

# Uczenie ze wzmocnieniem

*(Reinforcement Learning)*

# Materiały dodatkowe

- Richard S. Sutton and Andrew G. Barto, *Reinforcement learning, an introduction*, second edition:

<http://incompleteideas.net/book/the-book-2nd.html>

- David Silver, *UCL Course on RL*:

<http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html>

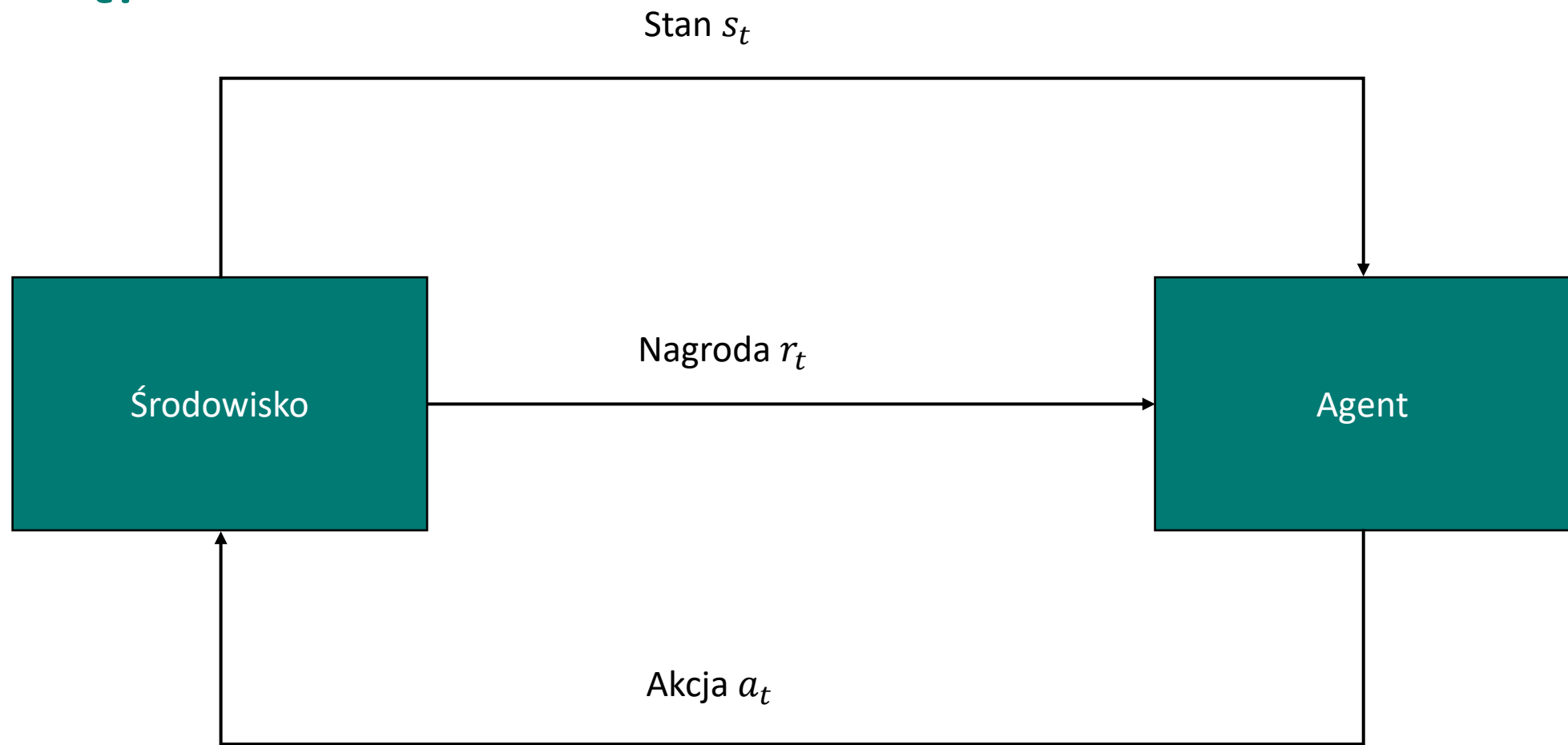
# Wstęp

- 3 podstawowe podejścia do uczenia maszynowego:
  - Uczenie nadzorowane
  - Uczenie nienadzorowane
  - Uczenie ze wzmocnieniem

# Wstęp

- 3 podstawowe podejścia do uczenia maszynowego:
  - Uczenie nadzorowane
  - Uczenie nienadzorowane
  - Uczenie ze wzmocnieniem
- Inspiracje:
  - Sieci neuronowe – budowa układu nerwowego żywych organizmów.
  - Uczenie ze wzmocnieniem – podejście behawioralne do uczenia.

# Wstęp



# Wstęp

**Hipoteza nagrody** - celem agenta jest maksymalizacja wartości oczekiwanej jego skumulowanej użyteczności.

# Łańcuch Markowa

- Ciąg zdarzeń, w którym prawdopodobieństwo każdego zdarzenia zależy jedynie od wyniku poprzedniego

$$P(X_{n+1} \leq y | X_0, X_1, X_2, \dots, X_n) = P(X_{n+1} \leq y | X_n)$$

# Łańcuch Markowa

- Ciąg zdarzeń, w którym prawdopodobieństwo każdego zdarzenia zależy jedynie od wyniku poprzedniego

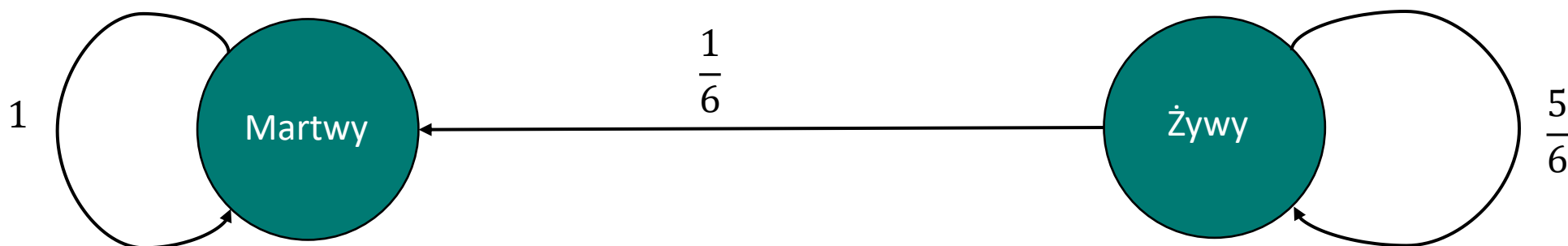
$$P(X_{n+1} \leq y | X_0, X_1, X_2, \dots, X_n) = P(X_{n+1} \leq y | X_n)$$

- **Własność Markowa** – warunkowe rozkłady prawdopodobieństwa przyszłych stanów procesu są zdeterminowane wyłącznie przez jego bieżący stan, bez względu na przeszłość.



# Łańcuch Markowa

- Przykład – gra w Rosyjską ruletkę:



- Macierz przejścia:

$$P = \begin{bmatrix} 1 & 0 \\ \frac{1}{6} & \frac{5}{6} \end{bmatrix}$$

# Łańcuch Markowa

- Macierz:

$$P^{(n)} = \begin{bmatrix} p_{11}^{(n)} & \cdots & p_{1n}^{(n)} \\ \vdots & \ddots & \vdots \\ p_{n1}^{(n)} & \cdots & p_{nn}^{(n)} \end{bmatrix}$$

nazywamy **macierzą przejść**.

- Gdy prawdopodobieństwa nie zależą od aktualnego stanu mówimy, że łańcuch jest **jednorodny**.

$$P = \begin{bmatrix} p_{11} & \cdots & p_{1n} \\ \vdots & \ddots & \vdots \\ p_{n1} & \cdots & p_{nn} \end{bmatrix}$$

# Przykład

## Monopoly



- Monopoly to jedna z najpopularniejszych gier planszowych w historii; jej historia sięga lat 30 XX wieku.
- Spróbujemy przedstawić ją jako łańcuch Markowa.
- Badamy tylko poruszanie się, na razie nie uwzględniamy decyzji graczy.

# Przykład

## Monopoly



- Gracze rzucają dwiema sześciennymi kostkami i przesuwają swój pionek o ilość pól, którą wyrzucił kostkami.
- Dla uproszczenia pominiemy zasady związane z dubletami (taką samą liczbę oczek na obu kostkach) - wyraźnie uprości to grę (**dlaczego?**).

## Przykład

# Monopoly



- Dodatkowo interesują nas jedynie karty typu „Kasa Społeczna”.
  - Takich kart jest **16**; **1** cofa gracza na **początek**, **1** wysyła go do **więzienia**, pozostałe nie mają wpływu na jego pozycję.
  - Karty typu „Szansa” nie zmieniają wyników a byłyby uciążliwe do zakodowania ;)

# Proces Markowa z nagrodami

- Proces Markowa z nagrodami (*Markov reward process* – MRP) definiujemy jako następującą krotkę  $(S, P, R, \beta)$ :
  - $S$  – jest skończonym zbiorem stanów.
  - $P(s, s')$  – Prawdopodobieństwo znalezienia się w stanie  $s'$  w czasie  $t + 1$  pod warunkiem znajdowania się w stanie  $s$  w czasie  $t$ .
  - $R(s, s')$  – Nagroda za przejście ze stanu  $s$  do stanu  $s'$  w czasie  $t$ .
  - $\beta$  – współczynnik dyskontujący  $\beta \in [0, 1)$

# Proces Markowa z nagrodami

- Skumulowana przyszła nagroda:

$$R_t = r_{t+1} + \beta r_{t+2} + \dots + \beta^{T-t} r_{T-t} = \sum_{k=1}^{T-t} \beta^{k-1} r_{t+k}$$

- **Funkcję wartości**  $V(s)$  definiujemy jako oczekiwaną wypłatę zaczynając od stanu  $s$

$$V(s) = E[R_t | S_t = s]$$

# Równanie Bellmana

- Oba równania można zdekomponować do postaci natychmiastowej nagrody i zdyskontowanej przyszłej wartości:

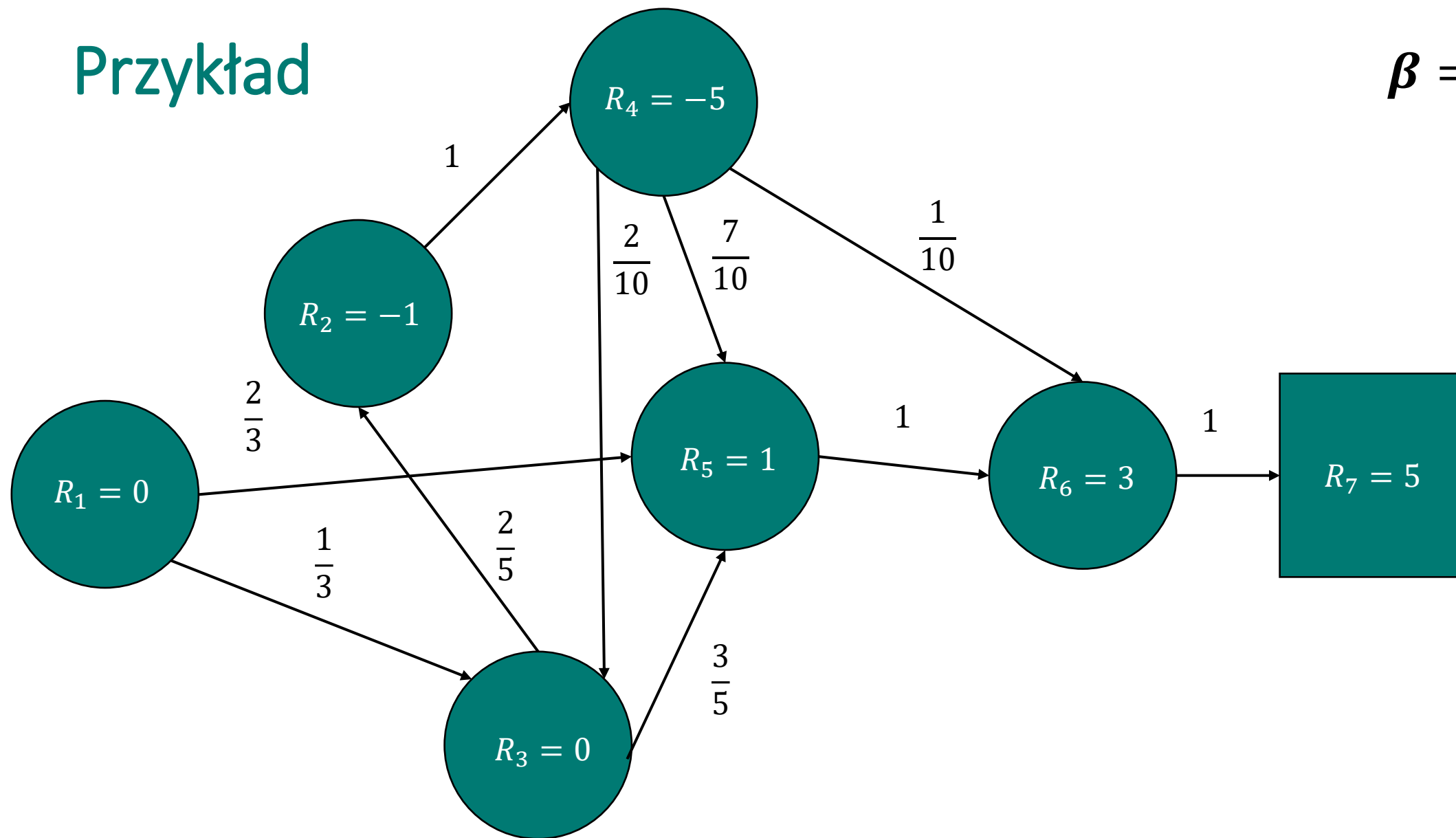
$$V(s) = E[R_t | S_t = s] = E[r_{t+1} + \beta V(S_{t+1}) | S_t = s]$$

Taka postać nosi nazwę **Równania Bellmana**



# Przykład

$$\beta = 1$$

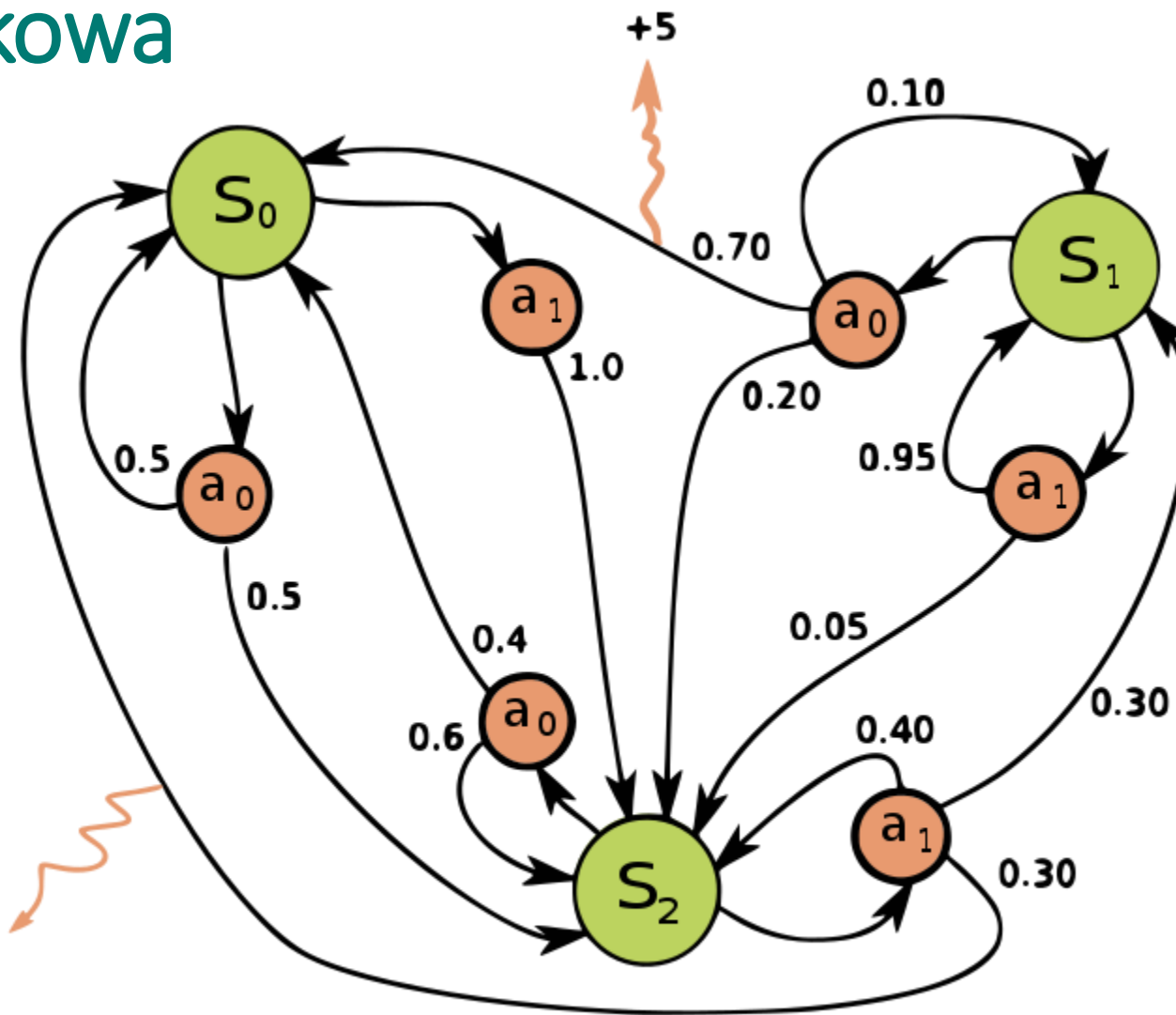


# Proces decyzyjny Markowa

- Proces decyzyjny Markowa (*Markov decision process* – MDP) definiujemy jako następującą krotkę  $(S, A, P, R, \beta)$ :
  - $S$  – jest skończonym zbiorem stanów.
  - $A$  – jest skończonym zbiorem akcji.
  - $P(s, a, s')$  – Prawdopodobieństwo znalezienia się w stanie  $s'$  w czasie  $t + 1$  w wyniku podjęcia działania  $a$  w stanie  $s$  w czasie  $t$ .
  - $R(s, a, s')$  – Nagroda za przejście ze stanu  $s$  do stanu  $s'$  wyniku podjęcia działania  $a$  w czasie  $t$ .
  - $\beta$  – współczynnik dyskontujący  $\beta \in [0, 1)$

# Proces decyzyjny Markowa

- Proces decyzyjny Markowa (*Markov decision process – MDP*) definiujemy jako następującą krotkę  $(S, A, P, R, \beta)$ :
  - $S$  – jest skończonym zbiorem stanów.
  - $A$  – jest skończonym zbiorem akcji.
  - $P(s, a, s')$  – Prawdopodobieństwo znalezienia się w stanie  $s'$  w czasie  $t + 1$  w wyniku podjęcia działania  $a$  w stanie  $s$  w czasie  $t$ .
  - $R(s, a, s')$  – Nagroda za przejście ze stanu  $s$  do stanu  $s'$  wyniku podjęcia działania  $a$  w czasie  $t$ .
  - $\beta$  – współczynnik dyskontujący  $\beta \in [0, 1)$



# Proces decyzyjny Markowa

- **Strategię** definiujemy jako:

$$\pi: S \rightarrow A$$
$$\pi(a|s) = P[A_t = a | S_t = s]$$

- Strategia w pełni opisuje zachowania agenta.
- Strategie zależą jedynie od aktualnego stanu MDP a nie od historii.
- Są *stacjonarne* (nie zależą od czasu).

# Proces decyzyjny Markowa

- **Funkcję wartości stanu** (*state-value function*)  $V_\pi(s)$  definiujemy jako oczekiwaną wypłatę zaczynając od stanu  $s$  i podążając za strategią  $\pi$ :

$$V_\pi(s) = E_\pi[R_t | S_t = s]$$

- **Funkcję wartości akcji** (*action-value function*)  $q_\pi(s, a)$  definiujemy jako oczekiwaną wypłatę zaczynając od stanu  $s$ , podejmując akcję  $a$  i podążając za strategią  $\pi$ :

$$q_\pi(s, a) = E_\pi[R_t | S_t = s | A_t = a]$$

# Równanie Bellmana

- Oba równania można zdekomponować do postaci natychmiastowej nagrody i zdyskontowanej przyszłej wartości:

$$V_{\pi}(s) = E_{\pi}[R_t | S_t = s] = E_{\pi}[r_{t+1} + \beta V_{\pi}(S_{t+1}) | S_t = s]$$

$$q_{\pi}(s, a) = E_{\pi}[R_t | S_t = s | A_t = a] = E_{\pi}[r_{t+1} + \beta q_{\pi}(S_{t+1}, A_{t+1}) | S_t = s | A_t = a]$$

Taka postać nosi nazwę **Równania Bellmana**

# Równanie Bellmana

- Upraszczając możemy zapisać oba równania w formie:

- $V_{\pi}(s) = R(s) + \beta P(s, \pi(s), s') V_{\pi}(s')$

- $q_{\pi}(s, \pi(s)) = R(s) + \beta P(s, \pi(s), s') q_{\pi}(s', \pi(s'))$

# Proces decyzyjny Markowa

- **Optymalna funkcja wartości stanu  $V_*$ :**

$$V_*(s) = \max_{\pi} V_{\pi}(s, a)$$

- **Optymalna funkcja wartości akcji  $q_*(s, a)$ :**

$$q_*(s, a) = \max_{\pi} q_{\pi}(s, a)$$



# Równanie Bellmana

- Dla każdego problemu, który można zdefiniować jako proces decyzyjny Markowa:
  - istnieje taka optymalna, deterministyczna strategia  $\pi_*$ , która jest nie gorsza od pozostałych dostępnych  $\pi_*(s) \geq \pi(s), \forall \pi \forall s \in S$
  - Optymalna strategia  $\pi_*$  osiąga optymalną wartość funkcji wartości  $V_{\pi_*}(s) = V_*(s)$ .
  - Optymalna strategia  $\pi_*$  osiąga optymalną wartość funkcji wartości akcji  $q_{\pi_*}(s, a) = q_*(s, a)$ .

# Równanie Bellmana

- Definiujemy ją jako:

$$\pi_*(s|a) = \begin{cases} 1 & \text{gdy } a = \operatorname{argmax}_{a \in A} q_*(s, a) \\ 0 & \text{w przeciwnym wypadku} \end{cases}$$

- Aby ją znaleźć musimy znaleźć maksymalizować po  $q_*(s, a)$ .
- Gdy znamy  $q_*(s, a)$  znalezienie optymalnej strategii jest trywialne.

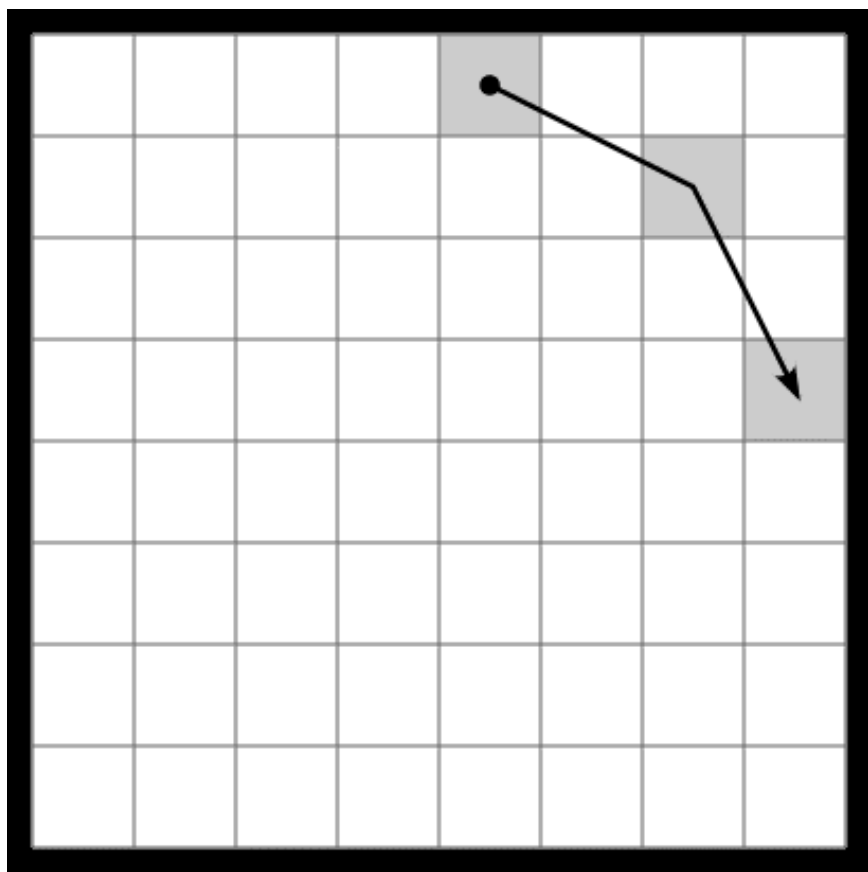
Wyszukiwanie wyczerpujące

# Exhaustive search

- Równania Bellmana można rozwiązać analitycznie.
- Złożoność obliczeniowa sięga  $O(n^3)$  gdzie  $n$  oznacza liczbę stanów.
- W przypadku rozwiązywania dużych problemów istnieje wiele metod iteracyjnych:
  - Exhaustive search
  - Programowanie dynamiczne
  - Metody Monte Carlo
  - Temporal Difference Learning

# Przykład

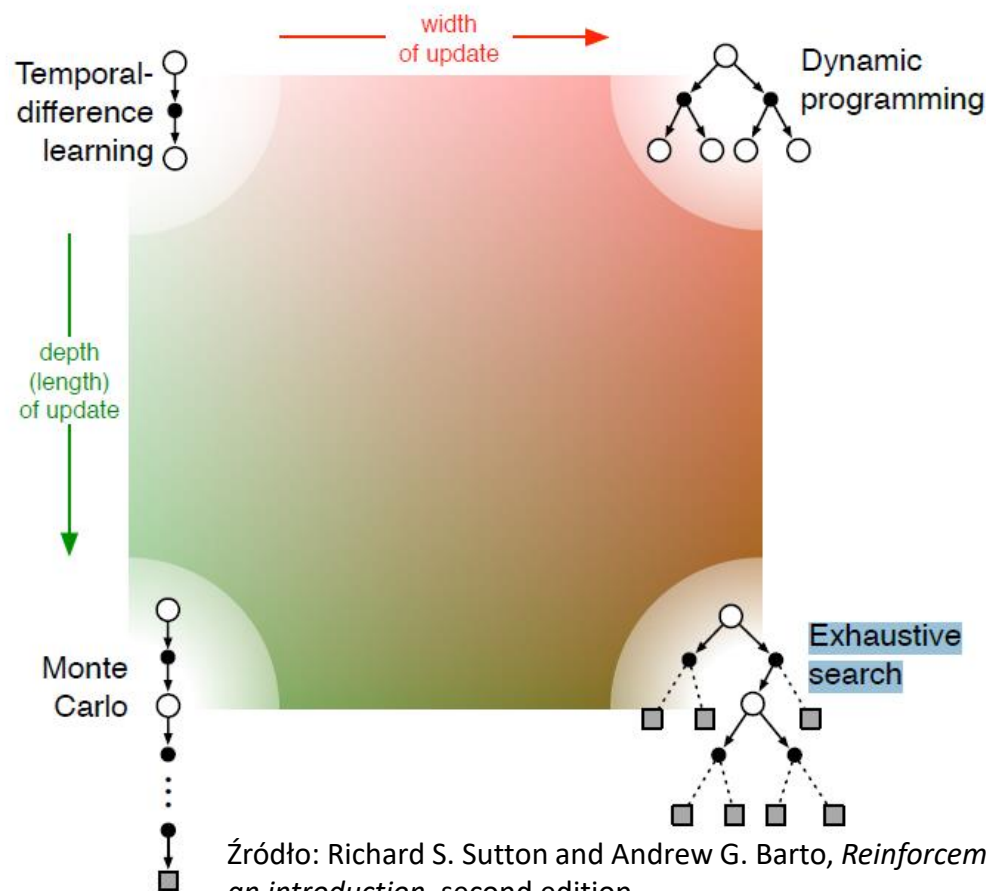
## Skoczek na szachownicy



- Celem jest obejście skoczkiem wszystkich pól planszy tak, żeby na każdym polu stanąć raz i tylko raz.

# Exhaustive search

- Np. backtracking
- Metody nieefektywne – wymagają przechowywania dużej ilości informacji:



Źródło: Richard S. Sutton and Andrew G. Barto, *Reinforcement learning, an introduction*, second edition