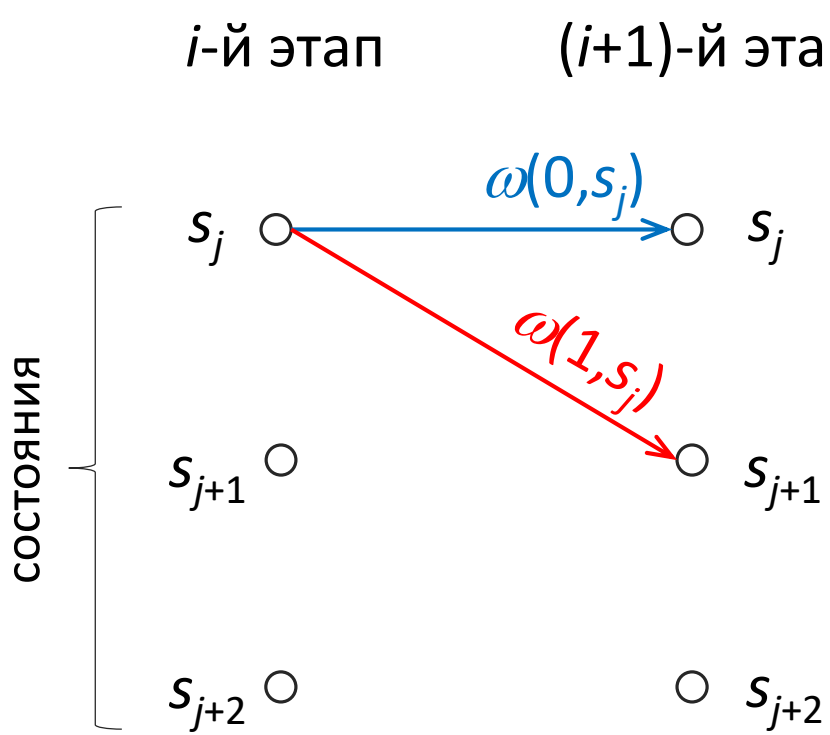
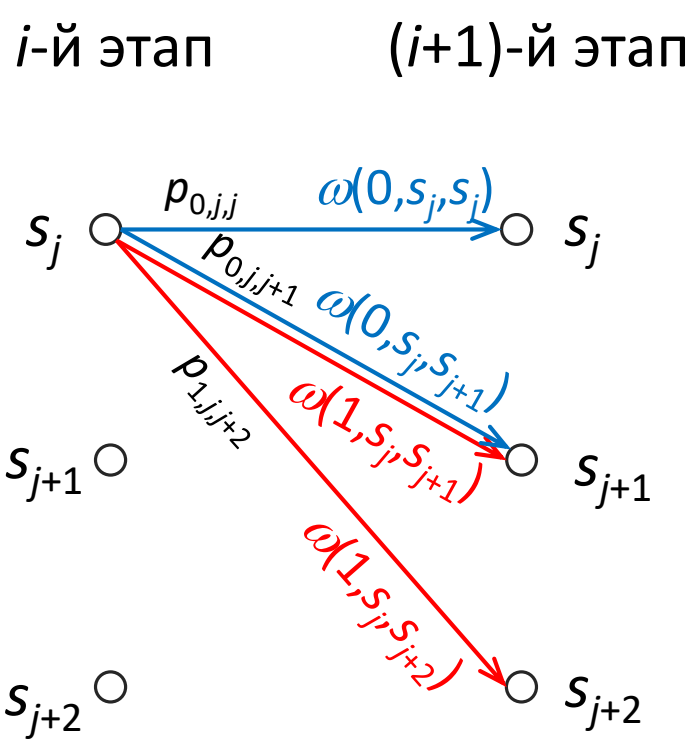
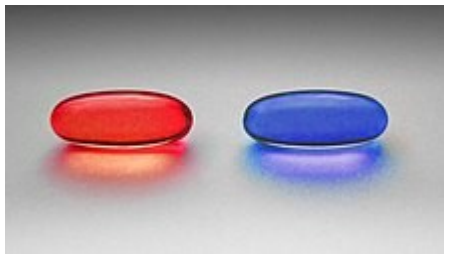
Динамическое программирование. Задачи с неопределенностью

Неопределенность в задаче управления





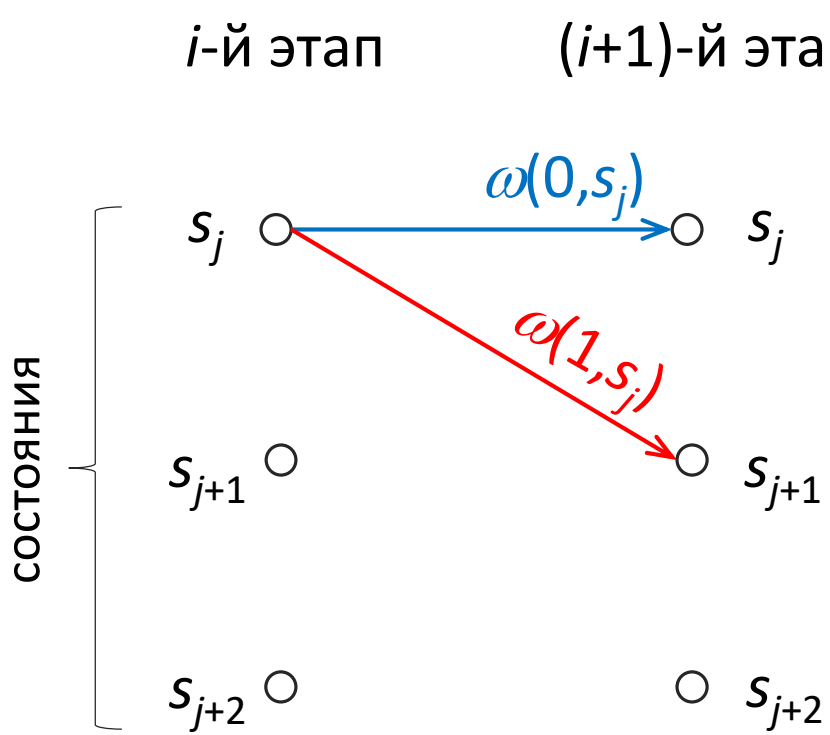


0

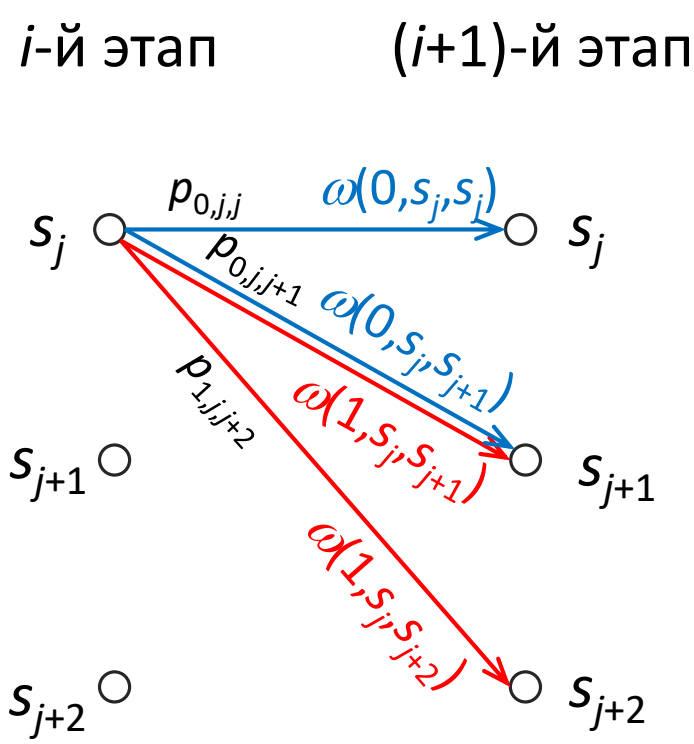
1

- разные значения управления

Неопределенность в задаче управления



$$W_i(s_j) = \max_{x \in X}\{\omega(x, s_j) + W_{i+1}(\varphi(s_j))\}$$



СВ

$$W_i(s_j) = \max_{x \in X}\{M[\omega(x, s_j, \varphi(x, s_j)) + W_{i+1}(\varphi(x, s_j))]\} =$$

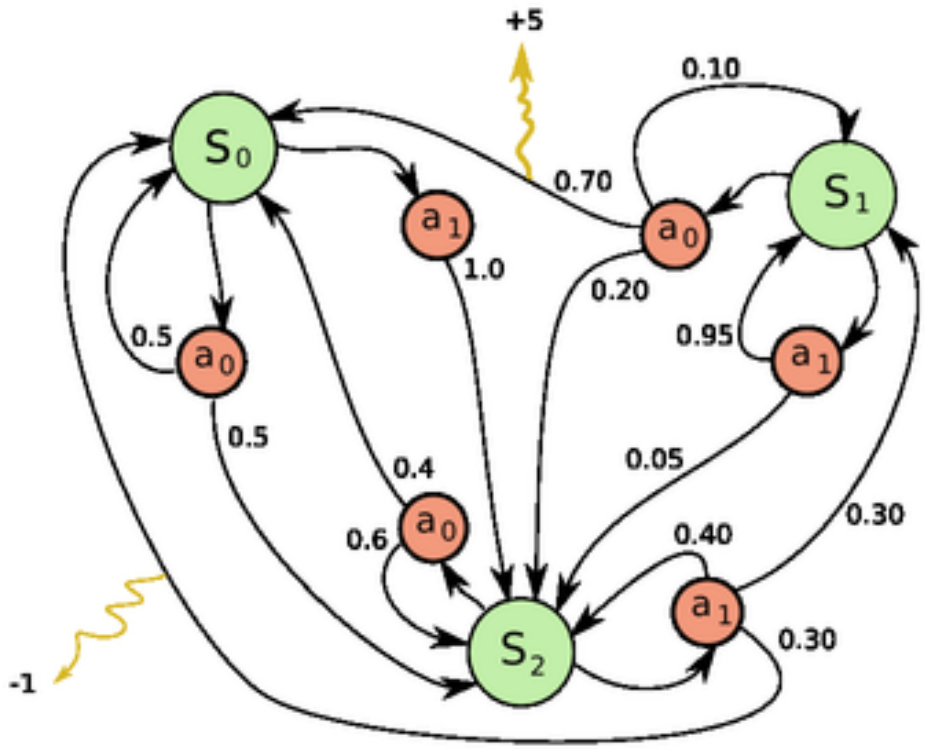
$$= \max_{x \in X}\left\{\sum_q p_{x,j,q}(\omega(x, s_j, s_q) + W_{i+1}(s_q))\right\}$$

Траектории

- Детерминированный случай:
  - Траектория полностью определена. Решив задачу, можем восстановить путь от начального состояния до конечного.
- Задача с неопределенностью:
  - Даже после выбора оптимального управления для каждого состояния мы можем оценить лишь распределение вероятностей для каждого из шагов.
  - Траектория «рождается» по мере реализации процесса. Фактическое выполнение дает нам новую **информацию**.
    - А значит, мы можем уточнять оценку стоимости, переходя от математического ожидания к фактическим значениям.

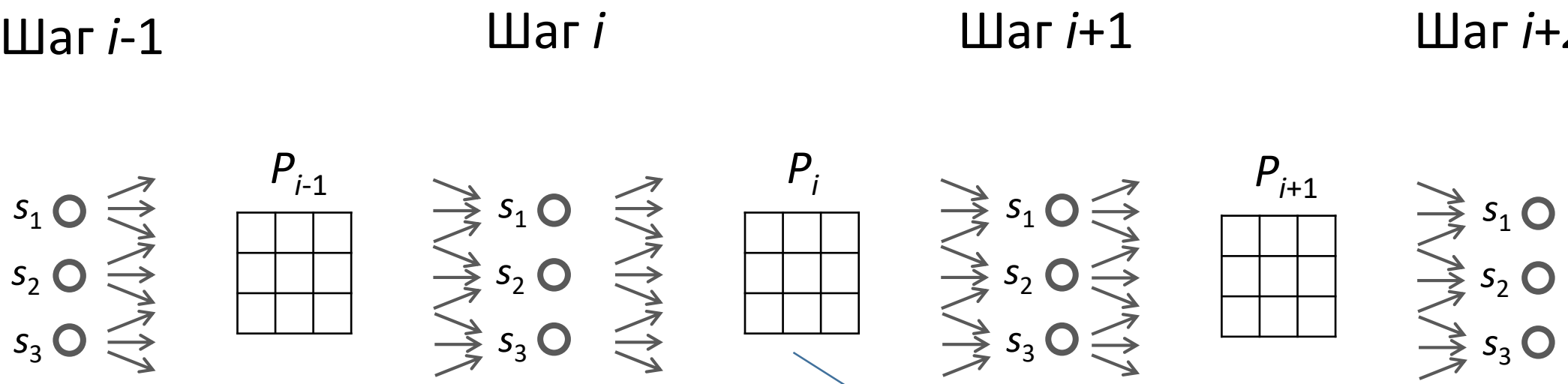
Марковский процесс принятия решений

- Дискретное время (этапы)
- Конечное число состояний
- Переходные вероятности между состояниями описывают *марковскую цепь*
  - Условное распределение последующего состояния не зависит от предыстории состояний
- Структура вознаграждений представима в виде матрицы дохода при переходе



Источник рисунка: https://commons.wikimedia.org/wiki/File:Markov_Decision_Process_example.png

Марковский процесс (марковская цепь)



$P_i(k,j) = p(S^{(i+1)} = s_j \mid S^{(i)} = s_k)$

Для перехода от шага *i* к шагу *i*+1:

$p_{i+1} = p_i P_i$

p_i – вектор-строка вероятностей

Марковский процесс (марковская цепь)

Для перехода от шага *i* к шагу *i*+1:

$p_{i+1} = p_i P_i$

$$\mathbb{P}(S^{(i+1)} = s_j) = \sum_{s_k} \mathbb{P}(S^{(i+1)} = s_j | S^{(i)} = s_k)\mathbb{P}(S^{(i)} = s_k)$$

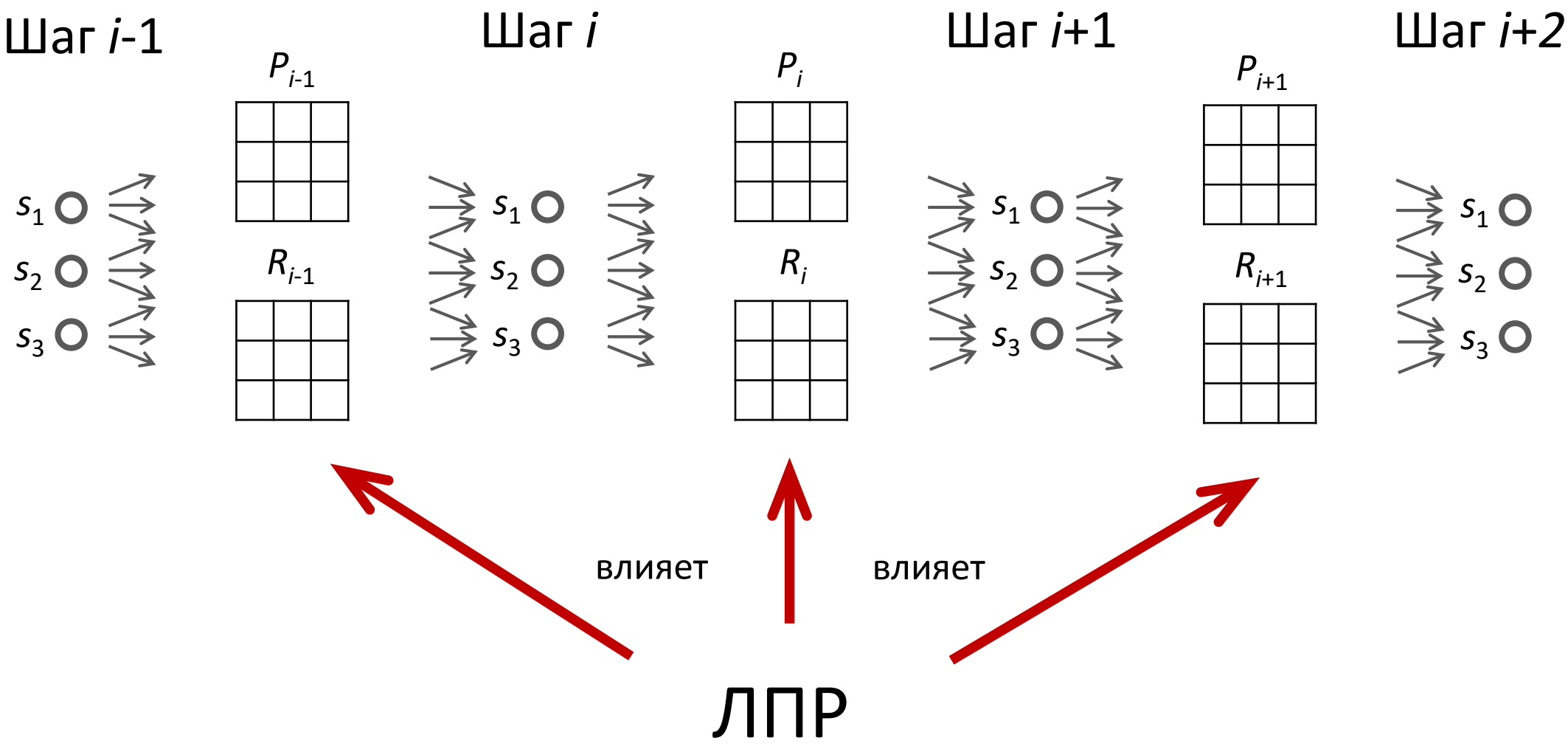
s_j

$\mathbb{P}(S^{(i)} = s_1)$	$\mathbb{P}(S^{(i)} = s_2)$	$\mathbb{P}(S^{(i)} = s_3)$

$\cdot$

…	$\mathbb{P}(S^{(i+1)} = s_j \mid S^{(i)} = s_1)$	…
…	$\mathbb{P}(S^{(i+1)} = s_j \mid S^{(i)} = s_1)$	…
…	$\mathbb{P}(S^{(i+1)} = s_j \mid S^{(i)} = s_1)$	…

Марковский процесс принятия решений



Задача о садовнике

- Продуктивность сада зависит от состояния почвы на начало сезона и может оцениваться как
  - 1) «хорошая»,
  - 2) «удовлетворительная»,
  - 3) «плохая».
- Состояние почвы в текущем году зависит **только от состояния почвы в предыдущем году** (с учетом проведенных агротехнических мероприятий).
- Необходимо выбрать наилучшую стратегию внесения удобрений на конечное число этапов.

Задача о садовнике

Матрицы дохода

Состояние почвы в следующем году

Состояние почвы в этом году

	хор.	уд.	пл.
хор.	0,2	0,5	0,3
уд.	0	0,5	0,5
пл.	0	0	1

$= P^{(1)}$

7	6	3
0	5	1
0	0	-1

$= R^{(1)}$

Не вносим удобрения

Матрицы вероятностей переходов

0,3	0,6	0,1
0,1	0,6	0,3
0,05	0,4	0,55

$= P^{(2)}$

6	5	-1
7	4	0
6	3	-2

$= R^{(2)}$

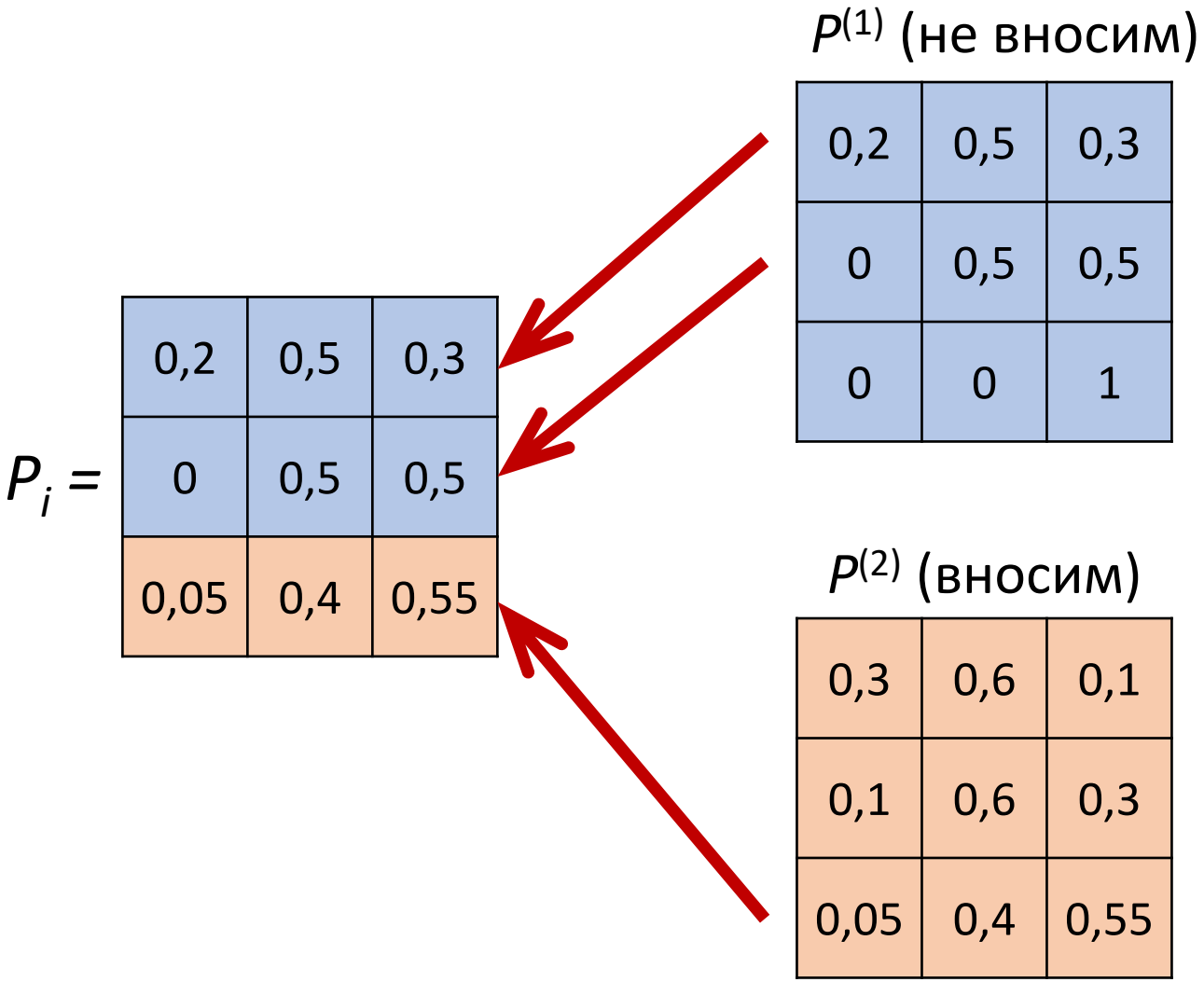
Вносим удобрения

Стратегия

- Стратегия – **правило** выбора действия **для каждого возможного состояния**
- Например, полностью заданная стратегия для некоторого **одного** шага может «говорить», что:
  - Если состояние почвы «хорошее», то «не вносить».
  - Если состояние почвы «удовлетворительное», то «не вносить».
  - Если состояние почвы «плохое», то «вносить».

Стратегия в МППР для одного шага

Состояние	Действие (управление)
Хорошее	Не вносить
Удовл.	Не вносить
Плохое	Вносить



Стратегия в МППР для одного шага

Состояние	Действие (управление)
Хорошее	Не вносить
Удовл.	Не вносить
Плохое	Вносить

$P_i =$

0,2	0,5	0,3
0	0,5	0,5
0,05	0,4	0,55

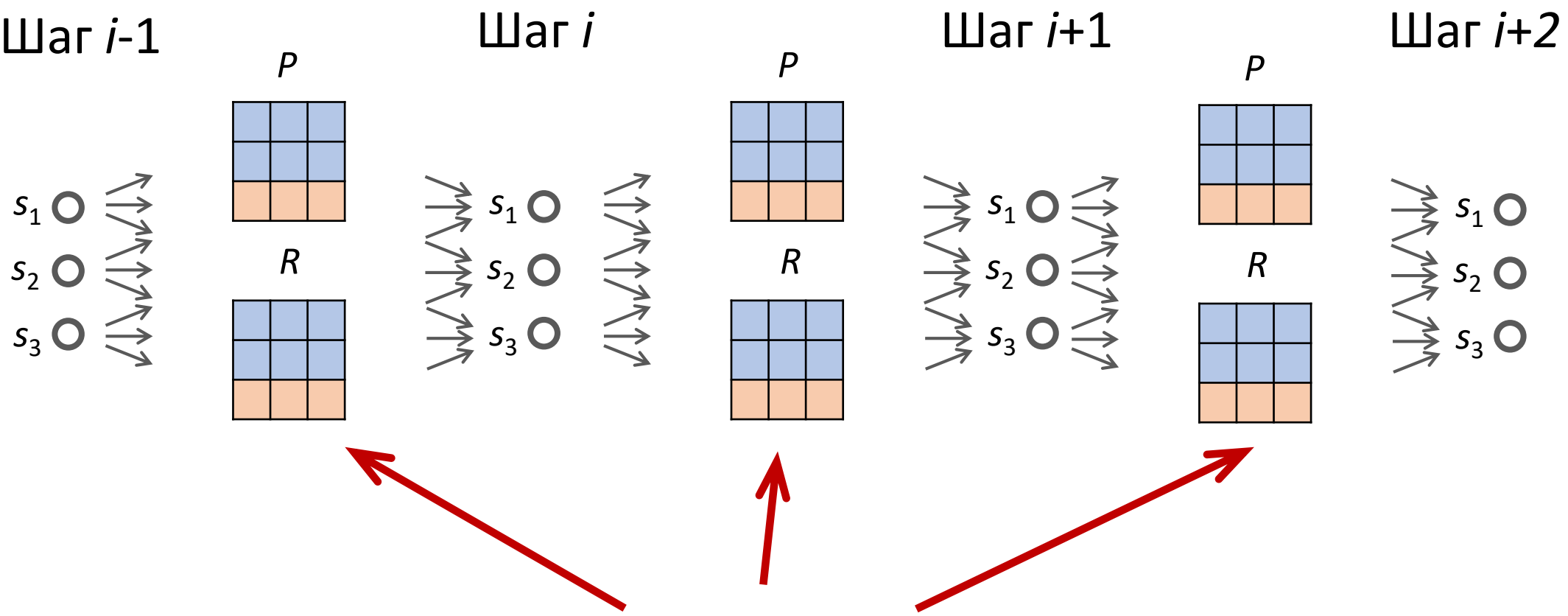
$R_i =$

7	6	3
0	5	1
6	3	-2

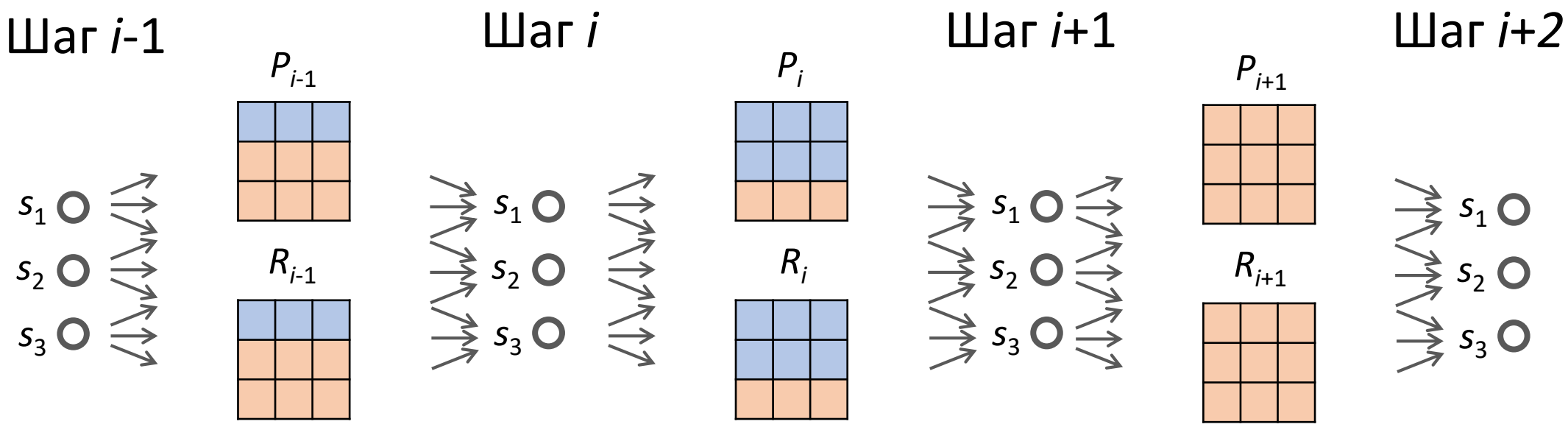
Задачи

- Оценка стратегии (предсказание)
- Оптимизация стратегии (управление)

Стационарная стратегия в МППР



Стратегия в МППР для всех шагов



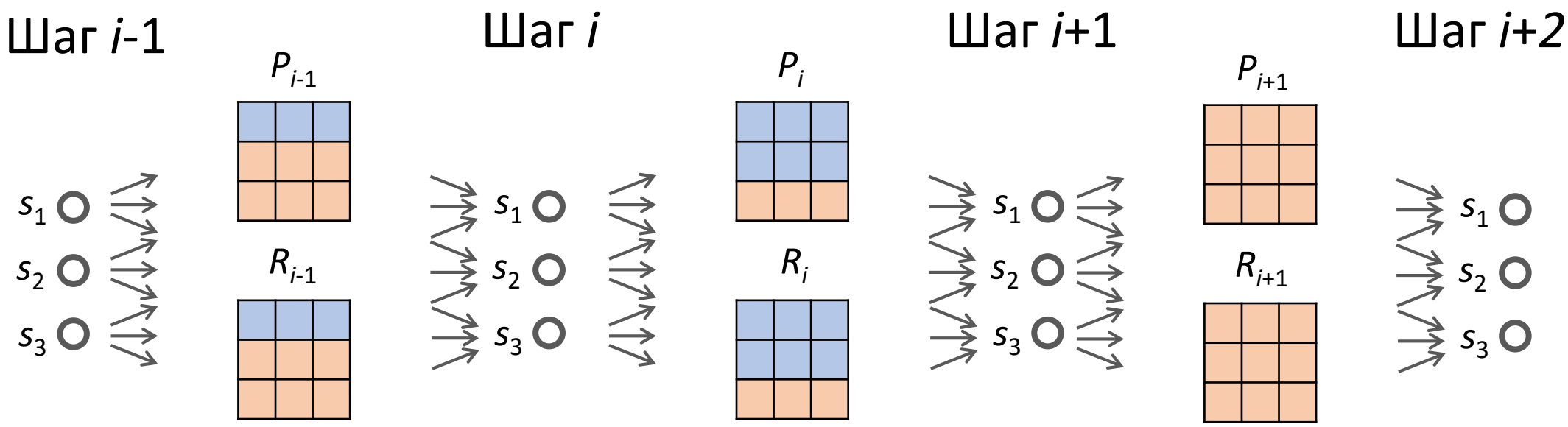
Для перехода от шага i к шагу i+1:

$$p_{i+1} = p_i P_i$$

$$v_{i,i+1} = sum\{p_i (P_i \circ R_i)\}$$

p_i – вектор-строка вероятностей

Стратегия в МППР для всех шагов



Рекуррентная формула для вычисления условного ожидаемого выигрыша на шагах, начиная *i*-того (оценка стратегии):

$$f_i(s_j) = \sum_{s_k \in S} P_i(s_j, s_k)\left(R_i(s_j, s_k) + f_{i+1}(s_k)\right)$$

Оценка выигрыша при заданной стратегии

```
1.  def finite_horizon_mdp_eval(P, R, periods):
2.      # Количество состояний
3.      n_states = P.shape[1]
4.
5.      profit = np.zeros((1, n_states))
6.      for period in reversed(range(periods)):
7.          profit = np.sum(P[period] * (R[period] + profit), axis=-1)
8.      return profit
```

$$f_i(s_j) = \sum_{s_k \in S} P_i(s_j, s_k)\left(R_i(s_j, s_k) + f_{i+1}(s_k)\right)$$

https://bitbucket.org/M_H/dmt-examples/src/master/Dynamic%20Programming.ipynb

Оптимизация стратегии методом ДП

Уравнение Беллмана:

$$W_i(s_j) = \max_{u \in U} \sum_{s_k \in S} P_i^{(u)}(s_j, s_k) \left(R_i^{(u)}(s_j, s_k) + W_{i+1}(s_k) \right)$$

Оптимизация стратегии (Python)

```
1.  # Предполагается, что P и R имеют размерность
2.  # 'количество управлений' x 'кол-во состояний' x 'кол-во состояний'
3.  def finite_horizon_mdp_solver(P, R, periods):
4.      # Количество состояний
5.      n_states = P.shape[1]
6.      # Сформируем результирующие матрицы
7.      profit = np.zeros((periods, n_states))                # Условно оптимальные выигрыши
8.      control = np.zeros((periods, n_states), dtype='int32') # Условно оптимальные управления
9.      # Для последнего этапа выигрыш на "следующем" этапе
10.     # должен быть равен нулю
11.     profit_next = np.zeros((1, n_states))
12.     for period in reversed(range(periods)):
13.         tmp = np.sum(P * (R + profit_next), axis=-1)
14.         profit[period, :] = np.max(tmp, axis=0)
15.         control[period, :] = np.argmax(tmp, axis=0)
16.         profit_next = profit[period, :]
17.     return profit, control
```

https://bitbucket.org/M_H/dmt-examples/src/master/Dynamic%20Programming.ipynb

Резюме

- Динамическое программирование оказывается полезным и при выборе оптимального управления в системах с неопределенностью
  - Например, максимизация *ожидаемой* выгоды
- Марковский процесс принятия решений (МППР, MDP) – широко распространенное обобщение учета неопределенности в задаче формирования управления
  - Решается (в том числе) с помощью динамического программирования
- Литература для дальнейшего углубления в тему:
  - Таха Х. Введение в исследование операций. 7е издание. – Вильямс, 2007.
  - Bertsekas D. Dynamic Programming and Optimal Control.