Roman ZAJDEL
POLITECHNIKA RZESZOWSKA,
Al. Powstańców Warszawy 12, 35-959 Rzeszów

Epokowo-inkrementacyjny algorytm uczenia się ze wzmocnieniem wykorzystujący kryterium średniego wzmocnienia

Dr inż. Roman ZAJDEL

Studia na wydziale Elektrycznym Politechniki Rzeszowskiej ukończył w 1990 roku. W 1999 roku uzyskał tytuł doktora na Politechnice Wrocławskiej. Pracuje na Politechnice Rzeszowskiej na stanowisku adiunkta. Główny kierunek badań naukowych to sieci neuronowe, systemy rozmyte oraz algorytmy uczenia się ze wzmocnieniem.

e-mail: rzajdel@prz.edu.pl

Streszczenie

W artykule zaproponowano nowy, epokowo – inkrementacyjny algorytm uczenia się ze wzmocnieniem. Główną ideą tego algorytmu jest przeprowadzenie w trybie epokowym dodatkowych aktualizacji strategii w oparciu o odległości aktywnych w przeszłości stanów od stanu terminalnego. Zaproponowany algorytm oraz algorytmy R(0)-learning, R(λ)-learning, Dyna-R oraz prioritized sweeping-R zastosowano do sterowania modelem samochodu górskiego oraz modelem kulki umieszczonej na balansującej belce.

Słowa kluczowe: uczenie się ze wzmocnieniem, R-learning, algorytm epokowo-inkrementacyjny.

The epoch-incremental reinforcement learning algorithm based on the average reward

Abstract

The application of the average reward reinforcement learning algorithms in the control were described in this paper. Moreover, new epoch-incremental reinforcement learning algorithm (EIR(0)-learning for short) was proposed. In this algorithm, the basic R(0)-learning algorithm was implemented in the incremental mode and the environment model was created. In the epoch mode, on the basis of the model, the distances of past active states to the terminal state were determined. These distances were then used in the update strategy. The proposed algorithm was applied to mountain car (Fig. 4) and ball-beam (Fig. 5) models. The proposed EIR(0)-learning was empirically compared to R(0)-learning [4, 6], R(λ)-learning and model based algorithms: Dyna-R and prioritized sweeping-R [11]. In the case of ball-beam system, EIR(0)-learning algorithm reached the stable control strategy after the smallest number of trials (Tab. 1, column 2). For the mountain car system, the number of trials was smaller than in the case of R(0)-learning and R(λ)-learning algorithms, but greater than for Dyna-R and prioritized sweeping-R. It is worth to pay attention to the fact that the execution times of Dyna-R and prioritized sweeping-R algorithms in the incremental mode were respectively 5 and 50 times longer than for proposed EIR(0)-learning algorithm (Tab. 2, column 3). The main conclusion of this work is that the epoch – incremental learning algorithm provided the stable control strategy in relatively small number of trials and in short time of single iteration.

Keywords: average reward reinforcement learning, R-learning, epoch-incremental reinforcement learning.

1. Wstęp

Celem uczenia ze wzmocnieniem jest pozyskanie strategii zachowania ucznia w pewnym nieznanym środowisku na podstawie interakcji z otoczeniem. Do chwili obecnej powstało szereg algorytmów uczenia się ze wzmocnieniem, które z powodzeniem realizują powyższą ideę. Podstawowe i najbardziej rozpowszechnione algorytmy Q-learning [1], AHC [2], czy Sarsa [3] dążą do maksymalizacji zdyskontowanej sumy wzmocnień otrzymywanych w długim horyzoncie czasowym. Rzadziej stosowane są algorytmy uczenia się ze wzmocnieniem działające w oparciu o kryterium średniego wzmocnienia. Przykładem mogą być tutaj algorytmy R-learning [4], czy h-learning i ah-learning [5]. Wymienione wcześniej algorytmy charakteryzują się dużą prostotą, która prowadzi do niewielkiej złożoności obliczeniowej, jednakże ich efektywność mierzona ilością prób niezbędnych do osiągnięcia stabilnej strategii optymalnej jest niewielka. Znaczną poprawę efektywności algorytmów uczenia się ze wzmocnieniem uzyskano poprzez wykorzystanie informacji o historii aktywności stanów reprezentowaną przez ich ślady aktywności (*eligibility traces*) [1, 6, 7].

Kolejnym etapem poszukiwania algorytmów pozwalających na osiągnięcie strategii optymalnej po mniejszej liczbie prób było zastosowanie modelu środowiska. Na uwagę zasługują tutaj dwa algorytmy: Dyna-Q [8] i prioritized sweeping [9, 10]. Ideą tych algorytmów było przeprowadzanie w każdej iteracji dodatkowo pewnej ustalonej liczby aktualizacji funkcji wartości akcji w oparciu o interakcje nie z rzeczywistym środowiskiem, lecz jego modelem. Stosunkowo proste adaptacje tych algorytmów do kryterium średniego wzmocnienia, w wyniku których powstały algorytmy Dyna-R i prioritized sweeping-R zostały przedstawione w pracy [11]. Wyniki przeprowadzonych eksperymentów porównawczych w środowisku gridowym pokazały, że tak zmodyfikowane algorytmy są wydajniejsze niż Dyna-Q i prioritized sweeping.

Innym podejściem pozwalającym na poprawę efektywności algorytmów uczenia się ze wzmocnieniem jest wykorzystanie modelu środowiska w trybie epokowym, tj. po zakończeniu pojedynczej próby. W pracy [12] zaprezentowano ideę algorytmu epokowo-inkrementacyjnego zbudowanego w oparciu o algorytm Q(0)-learning. Przeniesienie etapu modyfikacji strategii w oparciu o model środowiska do trybu epokowego pozwoliło na zmniejszenie czasu realizacji algorytmu w trybie inkrementacyjnym, co jest szczególnie istotne w przypadku zastosowania tych algorytmów do sterowania rzeczywistymi obiektami.

W niniejszym artykule zaproponowano algorytm epokowo-inkrementacyjny zbudowany w oparciu o algorytm R(0)-learning. Efektywność algorytmu zostanie zweryfikowana w sterowaniu modelem samochodu wjeżdżającego na wzniesienie (*mountain car*) oraz modelem kulki balansującej na belce (*ball-beam*). Efekty użycia nowego algorytmu zostaną porównane z wynikami zastosowania algorytmów R(0)-learning, R(λ)-learning, Dyna-R i prioritized sweeping-R.

2. Algorytm R-learning

Algorytm R-learning dąży do maksymalizacji średniego wzmocnienia ρ danego zależnością:

$$\rho = \lim_{n\to\infty} E\left[\frac{1}{n}\sum_{t=1}^{n} r_t\right] \tag{1}$$

w której r_t jest wzmocnieniem otrzymanym w chwili czasu t, zaś E oznacza operator wartości oczekiwanej. Funkcja wartości akcji Q dla każdej pary stan-akcja (s, a) definiowana jest względem średniej wartości oczekiwanego wzmocnienia w jednostce czasu, danego zależnością (1), jako:

$$Q(s_t, a_t) = \sum_{n=1}^{\infty} E\left\{r_{t+n} - \rho \middle| s_t = s, a_t = a\right\}. \tag{2}$$

Przyjmując, że proces jest ergodyczny (istnieje niezerowe prawdopodobieństwo osiągnięcia dowolnego stanu z innego dowolnego stanu, przy zastosowaniu dowolnej strategii), wówczas ρ nie zależy od stanu początkowego. Innymi słowy z dowolnego stanu w wystarczająco długim czasie osiągnięte średnie wzmocnienie jest takie samo.

Poza użyciem wartości względnej funkcji wartości akcji Q, R-learning jest typowym algorytmem różnic czasowych (TD - *Temporary Differences*) niezależnym od strategii (*off-policy*), w którym aktualizacja funkcji wartości akcji prowadzona będzie według reguły:

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \\ + \alpha(r - \rho + \max_a Q(s_{t+1},a) - Q(s_t,a_t)), \quad (3)$$

w której $\alpha \in (0,1]$ jest współczynnikiem uczenia funkcji wartości akcji Q. Średni błąd ρ będzie aktualizowany tylko w przypadku wykonania akcji zachłannej, czyli dla $a_t = \arg\max_a Q(s_t,a)$ poprzez zależność:

$$\rho \leftarrow \rho + \beta(r - \rho + \max_a Q(s_{t+1},a) - \max_a Q(s_t,a)), \quad (4)$$

gdzie $\beta \in (0,1]$ jest współczynnikiem aktualizacji średniego wzmocnienia oraz $\beta < \alpha$ [4, 6]. Zastosowanie śladów aktywności e do algorytmu R(0)-learning prowadzi do uzyskania jego przyśpieszonej wersji, oznaczanej jako R(λ)-learning, w której aktualizacji funkcji wartości akcji Q dokonuje się w każdej iteracji dla wszystkich elementów tablicy Q według zależności:

$$Q(s,a) \leftarrow Q(s,a) + \\ + \alpha(r - \rho + \max_a Q(s_{t+1},a) - Q(s_t,a_t)) \cdot e(s,a), \quad (5)$$

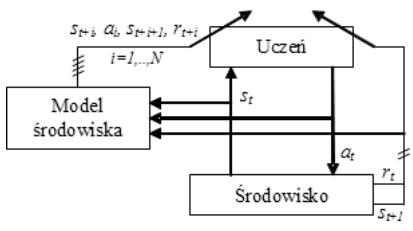
Aktualizację stanu aktywności przeprowadza się najczęściej w oparciu o zależność akumulacyjną [6]:

$$e(s,a) = \begin{cases} \lambda \cdot e(s,a) + 1 & \text{jezeli } s = s_t \text{ i } a = a_t \\ \lambda \cdot e(s,a) & \text{w przeciwnym przypadku} \end{cases}, \quad (6)$$

w której $\lambda \in [0,1]$ jest współczynnikiem świeżości.

3. Dyna-R i prioritized sweeping-R

Dyna-R i prioritized sweeping-R należą do klasy algorytmów uczenia się ze wzmocnieniem, które poprawę strategii decyzyjnej realizują w oparciu o model środowiska. Znajomość modelu środowiska pozwala oszacować na podstawie aktualnego stanu i wykonanej akcji następny stan, w którym znajdzie się środowisko oraz towarzyszącą temu przejściu wartość sygnału wzmocnienia. Dyna-R jest metodą uczenia się ze wzmocnieniem wykorzystującą algorytm R-learning.



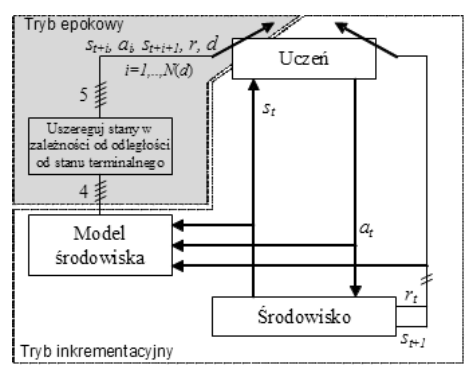
Rys. 1. Uczenie się ze wzmocnieniem z wykorzystaniem modelu środowiska
Fig. 1. Model based reinforcement learning

Algorytm Dyna-R składa się z dwóch głównych etapów (rys. 1). Etap pierwszy polega na wykorzystaniu interakcji ze środowiskiem zarówno do tworzenia jego modelu, jak i modyfikacji strategii sterowania przy pomocy zależności (3) i (4). Tworzenie modelu polega na zapamiętaniu kolejnych czwórek złożonych z akcji a_t wykonanej w stanie s_t, następnego stanu s_{t+1} oraz wartość sygnału wzmocnienia r_t. W drugim etapie ma miejsce modyfikacja strategii sterowania wyłącznie w oparciu o model. Wykonuje się wówczas pewną, ustaloną liczbę (oznaczaną jako N na rys. 1) aktualizacji funkcji wartości Q w oparciu o zgromadzony w modelu ciąg czwórek $<s_{t+i}, a_{t+i}, s_{t+i+1}, r_t>$ (zależności (3) i (4)). W algorytmie Dyna-R elementy modelu, w oparciu o które prowadzona jest aktualizacja strategii wybierane są losowo.

Algorytm *prioritized sweeping* jest zasadniczo podobny do Dyna-learning, poza dwoma istotnymi wyjątkami. Pierwszym jest brak aktualizacji funkcji wartości prowadzonej w oparciu o interakcje z rzeczywistym środowiskiem. Aktualizację funkcji wartości przeprowadza się wyłącznie w oparciu o model. Druga różnica koncentruje się na wyborze par stan-akcja, dla których mają być prowadzone aktualizacje funkcji wartości akcji. Inaczej niż w Dyna-R, tutaj są one prowadzone dla N par o największej wartości bezwzględnego błędu różnic czasowych algorytmu R-learning $|r_t - \rho + \max_a Q(s_{t+1},a) - \max_a Q(s_t,a)|$. Dobór liczby dodatkowych modyfikacji N podyktowany bywa kompromisem pomiędzy złożonością problemu sterowania a czasem, który może zostać przeznaczony na poprawę strategii sterowania [14].

4. EIR(0)-learning

Epokowo-inkrementacyjny algorytm uczenia się ze wzmocnieniem wykorzystujący metodę R(0)-learning (w skrócie EIR(0)-learning) wykonywany jest w dwu etapach (rys. 2). Etap pierwszy - inkrementacyjny - realizowany jest do chwili osiągnięcia stanu terminalnego, tzn. takiego, którego nie można opuścić. W trybie tym ma miejsce modyfikacja strategii ucznia za pomocą algorytmu R(0)-learning (zależności (3) i (4)) oraz tworzony jest model środowiska.



Rys. 2. Algorytm epokowo - inkrementacyjny uczenia się ze wzmocnieniem
Fig. 2. Epoch - incremental reinforcement learning algorithm

W etapie epokowym ma miejsce modyfikacja strategii ucznia z wykorzystaniem odległości d wszystkich odwiedzonych stanów od stanu terminalnego, którego osiągnięciu towarzyszy przyznanie sygnału wzmocnienia r.

Algorytm EIR(0)-learning w pseudokodzie przedstawiono na rys. 3. Rozpoczyna się on od inicjalizacji $Q(s,a)$ - tablicowej postaci funkcji wartości akcji oraz $N(s,a)$ będącej liczbą dyskretnych chwil czasu, w których system znajdując się w stanie s wykonał akcję a i $N(s,a,s')$ oznaczającej liczbę chwil czasu, w których po podjęciu akcji a w stanie s system znalazł się w stanie s' (krok 1). W każdym kroku trybu inkrementacyjnego ma miejsce wyznaczenie akcji a w oparciu zaobserwowany aktualny stan i bieżącą strategię reprezentowaną przez funkcję wartości akcji Q (krok 5 i 6). Po wykonaniu akcji generowany jest sygnał wzmocnienia r, a system przechodzi do kolejnego stanu s' (krok 7 i 8). Następnie przeprowadzana jest aktualizacja funkcji wartości akcji (3) i średniego wzmocnienia (4). W oparciu o iloraz aktualizowa-

nych $N(s,a,s')$ i $N(s,a)$ wyznaczane jest prawdopodobieństwo $p(s,a,s')$ podjęcia w stanie s akcji a i osiągnięcia stanu s' [9].

1: zeruj $Q(s,a)$, $N(s,a)$ i $N(s,a,s')$ dla wszystkich $s,s' \in S$ i $a \in A$ oraz kolejkę $PQueue$
2: powtarzaj (dla wszystkich epizodów):
3: inicjalizuj s
4: dla wszystkich kroków epizodu wykonuj:
5: obserwuj aktualny stan s
6: $a \leftarrow$ *wybierz-akcję*(s,Q)
7: wykonaj akcję a,
8: obserwuj wzmocnienie r i następny stan s'
9: $Q(s,a) \leftarrow Q(s,a) + \alpha[r - \rho + \max_{a'} Q(s',a') - Q(s,a)]$
10: jeżeli $Q(s,a) = \max_{a'} Q(s',a')$, to
11: $\rho \leftarrow \rho + \beta[r - \rho + \max_{a'} Q(s',a') - \max_{a} Q(s,a)]$
12: $N(s,a) \leftarrow N(s,a) + 1$; $N(s,a,s') \leftarrow N(s,a,s') + 1$
13: $p(s,a,s') \leftarrow N(s,a,s') / N(s,a)$
14: dopóki s' nie jest stanem absorbującym
15: $PQueue_0 \leftarrow (s,a,s',d=0)$
16: powtarzaj dopóki $PQueue$ nie jest pusta
17: $d \leftarrow d+1$
18: dla wszystkich $s' \in PQueue_{d-1}$
19: dla wszystkich $s \in preds(s')$ i $a \in A$ dla których $p(s,a,s') > 0$
20: $PQueue_d \leftarrow (s,a,s',d)$
21: $Q(s,a) \leftarrow Q(s,a) + \alpha(r\lambda^d - \rho + \max_{a'} Q(s',a') - Q(s,a))$
22: $\rho \leftarrow \rho + \beta[r - \rho + \max_{a'} Q(s',a') - \max_{a} Q(s,a)]$

Rys. 3. Algorytm EIR(0)-learning
Fig. 3. EIR(0)-learning algorithm

Część epokowa algorytmu rozpoczyna się po osiągnięciu przez system stanu terminalnego. W kroku 15 ma miejsce inicjalizacja kolejki *PQueue* czwórką złożoną ze stanu s, akcji a, stanu terminalnego s' i odległości d równej 0. Następnie wyznacza się te pary stan-akcja, które prowadzą do stanu s, przypisuje im odległość $d=d+1$ i umieszcza w kolejce *PQueue* zaś z kolejki usuwa się elementy będące następnikami dla nowo wstawionych (krok 19 i 20). Następnie przeprowadza się aktualizację funkcji wartości akcji (krok 21) dla wszystkich czwórek z kolejki *PQueue* według zależności:

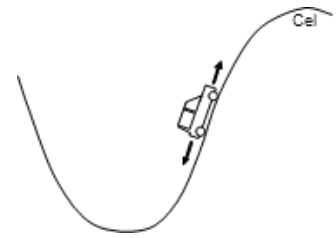
$$Q(s_{t+i},a_{t+i}) \leftarrow Q(s_{t+i},a_{t+i}) + \\ + \alpha(r\lambda^d - \rho + \max_{a'} Q(s_{t+i+1},a') - Q(s_{t+i},a_{t+i})) . \quad (7)$$

w której r jest sygnałem wzmocnienia towarzyszącym osiągnięciu stanu terminalnego. Składnik $r\lambda^d$ jest formą wykładniczo malejącej pamięci o fakcie, że dany stan przyczynił się do osiągnięcia stanu terminalnego. Inspiracją mnożenia sygnału wzmocnienia przez czynnik λ^d była obserwacja działania algorytmu TD(λ), w którym to ślad aktywności reguły w każdej iteracji był zmniejszany w analogiczny sposób (zależność (6)). Ponadto, o ile w metodzie TD(λ) wszystkie elementy funkcji wartości akcji ulegały aktualizacji w stopniu zależnym od odległości od aktualnego stanu, o tyle w przypadku algorytmu epokowo-inkrementacyjnego aktualizacji dokonuje się wyłącznie dla tych elementów funkcji wartości akcji, które są odpowiedzialne za strategię optymalną, wyznaczoną w oparciu o najkrótszą odległość od stanu terminalnego. Z tego też powodu aktualizację średniego wzmocnienia ρ przeprowadza się dla każdej czwórki z kolejki *PQueue*. Procedurę kontynuuje się do czasu aż nie opróżni się kolejki *PQueue*, co jest równoważne z aktualizacją funkcji wartości akcji dla wszystkich stanów aktywnych w trakcie pojedynczego epizodu.

Warto w tym miejscu zaznaczyć, że o ile w algorytmach Dyna-R oraz prioritized sweeping-R liczba aktualizacji przeprowadzana w oparciu o model w trybie inkrementacyjnym jest arbitralnie ustalana, o tyle w zaproponowanym algorytmie EIR(0)-learning jest ona funkcją liczby stanów aktywnych w trakcie epizodu, co upraszcza proces parametryzacji algorytmu.

5. Eksperymenty

Eksperymenty porównawcze przeprowadzono z zastosowaniem dwóch popularnych modeli obiektów sterowania, tj. samochodu górskiego [6] (rys. 4) oraz kulki umieszczonej na belce [13] (rys. 5). Efekty sterowania z wykorzystaniem algorytmu EIR(0)-learning porównano z efektami pracy algorytmów: R(0)-learning, R(λ)-learning oraz: Dyna-R i prioritized sweeping-R. Modele matematyczne obu obiektów opisują te same zmienne stanu, tj. [x (pozycja) i $\dot{x}$ (prędkość)].

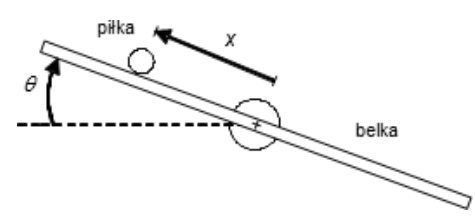


Rys. 4. Samochód górski
Fig. 4. Mountain car

W przypadku samochodu górskiego wartości zmiennych x i $\dot{x}$ zostały ograniczone odpowiednio do przedziału [-1.2; 0.5] i [-0.07; 0.07]. Celem układu sterującego jest doprowadzenie samochodu do punktu trajektorii x=0.5 oznaczonego jako „Cel” (rys. 4), zatem sygnał wzmocnienia przyjęto jako:

$$r = \begin{cases} 1 & \text{gdy } x \geq 0.5 \\ 0 & \text{w przeciwnym razie} \end{cases} . \quad (7)$$

Każdy eksperyment rozpoczynano w punkcie o wartościach zmiennych stanu $x_0 = -0.5$ i $\dot{x}_0 = 0$. Uniwersum poszczególnych zmiennych podzielono każdorazowo na 5 dyskretnych przedziałów następująco: x={[−1.2;-0.8) ∪ [−0.87;-0.52) ∪ [−0.52;−0.18) ∪ [−0.18;0.16) ∪ [0.16;0.5]} oraz $\dot{x}$ ={[−0.07;-0.042) ∪ [−0.42;−0.014) ∪ [−0.014;0.014) ∪ [0.014;0.042) ∪ [0.042;0.07]}. Zbiór akcji (sterowań) jest trójelementowy $A \in \{-1, 0,1\}$, co można interpretować, jako „jazda do tyłu”, „luz” i „jazda do przodu”.



Rys. 5. Kulka umieszczona na belce
Fig. 5. Ball-beam system

Dla układu kulki umieszczonej na belce o długości 2 m zamocowanej obrotowo w jej środku, zmienne stanu o uniwersum $x, \dot{x} \in [-1;1]$ podzielono na takie same przedziały następująco: $x, \dot{x}$ = {[−1;-0.5) ∪ [−0.5;-0.1) ∪ [−0.1;0.1) ∪ [0.1;0.5) ∪ [0.5;1]}. Eksperymenty rozpoczynano z nieruchomą kulką umieszczoną w środku belki, zatem $x_0 = 0$ i $\dot{x}_0 = 0$. Położenie kulki na belce ustala się poprzez zmianę kąta odchylenia belki od poziomu. Akcje pochodzą ze zbioru pięcioelementowego $A \in \{-\pi/4, -\pi/8, 0, \pi/8, \pi/4\}$. Celem uczenia się jest taka modyfikacja strategii, aby unikać niepożądanego stanu, jakim jest zsunię-

cie się kulki z belki. Jest to przykład zadania do porażki, zatem sygnał wzmocnienia ustalono jako:

$$r = \begin{cases} -1 & \text{gdy} \mid x \mid \geq 1 \\ 0 & \text{w przeciwnym razie} \end{cases}. \quad (8)$$

Dla każdego z obiektów otrzymano 5×5 = 25 elementów tablicy funkcji wartości akcji. Dla wszystkich testowanych algorytmów przyjęto ponadto α=0.1, β=0.001, zaś w algorytmach Dyna-R i prioritized sweeping-R liczbę eksperymentów przeprowadzanych w oparciu o model środowiska ustalono na 25. Wyboru akcji dokonywano za pomocą strategii ε-zachłannej, z ε=0.1. Liczba epizodów wynosiła 40, które powtarzano 30-to krotnie. Eksperymenty zrealizowano w środowisku Matlab na komputerze z procesorem Intel PentiumM 2.0GHz i pamięcią RAM 1 GB.

Wyniki eksperymentów zamieszczono w tabelach 1 i 2. Kolumna 2 tabeli 1 zawiera uśrednioną liczbę iteracji po której samochód wjeżdżał na wzniesienie. Najlepsze wyniki uzyskano kolejno dla algorytmu prioritized sweeping-R, następnie Dyna-R i dalej EIR(0)-learning, R(λ)-learning oraz R(0)-learning. Równocześnie, czas pojedynczej iteracji w algorytmie prioritized sweeping-R i Dyna-R był odpowiednio około 50 i 5 razy dłuższy, niż w przypadku pozostałych algorytmów.

Tab. 1. Wyniki eksperymentów dla samochodu górskiego
Tab. 1. The results of experiments for the mountain car

Algorytm	Liczba iteracji	Czas [ms]	
		inkrementacyjny	epokowy
R(0)-learning	1821.2	0.10998	0
R(λ)-learning	1803.4	0.11308	0
Dyna-R	1106.3	0.58136	0
prioritized sweeping-R	924.7	5.9738	0
EIR(0)-learning	1360.6	0.11169	1.8034

Tab. 2. Wyniki eksperymentów dla układu „kulki na belce”
Tab. 2. The results of experiments for the ball-beam system

Algorytm	Liczba prób	Czas [ms]	
		inkrementacyjny	epokowy
R(0)-learning	17.4	0.094495	0
R(λ)-learning	7.1	0.10291	0
Dyna-R	20.1	0.41011	0
prioritized sweeping-R	16.0	1.7353	0
EIR(0)-learning	4.3	0.095284	1.7143

Efektywność algorytmów w przypadku układu „kulki na belce” oceniono na podstawie liczby prób, po której układ sterujący „nauczył się” tak sterować belką, aby kulka nie spadała. Najmniejszą liczbę prób uzyskano w przypadku algorytmu EIR(0)-learning. Czas trybu inkrementacyjnego w tym algorytmie był porównywalny z algorytmami R(0)-learning i R(λ)-learning, zaś w algorytmach prioritized-sweeping-R i Dyna-R odpowiednio około 17 i 4 razy dłuższy. Ponadto można zauważyć, że w przypadku tego typu zadania („do porażki”) algorytmy te nie dały tak spektakularnych rezultatów, jak w przypadku poprzedniego obiektu, co może sugerować, że są one predysponowane do zadań typu „do sukcesu”.

Tryb epokowy był wykonywany po zakończeniu każdego epizodu. Czas wykonania trybu epokowego był około 17 razy dłuższy, niż trybu inkrementacyjnego

6. Podsumowanie

Algorytmy wykorzystujące model środowiska w trybie inkrementacyjnym, tj. prioritized sweeping-R oraz Dyna-R poprawiły wydajność algorytmu uczenia ze wzmocnieniem jedynie w przypadku zadania typu „do sukcesu”. Dobre wyniki tych algorytmów zostały jednak okupione znacznym dłuższym czasem ich wykonania, który był odpowiednio kilkadziesiąt i kilka razy dłuższy, niż dla pozostałych algorytmów. Algorytm epokowo-inkrementacyjny charakteryzuje się krótkim czasem trybu inkrementacyjnego, typowym dla algorytmów podstawowych (R(0)-learning, R(λ)-learning). Wydajność tego algorytmu była niewiele gorsza w przypadku zadania „do sukcesu” w porównaniu do algorytmów wykorzystujących model środowiska. Dla zadania typu „do porażki” zaproponowany algorytm okazał się najlepszy.

Praca realizowana w ramach grantu MNiSW nr 3745/B/T02/2009/36.

7. Literatura

[1] Watkins, C.J.C.H.: Learning from delayed Rewards. PhD thesis, Cambridge University, Cambridge, England, 1989.
[2] Barto A., Sutton R., Anderson C.: Neuronlike adaptive elements that can solve difficult learning problem, IEEE Trans. SMC, 13, pp. 834-847, 1983.
[3] Rummery G., Niranjan M.: On line q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166, Cambridge University Engineering Department, 1994.
[4] Schwartz A.: A reinforcement learning method for maximizing undiscounted rewards, Proc. of Tenth International Conference on Machine Learning, Amhest, Massachusetts. Morgan Kaufman, 298-305, 1993.
[5] Tadepalli P., Ok D.: Model-Based Average Reward Reinforcement Learning. Artificial Intelligence, 100, pp. 177-224, 1998.
[6] Sutton R., Barto A.: Reinforcement learning: An Introduction, MIT Press, Cambridge, 1998.
[7] Cichosz P.: Systemy uczące się. WNT, Warszawa, 2000.
[8] Sutton R.: Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming. In Proc. of Seventh Int. Conf. on Machine Learning, pp. 216-224, 1990.
[9] Moore A., Atkeson C.: Prioritized sweeping: Reinforcement learning with less data and less time. Machine Learning, 13, pp. 103-130, 1993.
[10] Peng J., Williams R.: Efficient learning and planning within the Dyna framework. In Proc. of the 2nd International Conference on Simulation of Adaptive Behavior, pp. 281-290, 1993.
[11] Zajdel R., Algorytmy uczenia ze wzmocnieniem Dyna-R i prioritized sweeping-R, Inżynieria wiedzy i systemy ekspertowe, Akademicka Oficyna Wydawnicza EXIT, Warszawa 2009, 161-169.
[12] Zajdel R., Epoch-Incremental Queue-Dyna Algorithm, The Ninth International Conference on Artificial Intelligence and Soft Computing, Zakopane, Lecture Notes in Artificial Intelligence 5097, 1160-1170, 2008.
[13] Wellstead, P.E.: Introduction to Physical System Modelling, Control Systems Principles, 2000.
[14] Kaelbing L.P., Litman, M.L., Moore, A.W., (1996). Reinforcement Learning: A Survey, Journal of Artificial Intelligence Research 4, 237–285.

otrzymano / received: 12.04.2013
przyjęto do druku / accepted: 03.06.2013 *artykuł recenzowany / revised paper*