Lokalizácia podstratégií modelu HABS

Viliam Dillinger

Katedra aplikovanej informatiky, FMFI, Univerzita Komenského
Mlynská dolina, 842 48 Bratislava
Email: viliam.dillinger@fmph.uniba.sk

Abstrakt

V príspevku sa venujeme úpravám modelu HABS (Hierarchické priraďovanie správaní pomocou samoorganizácie), so snahou o stabilizácii a zrýchleniu jeho konvergencie. K naším predchádzajúcim zmenám (Dillinger, 2014), ktoré zlepšili stabilitu modelu, sme pridali ďalšie, vďaka ktorým model vždy skonverguje. Okrem toho sme navrhli úpravu v štruktúre vstupov, takzvanú lokalizáciu podstratégií, čo urýchlilo konvergenciu modelu. Rýchlosť konvergencie modelu HABS, ako aj vplyv lokalizácie podstratégií experimentálne porovnávame s agentmi učiacimi sa posilňovaním vo vytvorenom testovacom prostredí.

1 Úvod

Medzi významné oblasti strojového učenia patrí učenie posilňovaním (reinforcement learning, RL). RL agent sa učí plánovať svoje akcie pomocou priamej interakcie s prostredím. Z hľadiska riadenia RL zahŕňa online aproximáciu riešenia stochastického problému optimálneho riadenia, väčšinou v podmienkach bez úplnej znalosti prostredia.

S narastajúcou veľkosťou prostredia a počtom parametrov jeho stavov sa v RL objavujú mnohé problémy. Medzi hlavné patria prekliatie dimenzionality, neefektívne šírenie odmeny, slabý prenos vedomostí a zdĺhavá náhodná chôdza (Barto a Mahadevan, 2003). Na tieto problémy sa snažia odpovedať modely hierarchického učenie posilňovaním (HRL) pomocou využitia rôznych hierarchických štruktúr a časovo rozšírených akcií.

Medzi modely HRL patrí aj model HABS (Moerman, 2009) (Hierarchické priraďovanie správaní pomocou samoorganizácie), ktorého úpravou sa zaoberáme v tomto príspevku s cieľom stabilizovať a zrýchliť jeho konvergenciu.

Tento príspevok je radený nasledovne. V druhej časti definujeme Semi-Markovov rozhodovací proces a následne v tretej časti pomocou neho formalizujeme testovacie prostredie, v ktorom budeme porovnávať jednotlivých agentov. Štvrtá časť obsahuje popis klasického RL. V piatej časti popisujeme funkčné aproximátory a porovnanie klasických RL agentov, ktorí ich využívajú. Šiesta časť obsahuje popis modelu HABS, naších úprav zameraných na jeho stabilizáciu a jeho porovnanie s klasickým RL. V Siedmej časti popisujeme lokalizáciu podstratégií modelu HABS ako nástroj na zrýchlenie konvergencie tohoto modelu.

2 Semi-Markovov rozhodovací proces

Semi-Markovov rozhodovací proces (Russell a Norvig, 2003) (Markov decision process, SMDP) je matematický model rozhodovania. Agent vyberá svoje akcie len na základe aktuálneho stavu. Jeho jedinou spätnou väzbou je odmena (trest), ktorú môže, ale nemusí dostať pri každom vykonaní akcie. Agent nemá spätnú väzbu o tom, čo spôsobilo zisk odmeny (trestu) a teda jednou z jeho úloh je zistiť postupnosť akcií, ktorá k nim vedie. Medzi najťažšie úlohy patria tie, pri ktorých agent dostáva odmenu len pri úspešnom, respektíve trest pri neúspešnom splnení úlohy.

Formálne, SMDP je štvorica $\langle S, A, P, R\rangle$ kde

- S je konečná množina stavov, v ktorých sa agent môže nachádzať,
- A je konečná množina akcií. V stave s agent môže vykonať akciu z množiny A_s, $A = \bigcup_{s\in S} A_s$;
- P je funkcia pravdepodobnostnej distribúcie nasledujúcich stavov. Ak agent vykoná akciu a v stave s, stav prostredia sa zmení na s' za τ časových jednotiek s pravdepodobnosťou $P(s', \tau|s, a)$;
- R je funkcia odmeny. Po vykonaní akcie a v stave s a presunu do stavu s' za τ časových jednotiek, agent dostane odmenu $R(s', \tau|s, a)$;

Výber akcií agenta v jednotlivých stavov porstredia popisuje stratégia (policy) $a = \pi(s), s \in S, a \in A_s$. Úlohou agenta je nájsť optimálnu stratégiu, ktorá maximalizuje sumu budúcich diskontovaných odmien v každom stave.

$$R_t = \sum_{n=0}^{\infty} \gamma^n r_{t+n}$$

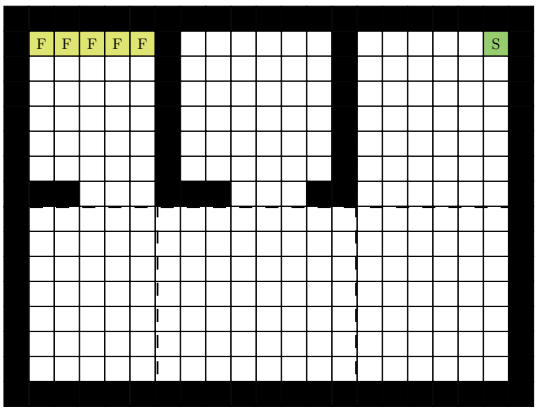
kde $\gamma \in (0, 1)$ je diskontný faktor.

3 Testovacie prostredie

Na porovnávanie rýchlosti konvergencie modelov sme vytvorili prostredie znázornené na obrázku 1. Stav prostredia je reprezentovaný pozíciou agenta (jeho x-ovou

a y-ovou súradnicou). Agent môže vykonať 4 akcie - krok smerom na sever, východ, juh a západ. Ak sa v danom smere nachádza stena (v obrázku čierne políčko), jeho pozícia sa nezmení. Agent začína epochu v počiatočnom stave, ktorý sa nachádza v pravom hornom rohu (označený S) a epocha končí prechodom agenta do jedného z finálnych stavov v ľavom hornom rohu (označené F). Agent dostane odmenu iba pri prechode do finálneho stavu. Najkratšia cesta z štartového stavu do finálneho má dĺžku 28 krokov.

Prostredie je rozdelené na 7 abstraktných stavov (pre potreby modelu HABS, pozri kapitolu 6). Na obrázku sú ich hranice zobrazené pomocou prerušovanej čiary.



Obr. 1: Testovacie prostredie. V pravom hornom rohu (označený S) sa nachádza počiatočný stav, v ľavom hornom rohu (označené F) sa nachádzajú finálne stavy. Prerušovanými čiarami sú označené hranice medzi abstraktnými stavmi.

4 Učenie posilňovaním

Jedným zo spôsobov, ako nájsť optimálnu stratégiu, je pomocou učenia posilňovaním. RL rieši úlohu pomocou aproximácie Q-funkcie $Q(s, a)$ (action-value function), ktorá odhaduje budúce odmeny pri výbere akcie a v stave s. Táto funkcia sa upravuje pomocou učiaceho pravidla pri každom vykonaní akcie na základe predchádzajúceho stavu, získanej odmeny pri aktuálnom kroku a nasledujúceho stavu.

Ak je získaná odmena vysoká alebo sa vďaka nej agent dostal do stavu, v ktorom je možné vykonať akciu s vysokým odhadom budúcich odmien, posilní práve vykonanú akciu. Naopak, ak agent získal negatívnu odmenu (trest) alebo sa dostal do stavu, kde sú odhady budúcich odmien všetkých akcií nízke, oslabí práve vykonanú akciu.

Tieto úpravy sa vykonávajú pozdĺž a v okolí trénovacích trajektórií v prostredí. Trajektórie môžeme získať pomocou simulácií, prípadne z priamej interakcie s porstredím. Takto získaná Q-funkcia bude vedieť dobre aproximovať v často navštevovaných, a teda relevantných stavoch.

4.1 Učiace pravidlá

Medzi najčastejšie používané učiace pravidlá v RL patrí Q-učenie. Ak sa agent nachádza v stave s, vykoná akciu a, dostane sa do stavu s' za τ časových jednotiek, pričom získa odmenu r, upraví svoju Q-funkciu pomocou vzťahu

$$Q(s,a) = (1-\alpha_t)Q(s,a) + \alpha_t \left(r + \gamma^\tau \max_{a' \in A_{s'}} Q(s', a') \right)$$

$$r = r_{t+1} + \gamma r_{\tau+2} + \ldots + \gamma^{t+\tau} r_{t+\tau}$$

kde $\alpha \in (0, 1)$ je rýchlosť učenia.

Ak majú susediace stavy v prostredí podobné maximálne Q-hodnoty, aj drobné chyby v aproximácii Q-funkcie môžu mať za následok veľké zmeny v stratégii agenta. Tento problém rieši učenie pomocou zvýhodňovania (Advantage learning) (Baird III, 1999), ktoré rozširuje učiace pravidlo Q-učenia na

$$Q(s,a) = (1-\alpha_t)Q(s,a) + \alpha_t \left(\max_{a \in A_s} Q(s,a) + \frac{r + \gamma^\tau \max_{a' \in A_{s'}} Q(s', a') - \max_{a \in A_s} Q(s,a)}{k\tau} \right)$$

kde $\alpha \in (0, 1)$ je rýchlosť učenia a $k \in (0, 1)$ je škálovací faktor.

Pri použití tohoto pravidla ostáva hodnota optimálnej akcie v danom stave rovnaká, avšak všetky ostatné akcie sú $(1/k\tau)$-krát viac vzdialené od Q-hodnoty optimálnej akcie ako pri Q-učení. To má za následok rýchlejšiu konvergenciu a menšiu náchylnosť na chyby.

4.2 Explorácia prostredia

Jednou z hlavných výhod RL je, že nepotrebujeme poznať zákonitosti prostredia (funkcie P a R), čo umožňuje využívať RL aj v reálnom prostredí. To má, naopak, za následok neschopnosť agenta rozoznať, či je aktuálne naučená stratégia optimálna, alebo nie. Agent preto potrebuje neustále skúmať (explorovať) prostredie, teda niekedy vykonať suboptimálnu akciu, čím môže nájsť lepšie riešenie.

Medzi najrozšírenejšie metódy explorácie patrí Boltzmannova explorácia, pri ktorej pravdepodobnosť výberu akcie je úmerná jej Q-hodnote vzhľadom na Q-hodnoty ostatných akcií v danom stave. Pravdepodobnosť, že v stave s vyberie akciu a je

$$P(s,a) = \frac{\exp(Q(s,a)/\lambda)}{\sum_{a' \in A_s} \exp(Q(s,a')/\lambda)}$$

kde λ je teplotný parameter, ktorý kvantifikuje mieru explorácie.

5 Funkčné aproximátory

Na reprezentáciu Q-funkcie sa v RL najčastejšie používajú funkčné aproximátory. Okrem šetrenia pamäte ponúkajú aj možnosť prenosu vedomostí do neznámych stavov, čo môže urýchliť konvergenciu Q-funkcie. Pre každú akciu je vytvorený jeden funkčný aproximátor, ktorého vstupom je zvolený stav prostredia a výstupom odhadovaná Q-hodnota akcie. V práci používame ako funkčné aproximátory neurónové siete, konkrétne viacvrstvový perceptrón a RBF siete.

5.1 Viacvrstvový perceptrón

Viacvrstvový perceptrón (multi-layer perceptron, MLP) sa skladá zo vstupnej vrstvy ($\mathbf{x}$), výstupnej vrstvy ($\mathbf{y}$), minimálne jednej skrytej vrstvy ($\mathbf{h}$) a synaptických prepojení medzi nimi ($\mathbf{V}$, $\mathbf{W}$). Počet neurónov na skrytých vrstvách je premenlivý a určuje výpočtovú zložitosť modelu. V našich experimentoch sme používali MLP s jednou skrytou vrstvou.

Učenie MLP sa realizuje pomocou úprav váh synaptických prepojení pomocou učiteľa, teda na základe rozdielu výstupu siete a požadovaným výstupom ($\mathbf{d}$). Výstup siete vyrátame ako

$$\mathbf{h} = f(\mathbf{V}\mathbf{x}) \qquad f(net) = \frac{1}{1+e^{-net}}$$
$$\mathbf{y} = \mathbf{W}\mathbf{h}$$

kde $f(net)$ je aktivačná funkcia (v našom prípade sigmoid).

Učenie siete prebieha nasledovne: najprv sa vyráta chyba neurónov výstupnej vrstvy ($\boldsymbol{\delta}_y$) a pomocou jej spätného šírenia (backpropagation, BP) získame chybu na skrytej vrstve ($\boldsymbol{\delta}_h$).

$$\boldsymbol{\delta}_y = \mathbf{d} - \mathbf{y}$$
$$\boldsymbol{\delta}_h = \mathbf{W}^T\boldsymbol{\delta}_y \,.*\, \mathbf{h} \,.*\, (1-\mathbf{h})$$

kde $.*$ je vektorové násobenie po prvkoch a $\mathbf{h} \,.*\, (1-\mathbf{h})$ je derivácia vybranej aktivačnej funkcie. Pomocou získaných chýb a aktivácií vrstiev upravíme synaptické prepojenia

$$\mathbf{W} = \mathbf{W} + \alpha\boldsymbol{\delta}_y\mathbf{h}^T$$
$$\mathbf{V} = \mathbf{V} + \alpha\boldsymbol{\delta}_h\mathbf{x}^T$$

5.2 RBF siete

Siete s radiálnymi bázovými funkciami (radial-basis function, RBF) sú, čo sa týka štruktúry, podobné ako MLP s jednou skrytou vrstvou. Každý z RBF neurónov na skrytej vrstve má svoj stred (vektor rovnakej veľkosti ako je vsutp) a jeho aktiváciu vyrátame podľa vzťahu

$$h_i = \exp(-\sigma||\mathbf{x}-\mathbf{c}_i||^2)$$
$$\mathbf{y} = \mathbf{W}\mathbf{h}$$

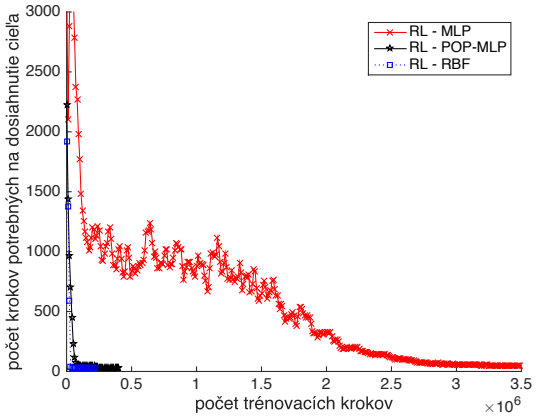
Učenie prebieha rovnako ako v prípade MLP, avšak menia sa iba synaptické prepojenia medzi skrytou a výstupnou vrstvou. Stredy sú určené pri vytvorení siete a počas trénovania sa nemenia.

Zásadný rozdiel oproti MLP je v lokálnosti správania. Ak RBF sieť učíme s určitým vstupom, zmena správania bude lokálna. Pri MLP by trénovanie s rovnakým vstupom ovplyvnilo výstup siete pri akomkoľvek vstupe. Ak sa teda agent dlhodobo „zasek-ne" v istej časti prostredia, môže sa odučiť správanie v iných častiach prostredia. V prípade RBF sietí by tento problém nenastal.

Naopak, RBF siete podliehajú prekliatiu dimenzionality, teda s narastajúcim počtom parametrov stavov prostredia exponenciálne rastie výpočtová zložitosť siete z dôvodu nutnosti väčšieho počtu neurónov na skrytej vrstve.

5.3 Porovnanie funkčných aproximátorov

V prvom experimente sme porovnávali rýchlosť konvergencie klasického RL agenta trénovaného pomocou učenia zvýhodňovaním s troma rôznymi funkčnými aproximátormi na reprezentáciu Q-funkcie. Konkrétne išlo o MLP, MLP s pozičným kódovaním vstupu a RBF sieť. Pozičné kódovanie poskytuje čiastočnú lokálnosť správania MLP, avšak len s lineárnym nárastom zložitosti s narastajúcim počtom parametrov prostredia (Sutton, 1996). Výsledky experimentu zobrazuje obrázok 2.



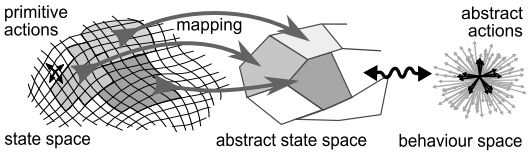
Obr. 2: Porovnanie priemernej rýchlosti konvergencie klasického RL agenta podľa použitého funkčného aproximátora.

Najrýchlejšiu konvergenciu z porovnávaných funkčných aproximátorov dosiahla RBF sieť. MLP s pozičným kódovaním vstupu bol v priemere približne 5-násobne a MLP až 70-násobne pomalší ako RBF sieť.

6 Model HABS

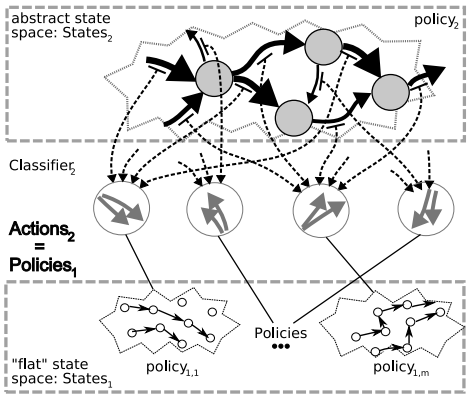
Model HABS pozostáva z dvoch vrstiev. Vrchná vrstva obsahuje stratégiu, ktorá operuje nad abstraktným stavovým priestorom vytvoreným programátorom. Abstraktný

stav združuje niekoľko kompaktných stavov prostredia a predpokladá sa istá geometrická štruktúra prostredia. Nižšia vrstva obsahuje zvolený počet podstratégií. Tieto podstratégie sú volané stratégiou ako podprogramy a realizujú prechody medzi abstraktnými stavmi (abstraktné akcie). Podstratégia vykonáva primitívne akcie, až kým nenastane zmena abstraktného stavu, alebo kým jej nevyprší časový limit.



Obr. 3: Abstraktný stavový priestor a abstraktné akcie. Prevzaté od Moerman (2009).

Správanie podstratégie popisuje jej vektor správania. Vektory správania sa počas učenia samoorganizujú tak, aby ich smery popisovali čo najviac smerov prechodov medzi abstraktnými stavmi. Ak podstratégia vykoná prechod medzi abstraktnými stavmi, je odmenená iba vtedy, keď vektor správania tohoto prechodu je klasifikovaný ako jej vektor správania. V opačnom prípade je potrestaná. Stratégia je trénovaná pomocou odmien naakumulovaných počas vykonávania vybranej podstratégie.



Obr. 4: Hierarchická štruktúra HABS. Prevzaté od Moerman (2009).

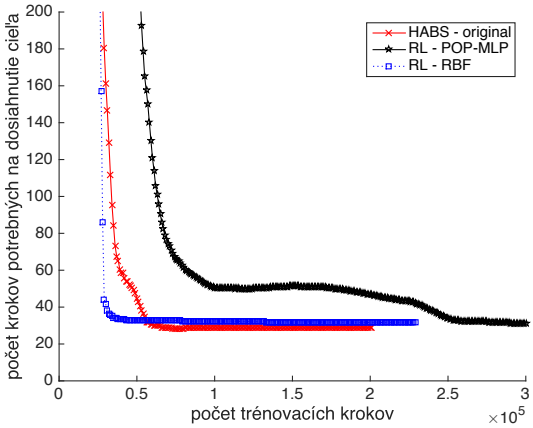
6.1 Stabilizácia konvergencie modelu

Kvôli snahe o lepšie šírenie odmeny a prehľadávanie priestoru má originálny model problémy s učením a počas experimentov vo väčšine prípadov neskonvergoval. Na zmiernenie problému sme navrhli niekoľko drobných úprav, ktoré tento problém minimalizovali. Konkrétne išlo o odstránenie odmeny podstratégie pri vypršaní časového limitu, použitia smerového vektora miesto vektora presunu ako vektora správania a zohľadňovanie trvania akcie pri učení stratégie (Dillinger, 2014).

V pôvodnom algoritme sa z dôvodu lepšieho využitia informácie trénovala akcia, ktorá bola na základe klasifikácie vykonaná, a nie akcia, ktorá bola vybraná, čo v má špecifických prostrediach za následok zacyklenie správania agenta a teda neúspešnú konvergenciu. Ak stratégia zavolá podstratégiu v stave, v ktorom nie je možné vykonať zmenu stavu podľa jej vektora správania, jej vykonávanie skončí s veľkou pravdepodobnosťou kvôli časovému limitu a tento presun bude klasifikovaný ako vektor správania inej akcie. Ak má vybraná podstratégia vysoký odhad budúcich odmien a všetky ostatné podstratégie majú tento odhad veľmi nízky, tento stav sa môže ešte prehlbovať, čo vedie k spomínanému zacykleniu. Preto sme toto učenie upravili a ak podstratégia ukončí svoje vykonávanie z dôvodu vypršania časového limitu, stratégia trénuje zvolenú abstraktnú akciu.

6.2 Porovnanie modelu HABS a klasického učenia posilňovaním

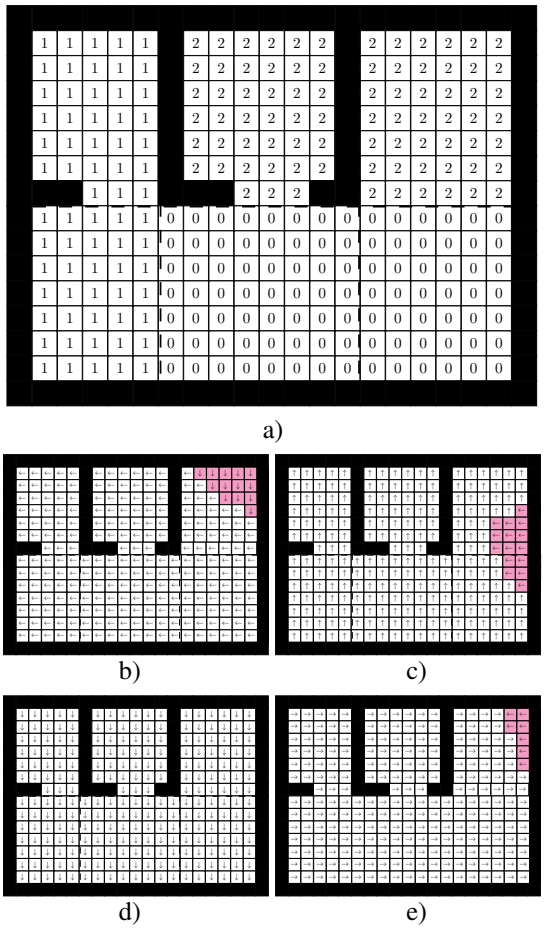
V druhom experimente sme porovnávali rýchlosť konvergencie modelu HABS obsahujúceho naše úpravy s klasickými RL agentmi z časti 5.3. Model obsahoval 4 podstratégie a jednu stratégiu, pričom všetky používali MLP ako funkčný aproximátor. Výsledky experimentu zobrazuje obrázok 5.



Obr. 5: Porovnanie priemernej rýchlosti konvergencie klasického RL agenta s rôznymi funkčnými aproximátormi a modelu HABS.

Model HABS s našími zmenami úspešne skonvergoval vo všetkých testoch a v priemere bol približne 1.4-krát pomaší ako RBF sieť. Príklad nájdenej stratégie a jej podstratégií vizualizuje obrázok 6.

Ak zložité prostredie obsahuje jednoduchú dynamiku, model HABS ju vie nájst a pomocou nej vie aj bez použitia lokálnych funkčných aproximátorov dosiahnuť veľmi dobré výsledky. Táto vlastnosť zároveň pomáha pri prieskume prostredia a ponúka priame využitie nadobudnutých schpností, čím sa zlepší prenos vedomostí.

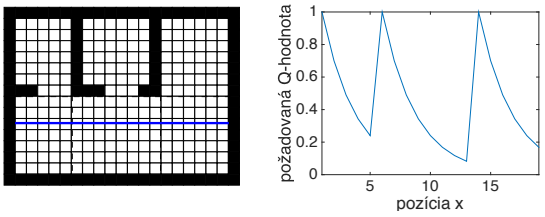


Obr. 6: Príklad úspešne natrénovanej stratégie a podstratégii modelu HABS. a) Stratégia. Čísla reprezentujú číslo akcie s najväčšou Q-hodnotou. b–d) Podstratégie. Šípky znázorňujú smer akcie s najväčšou Q-hodnotou, miesta kde sú stratégie v nesúlade s ich vektorom správania sú farebne označené

7 Lokalizované podstratégie modelu HABS

Ak má podstratégia ako svoj vstup priamy vstup z prostredia, na rozhraní dvoch abstraktných stavov vzniká veľký skok v Q-hodnotách akcie, ktorá spôsobí prechod medzi nimi. Jej Q-hodnota pred prechodom bude extrémna (či pri správnom alebo nesprávnom prechode) a v stave, do ktorého sa dostane, bude blízka 0. Keďže väčšina funkčných aproximátorov (vrátane MLP a RBF siete) sa snaží aproximovať funkciu pri čo najmenšej derivácii, tento skok spomalí konvergenciu. Príklad v testovacom prostredí zobrazuje obrázok 7.

Ako riešenie tohoto problému sme navrhli lokalizované podstratégie, ktorých vstup je relatívna pozícia vzhľadom na stred abstraktného stavu, v ktorom sa agent nachádza. Stredy abstraktných stavov sú určené programátorom spolu so samotnými abstraktnými stavmi. V našom testovacom prostredí sme ako stredy použili ťa-



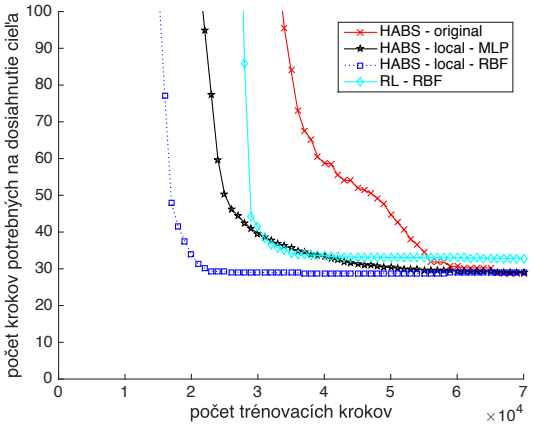
Obr. 7: Príklad skoku hodnôt trénovanej Q-funkcie. a) Rez priestorom b) Q-hodnoty podstratégie s vektorom správania smerujúcim na západ pre akciu krok na západ na zvolenom reze priestorom.

žiská jednotlivých abstraktných stavov.

Okrem odstránenia skokov táto zmena spôsobí aj lepší prenos vedomostí, pretože ako je vidno z obrázku 6, podstratégie bez lokalizácie majú bezchybné správanie, teda správanie podľa ich vektora správania, iba v stavoch, v ktorých sú využívané stratégiou. Ak by bolo potrebné, aby podstratégie reagovali mierne inak v rôznych abstraktných stavoch, je možné na vstup pridať stred aktuálneho abstraktného stavu.

7.1 Vplyv lokalizovaných podstratégií na rýchlosť konvergencie

V poslednom experimente sme porovnávali rýchlosť konvergencie dvoch modelov HABS s lokalizovanými podstratégiami a ostatných modelov spomínaných v tomto článku. Prvý model s lokalizovanými príznakmi využíval MLP v oboch vrstvách, druhý obsahoval RBF sieť pre stratégiu a a MLP ako funkčný aproximátor podstratégií. Výsledky experimentu zobrazuje obrázok 8.



Obr. 8: Porovnanie priemernej rýchlosti konvergencie klasického RL agenta s RBF sieťou, modelu HABS, modelu HABS s lokalizovanými podstratégiami s MLP na oboch vrstvách a modelu HABS s lokalizovanými podstratégiami s RBF sieťou na vyššej vrstve a MLP na nižšej.

Z tretieho experimentu vyplýva, že lokalizácia podstratégií mala pozitívny vplyv na rýchlosť konvergencie

a takto upravený model HABS dokázal skonvergovať rovnako rýchlo ako klasický RL agent s RBF sieťou. Ak sa však využila RBF sieť na stratégiu, dokázal náš model skonvergovať 2-krát rýchlejšie ako RL agent s RBF sieťou. Náš agent potreboval iba približne 30 epizód na naučenie sa optimálnej stratégie. Naučené lokalizované podstratégie, na rozdiel od nelokalizovaných, v každom stave zvolili správnu akciu vzhľadom na ich vektor správania.

8 Záver

V príspevku sme sa zaoberali úpravami modelu HABS. Pomocou naších predchádzajúcich úprav (Dillinger, 2014) a úpravy učenia stratégie pri vypršaní časového limitu podstratégie sme stabilizovali model natoľko, že vo všetkých experimentoch vždy skonvergoval.

Ďalej sme navrhli lokalizáciu podstratégií, kedy vstupom podstratégie je relatívna pozícia vzhľadom na stred aktuálneho abstraktného stavu určeného programátorom. Táto úprava značne zrýchlila konvergenciu modelu.

Model HABS, ako aj jeho lokalizáciu podstratégií, sme experimentálne porovnali s klasickými RL agentmi využívajúcimi rôzne funkčné aproximátory. Najrýchlejší bol model HABS s lokalizáciou podstratégií, stratégiou s RBF sieťou a podstratégiami s MLP. Agent s týmto modelom dokázal v testovacom prostredí skonvergovať za približne 22000 krokov, respektíve behom 30 epoch, čo pri optimálnej dĺžke cesty 28 považujeme za veľmi dobrý výsledok.

Poďakovanie

Tento príspevok bol podporený grantom VEGA 1/0898/14.

Literatúra

Baird III, L. C. (1999). *Reinforcement Learning Through Gradient Descent*. Dizertačná práca, Carnegie Mellon University Pittsburgh.

Barto, A. G. a Mahadevan, S. (2003). Recent advances in hierarchical reinforcement learning. *In Discrete-Event Dynamic Systems: Theory and Applications*, 13:343–379.

Dillinger, V. (2014). Učenie posilňovaním: hierarchia v rozhodovaní alebo vo funkčnej aproximácii? V *Kognitivní věda a umělý život II*, str. 45–51. Slezská univerzita v Opavě.

Haykin, S. (1999). *Neural networks: A Comprehensive Foundation*. Prentice Hall International, 2nd. vyd.

Moerman, W. (2009). *Hierarchical Reinforcement Learning: Assignment of Behaviours to Subpolicies by Self-Organization*. Dizertačná práca, Utrecht University.

Russell, S. a Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*. Prentice-Hall, 2. vyd.

Sutton, R. S. (1996). Generalization in reinforcement learning: Successful examples using sparse coarse coding. V *Advances in NIPS 8*, str. 1038–1044. MIT Press.