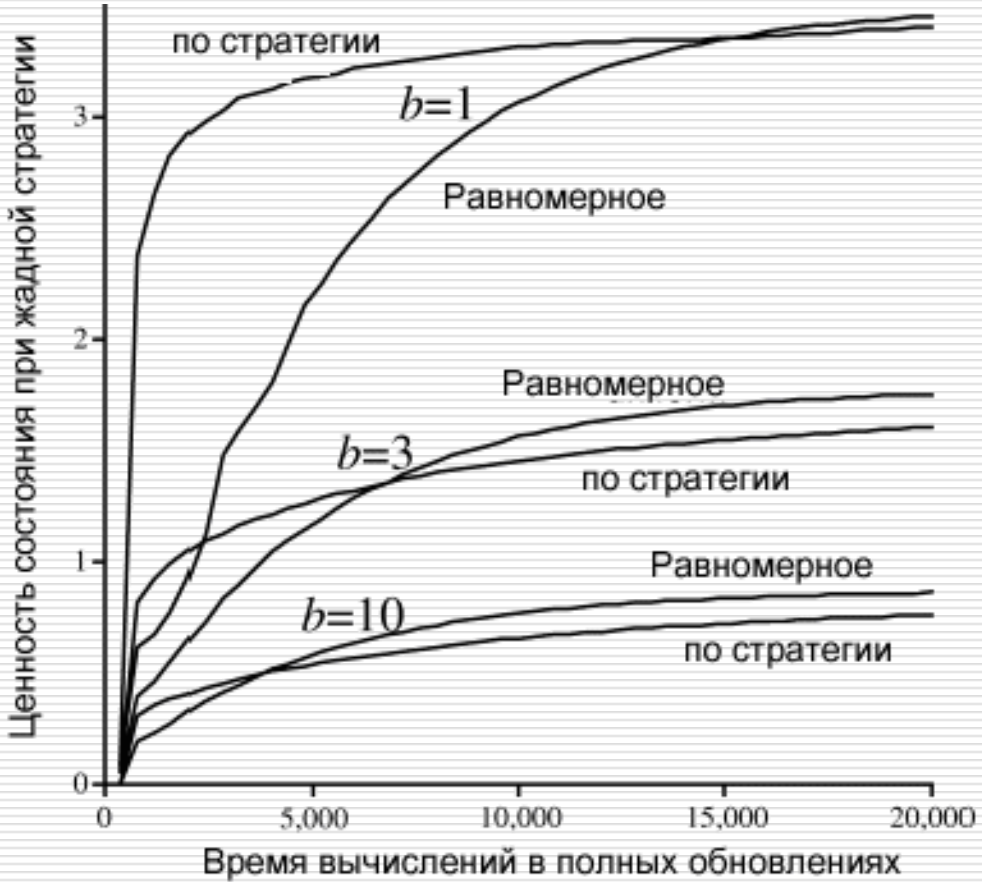
- Почему хорошо использовать?
  - Фокусирует внимание агента на полезных состояниях, игнорируя (возможно большие) ненужные области.
- Почему плохо использовать?
  - Повторяет вычисления по старым областям, в которых всё известно.

Распределение обновлений согласно траектории

- Равномерные обновления: проход по всем парам (s,a)
- Обновления по траектории: моделировались эпизоды, обновления выполнялись согласно выбору действий ε-жадной стратегией.

- Задача:
  - Из |S|=1000 состояний возможны два действия, каждое из которых равновероятно приводит в одно из b последующих состояний (разные для разных начальных состояний).
  - С вероятностью 0.1 происходит переход в терминальное состояние.
  - В любой момент можно вычислить V^{π} и сравнить поведение агента с оптимальным.
  - 200 случайных задач

Распределение обновлений согласно траектории



- Распределение по траектории обеспечивает более эффективное планирование в начале эпизода, но со временем уступает равномерному

- Начальное преимущество является более сильным и продолжительным при:
  - небольших факторах ветвления
  - увеличении числа состояний

Эвристический поиск

- Строиться дерево возможных продолжений развития ситуации.
- На листьях дерева оценивается приблизительное значение функции ценности.
- Вычисляется ценность состояний выше по дереву, таким же методом, каким мы вычисляли обновления для V^*/Q^*.
- Коррекции функции ценности не сохраняются, во многих случаях функция ценности определяется человеком при создании системы.

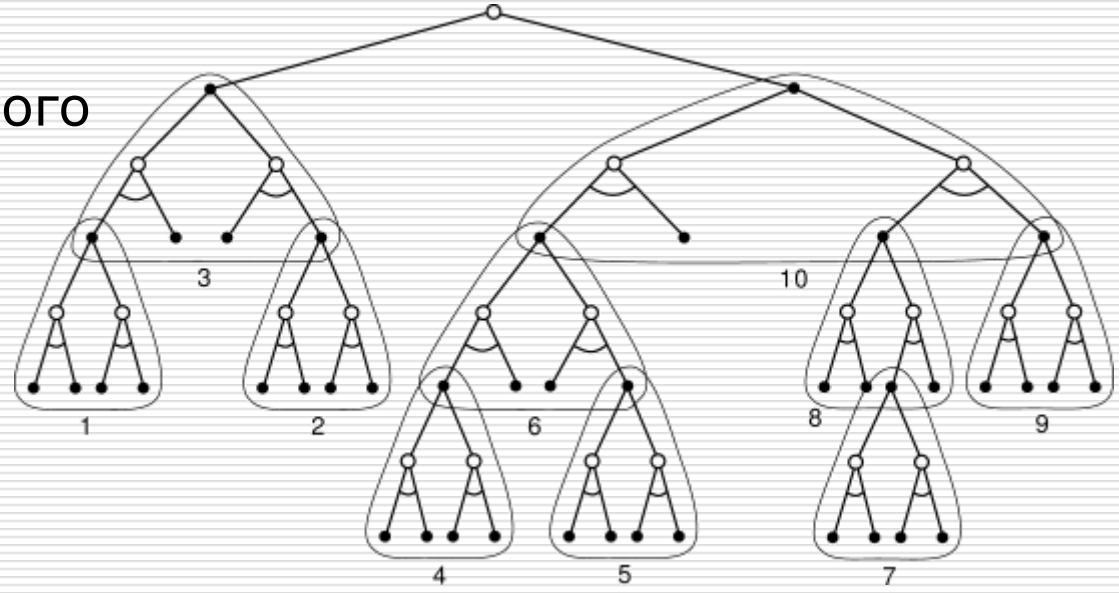
Эвристический поиск

- Если имеется точная модель и неточная функция ценности, то эвристический спуск, как правило, позволяет получить более точный результат
  - Если доходим до конца эпизода, то получаем точное значение
  - Если спускаемся на достаточную глубину *k*, для которой γ^k мало, то ошибка будет соответственно мала.
  - Более глубокий спуск требует больше вычислительных ресурсов.
  - Во многих случаях спуск производится не равномерно:
    - Глубже для действий, которые представляются более эффективными
    - Мельче для неэффективных действий

Эвристический поиск и обновления

- Начинаем с текущего состояния.
- Строим дерево эвристического поиска.
- Выполняем одношаговые обновления в направлении снизу-вверх по дереву.

- В случае табличного представления получаем результат, совпадающий с эвристическим поиском.



Выводы

- Обучение и планирование можно совместить:
  - Оба процесса меняют одну функцию ценности;
  - Применяются одинаковые алгоритмы, различаются лишь источники опыта.

- Методы инкрементального планирования можно совместить с действием агента в реальной среде и построением модели:
  - Все три процесса могут осуществляться одновременно, при этом каждый производит информацию, которая используется другими для улучшения.
  - Распределение вычислительных ресурсов может быть произвольным и осуществляться из соображений удобности разработки системы.

Выводы

- Различия между методами планирования
  - Распределение обновлений:
    - Приоритетная выборка примеров.
    - Эвристический поиск.
    - Распределение согласно траектории.
  - Размер обновлений:
    - Меньший размер обновления позволяет сделать метод более инкрементным.
    - Самое короткое – одношаговые обновления по примерам.
    - Есть основания полагать, что подобные обновления являются наиболее эффективными для больших задач.
  - Глубина обновлений
    - Глубокие обновления часто могут быть реализованы в виде последовательности мелких обновлений.

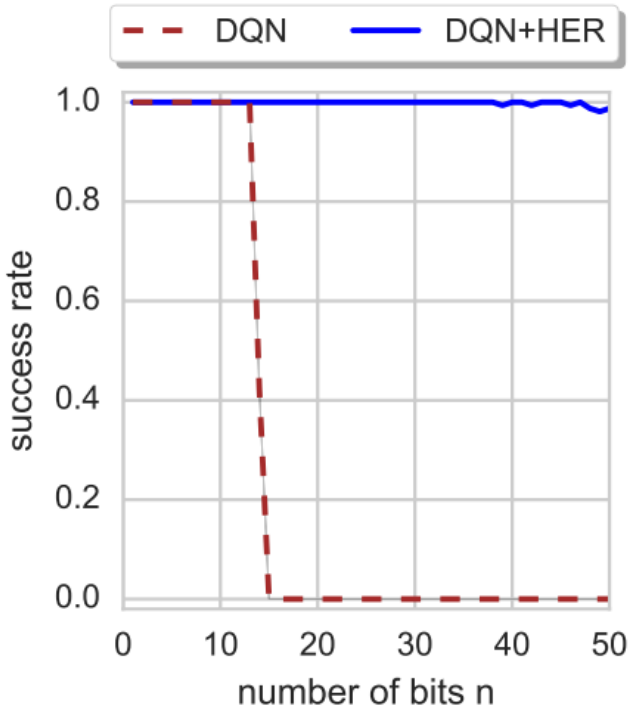
HINDSIGHT EXPERIENCE REPLAY

Задача – переключение битов

$$\text{Initial state} : \{0,1\}^n, e.g. [0,1,1,0,\ldots 1]$$

$$\text{Final Goal} : \{0,1\}^n, e.g. [1,0,0,1,\ldots 0]$$

$$\text{Action Space} : i \in \{0,1,2,\ldots,n-1\}$$



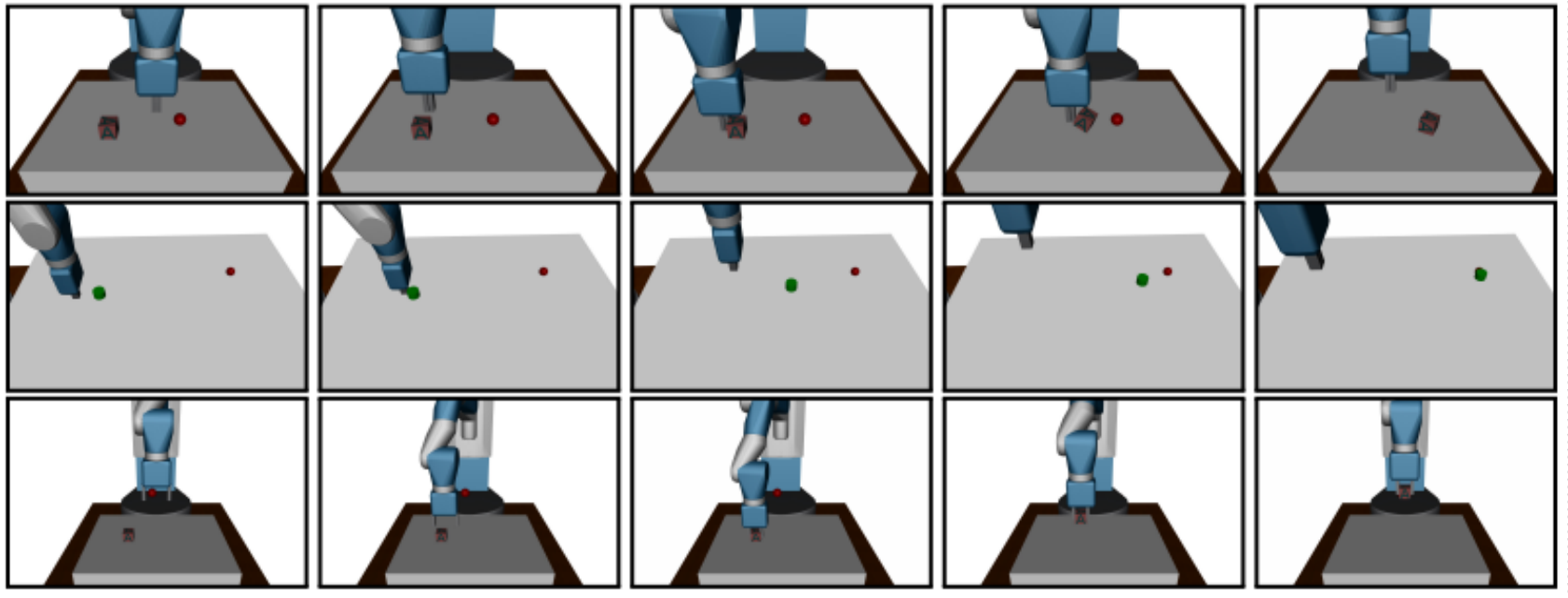
Algorithm 1 Hindsight Experience Replay (HER)

Given:

- an off-policy RL algorithm $\mathbb{A}$, ▷ e.g. DQN, DDPG, NAF, SDQN
- a strategy $\mathbb{S}$ for sampling goals for replay, ▷ e.g. $\mathbb{S}(s_0, \ldots, s_T) = m(s_T)$
- a reward function $r : \mathcal{S} \times \mathcal{A} \times \mathcal{G} \to \mathbb{R}$. ▷ e.g. $r(s, a, g) = -[f_g(s) = 0]$

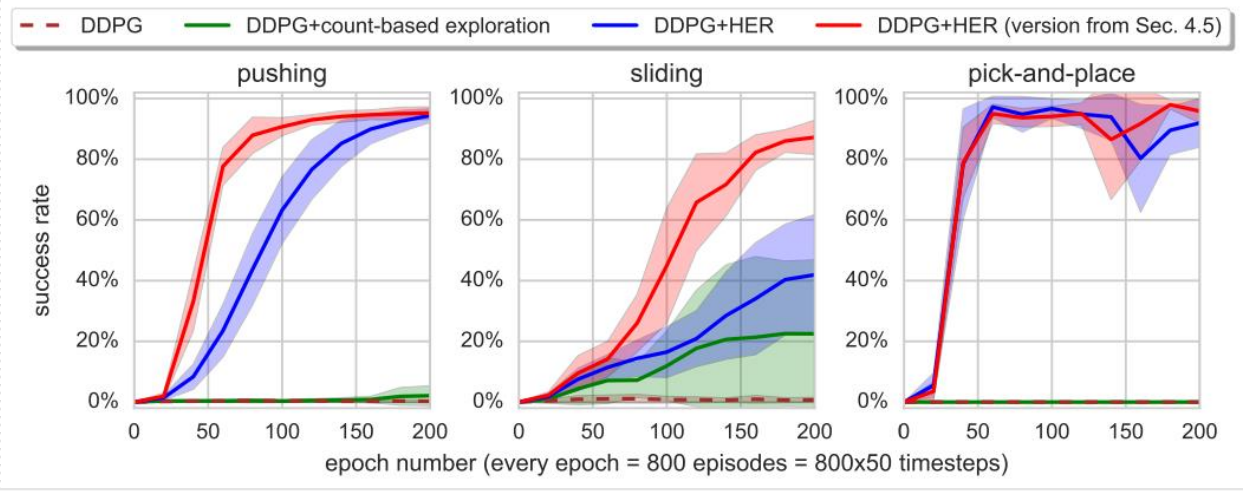
Initialize $\mathbb{A}$ ▷ e.g. initialize neural networks
Initialize replay buffer R
for episode = 1, M **do**
  Sample a goal g and an initial state s_0.
  for $t = 0, T-1$ **do**
    Sample an action a_t using the behavioral policy from $\mathbb{A}$:
      $a_t \leftarrow \pi_b(s_t||g)$ ▷ $||$ denotes concatenation
    Execute the action a_t and observe a new state s_{t+1}
  end for
  for $t = 0, T-1$ **do**
    $r_t := r(s_t, a_t, g)$
    Store the transition $(s_t||g,\ a_t,\ r_t,\ s_{t+1}||g)$ in R ▷ standard experience replay
    Sample a set of additional goals for replay $G := \mathbb{S}$(**current episode**)
    for $g' \in G$ **do**
      $r' := r(s_t, a_t, g')$
      Store the transition $(s_t||g',\ a_t,\ r',\ s_{t+1}||g')$ in R ▷ HER
    end for
  end for
  for $t = 1, N$ **do**
    Sample a minibatch B from the replay buffer R
    Perform one step of optimization using $\mathbb{A}$ and minibatch B
  end for
end for

Пример – управление роботом

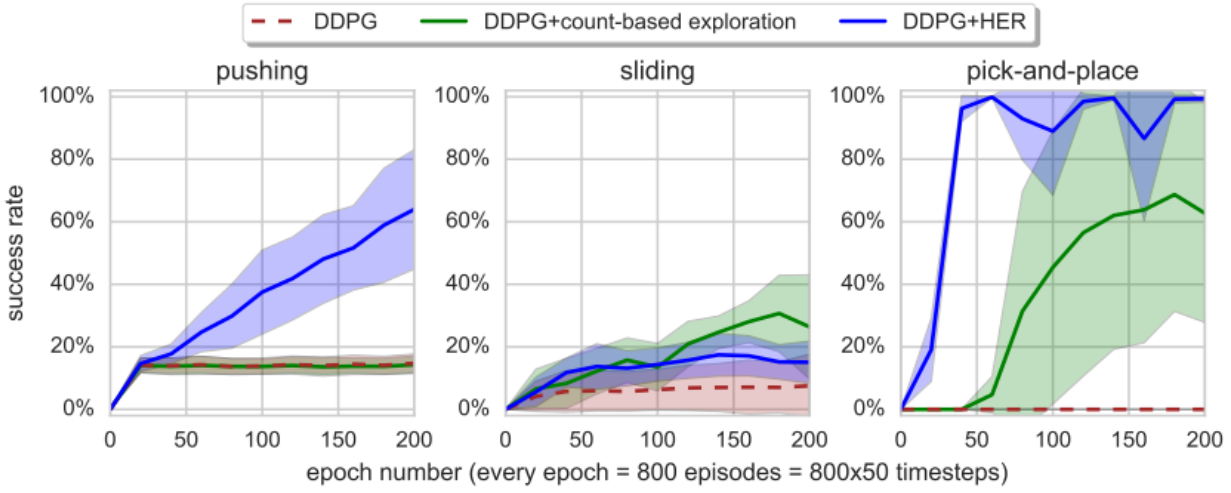


Пример – управление роботом

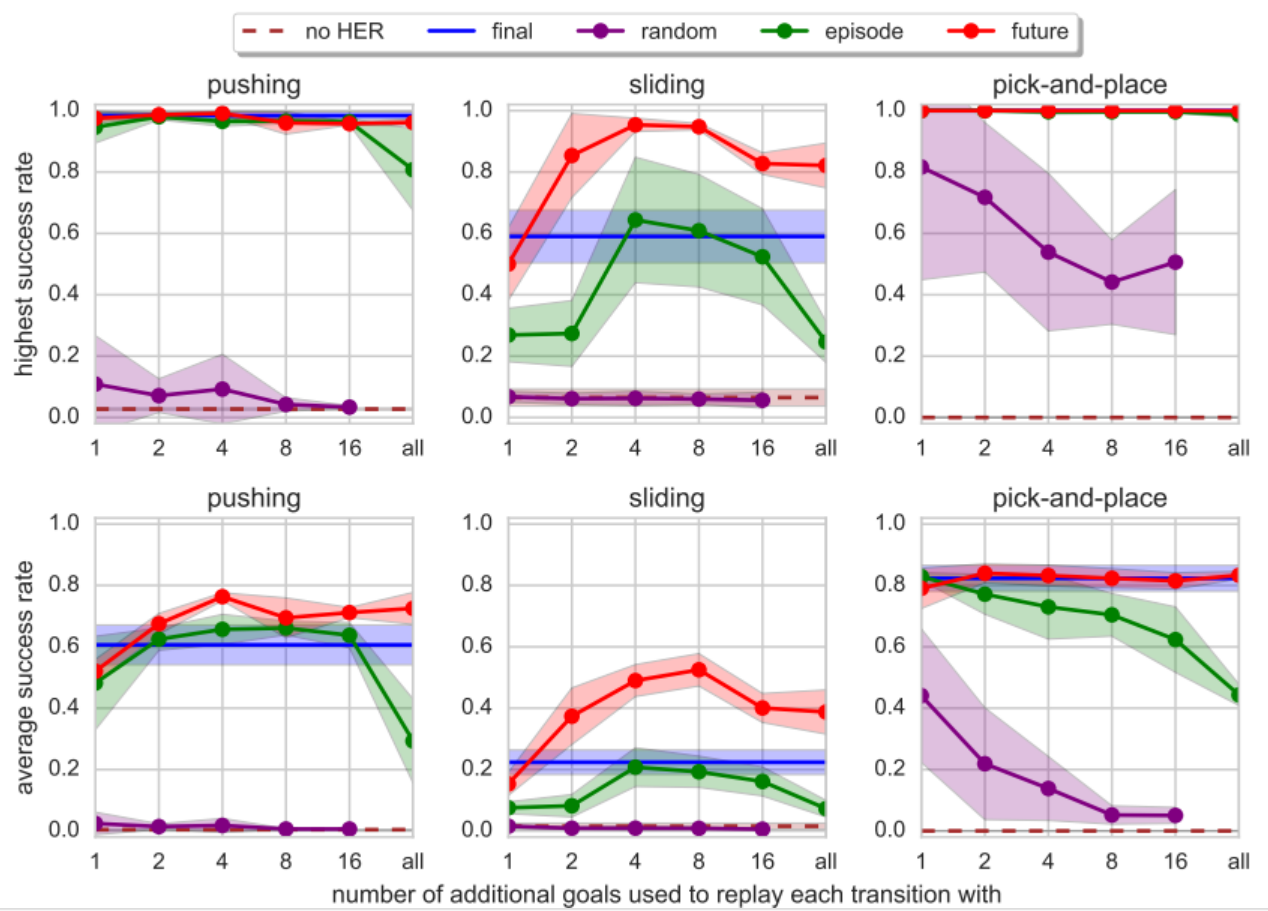
Несколько целей



Одна цель

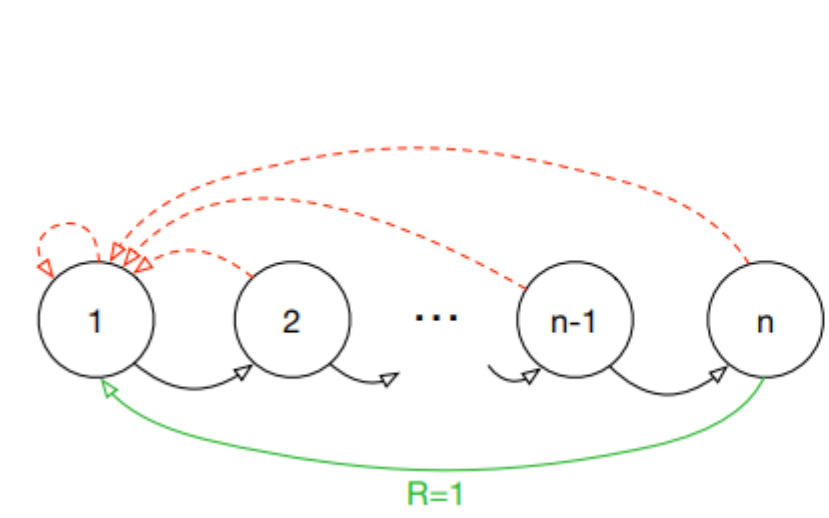


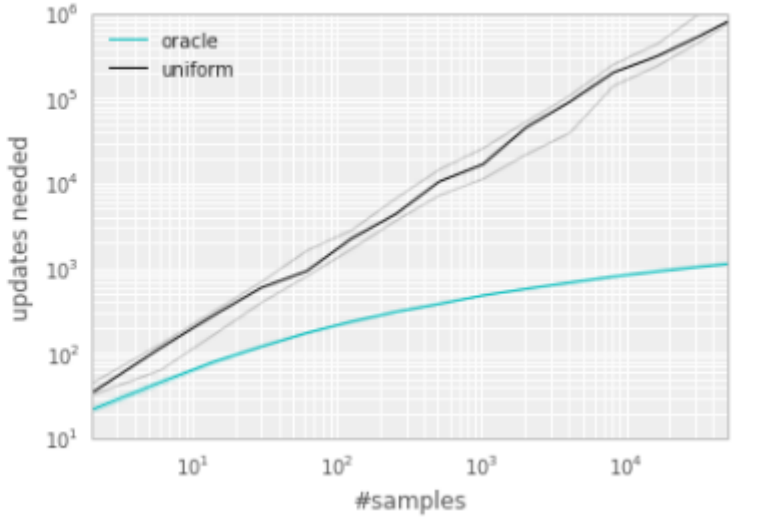
Какие цели выбирать?



PRIORITIZED EXPERIENCE REPLAY

Пример «прогулка по обрыву»

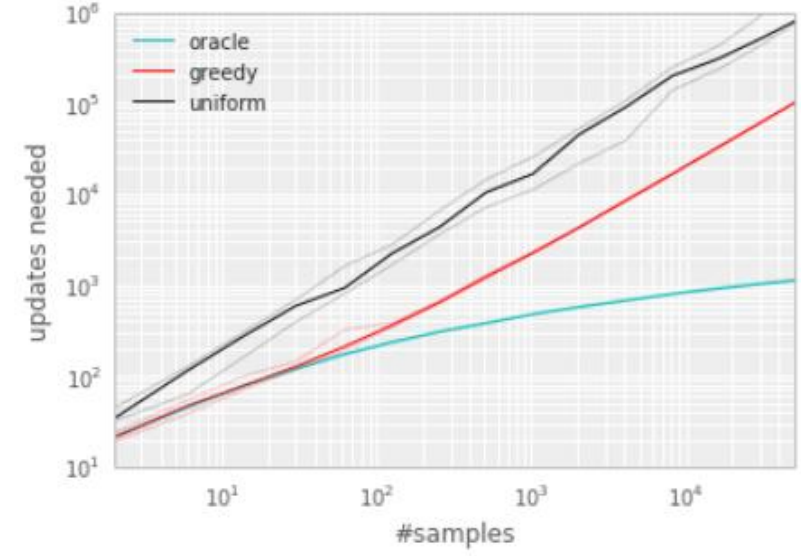


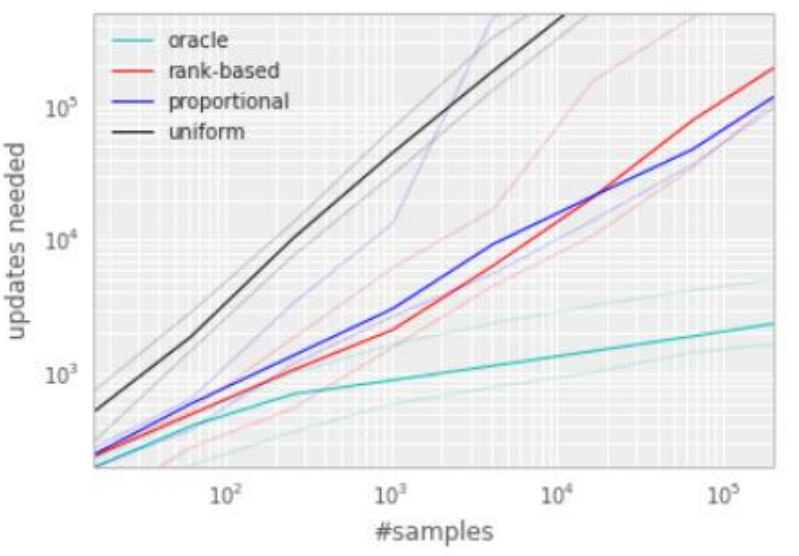


Algorithm 1 Double DQN with proportional prioritization

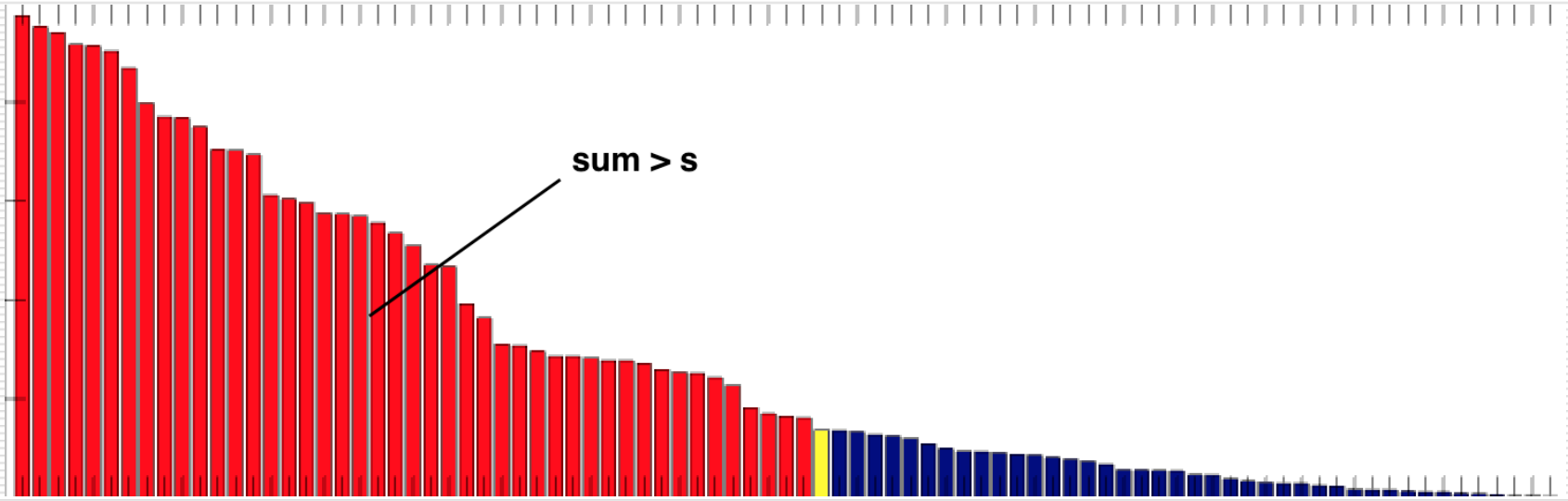
Input: minibatch k, step-size η, replay period K and size N, exponents α and β, budget T.
Initialize replay memory $\mathcal{H} = \emptyset$, $\Delta = 0$, $p_1 = 1$
Observe S_0 and choose $A_0 \sim \pi_\theta(S_0)$
for $t = 1$ **to** T **do**
  Observe S_t, R_t, γ_t
  Store transition $(S_{t-1}, A_{t-1}, R_t, \gamma_t, S_t)$ in $\mathcal{H}$ with maximal priority $p_t = \max_{i<t} p_i$
  if $t \equiv 0 \mod K$ **then**
    for $j = 1$ **to** k **do**
      Sample transition $j \sim P(j) = p_j^\alpha / \sum_i p_i^\alpha$
      Compute importance-sampling weight $w_j = (N \cdot P(j))^{-\beta} / \max_i w_i$
      Compute TD-error $\delta_j = R_j + \gamma_j Q_{\text{target}}(S_j, \arg\max_a Q(S_j, a)) - Q(S_{j-1}, A_{j-1})$
      Update transition priority $p_j \leftarrow |\delta_j|$
      Accumulate weight-change $\Delta \leftarrow \Delta + w_j \cdot \delta_j \cdot \nabla_\theta Q(S_{j-1}, A_{j-1})$
    end for
    Update weights $\theta \leftarrow \theta + \eta \cdot \Delta$, reset $\Delta = 0$
    From time to time copy weights into target network $\theta_{\text{target}} \leftarrow \theta$
  end if
  Choose action $A_t \sim \pi_\theta(S_t)$
end for

Результаты для «прогулки по обрыву»

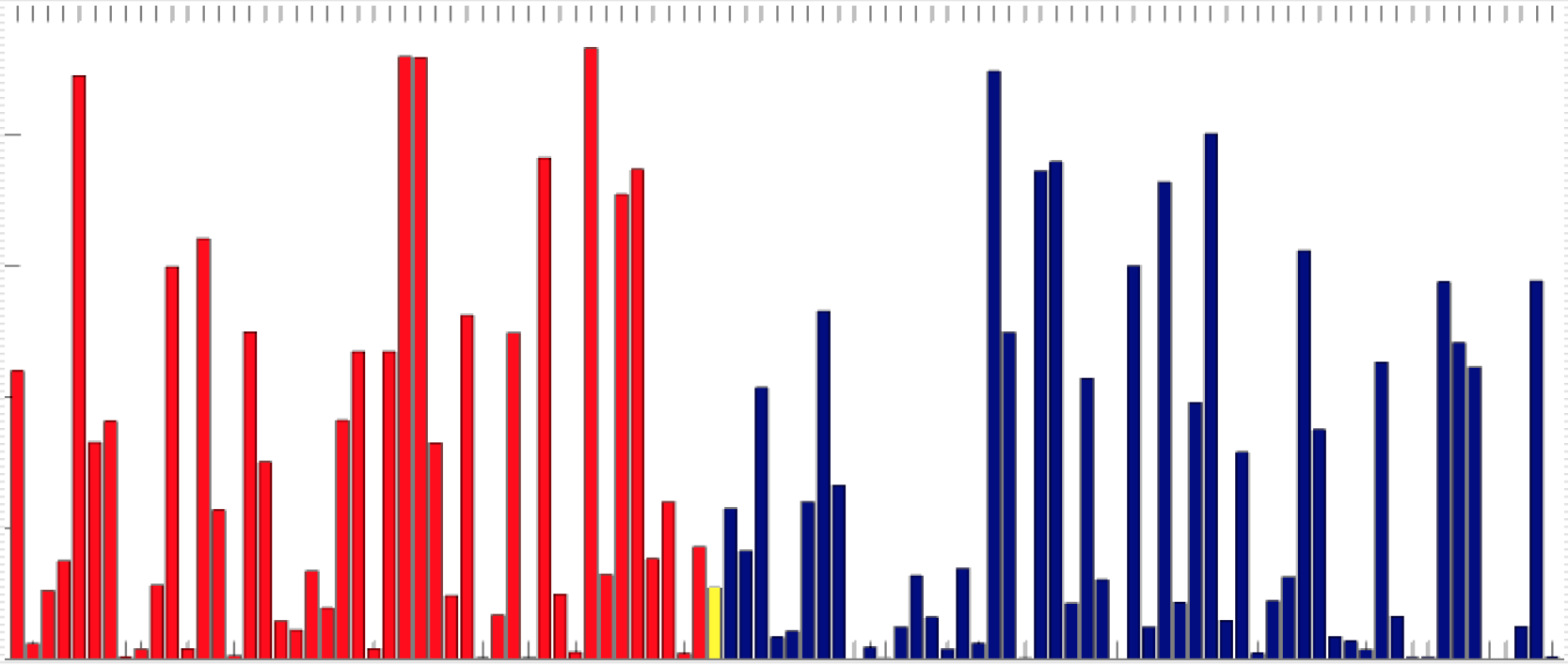




Реализация приоритетной выборки

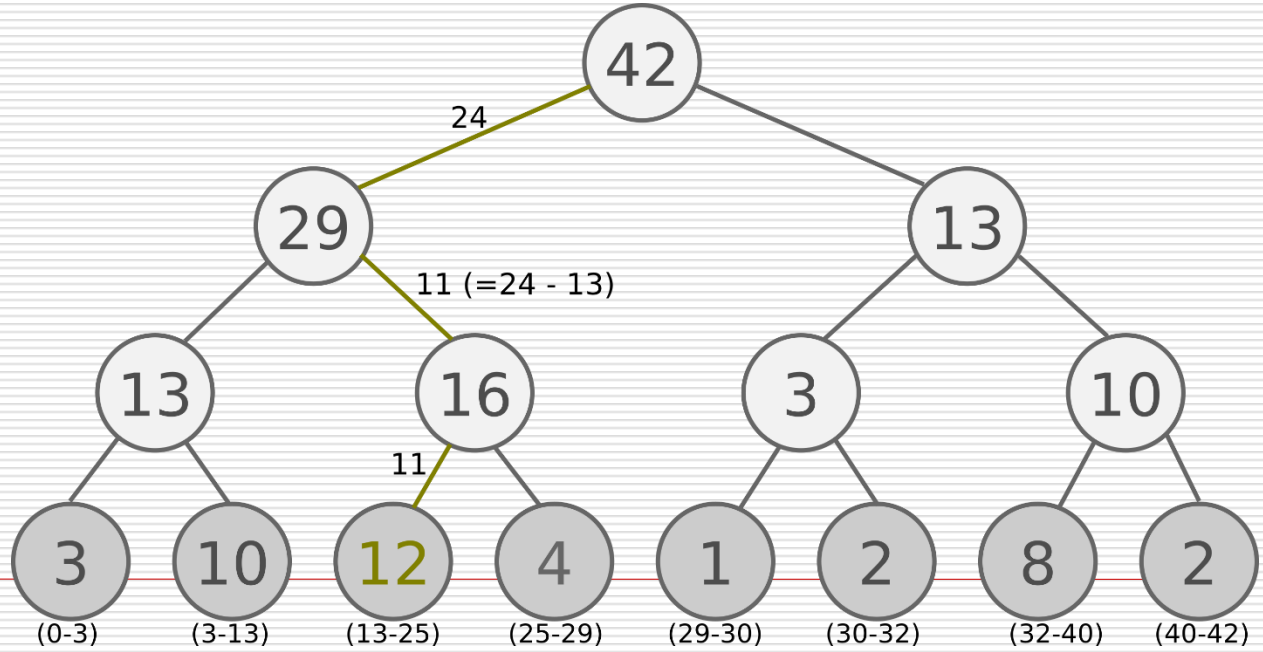


Реализация приоритетной выборки



Реализация приоритетной выборки - sumtree

```
def retrieve(n, s):
  if n is leaf_node: return n
  if n.left.val >= s: return retrieve(n.left, s)
  else: return retrieve(n.right, s - n.left.val)
```



Результаты для Atari