Robotická manipulace pomocí sekvence neurálních modulů s vlastní policy

Michal Vavrečka, Jonas Kříž, Nikita Sokovnin, Gabriela Šejnová

CIIRC ČVUT
Jugoslávských Partyzánů 3, Praha
michal.vavrecka@cvut.cz

Abstrakt

Vyvinuli jsme nový multi-policy algoritmus (MultiPPO2), který zlepšuje učení posilováním v úlohách, kde jsou vyžadovány různé dílčí cíle (dovednosti). Algoritmy s jednou policy často uspějí pouze v úlohách vyžadujících osvojení podobných nerůznorodých dovedností. Vzhledem k tomu, že většina současných robotických testů je založena na přemisťování objektů ve scéně, řeší algoritmy s jednou policy tyto úlohy s vysokou úspěšností. Pokud je vyžadována posloupnost různorodých dílčích cílů (translace, rotace, manipulace v 6DOF, sledování trajektorie), síť s jednou policy selže. Navrhujeme algoritmus s více policy pro každou dovednost. Je schopen naučit se vícekrokovou manipulaci s různými dílčími cíli. Náš algoritmus jsme testovali ve virtuálním robotickém simulátoru jak na jednokrokových i vícekrokových úlohách vyžadujících nerůznorodé (Pick and Place) i různorodé dovednosti (Pick and Rotate a Swipe). Náš multi-policy algoritmus dosáhl podobného výkonu v případě nerůznorodých dovedností a překonal single-policy algoritmy v úlohách vyžadujících různorodou sadu dovedností. Výsledek dokazuje, že algoritmus s více policy nabízí lepší škálovatelnost pro složité úlohy. Potenciál nabízí v kombinaci s hierarchickým posilovacím učením.

1 Úvod

S využitím posilovaného učení pro robotické manipulační úlohy je spojeno několik klíčových problémů. Jedním z nich je, že robot se musí naučit provádět úkoly zahrnující rozsáhlé sekvence akcí, jako je například "pick-and-place". Použití tradičních metod RL pro učení složité úlohy a single-policy je neefektivní, protože vyžaduje, aby agent prozkoumal rozsáhlý prostor stavů a akcí a shromáždil mnoho dat, než obdrží smysluplný učební signál.

V porovnání s jednokrokovými úlohami vyžadují komplexní úlohy ve většině reálných problémů více transformací k dosažení cílového stavu. Tento problém lze odstranit použitím algoritmů s několika policy. Ty jsou založeny na modulech odpovídajících základním schopnostem (dovednostem) řešit komplexní úlohy Andreas a spol. (2017). Takové policy netrpí zapomínáním a nabízejí lepší interpretovatelnost Alet a spol. (2018).

Devine a spol. použili modulární neuronové sítě, kde každý modul představuje samostatné dovednosti Devin a spol. (2017). Modulární sítě jsou sítě s více policy, které lze kompozičně řetězit, takže systém je schopen řešit složitější úlohy. Pore a kol. Pore a Aragon-Camarasa (2020) používají behaviorální klonování k trénování jednotlivých modulů dílčích úloh a hierarchicky vyšší modul, který je spojuje dohromady. Nevyřešeným problémem je způsob řazení modulů.

Někteří autoři Marzari a spol. (2021) řadí moduly, které mají být aktivovány, jeden po druhém a někteří Dalal a spol. (2021) zahrnují parametry pro konkrétní dílčí cíle. Způsob orchestrace modulů Plappert a spol. (2018) je stále nevyřešen. Objevily se přístupy k návrhu obecného sekvenceru Clegg a spol. (2018), ale jeho univerzálnost je stále omezená.

Protože bychom rádi přispěli k řešení tohoto problému, navrhujeme nový algoritmus optimalizace několika policy (MultiPPO2) a testujeme jej na jedno- i vícekrokových úlohách s různými dílčími cíli. Jeho výsledky překonávají tradiční algoritmy s jednou policy. Hlavní přínosy našeho přístupu jsou následující:

- Implementovali jsme algoritmus, který je schopen trénovat několik policy v rámci jedné epizody a přepínat mezi nimi během testování.

- Trénovali jsme a testovali náš algoritmus MultiPPO2 ve virtuálním robotickém simulátoru, abychom prokázali jeho lepší výkonnost ve srovnání s metodami s jednou policy.

2 Metody

Náš přístup se zaměřuje na robotické úlohy, které lze rozdělit na menší podúlohy. Trénování urychlíme tím, že trénujeme všechny dílčí úlohy za sebou. Pro podrobné srovnání tréninkových algoritmů jsme vyvinuli nové benchmarky, které jsou založeny na pokročilé robotické manipulaci, konkrétně na rotaci objektu v 6DOF (Pick and Rotate) a následování trajektorie při manipulaci s objektem (Swipe).

2.1 Prostředí

Naše experimenty provádíme pomocí toolboxu pro trénování a testování myGym Vavrecka a spol. (2021). Je postaven na fyzikálním enginu Pybullet. Používáme dva modely skutečných průmyslových robotů - Kuka IIWA (Kuka) a Franka Emika Panda (Panda). Používáme kloubové úhlové řízení robotů. Každý robot má 7 pohyblivých kloubů. K manipulaci s předmětem používáme magnetické chapadlo.

2.1.1 MultiPPO2

Vyvinuli jsme algoritmus, který dokáže v rámci jedné epizody trénovat více policy a přepínat mezi nimi na základě průběhu řešení úlohy. Vycházíme z algoritmu PPO2 Schulman a spol. (2017) z knihovny Stable Baselines Hill a spol. (2018). V porovnání se standardním PPO2 se v MultiPPO2 během tréninku instancují samostatné moduly. Úlohy, které se snažíme trénovat, rozdělíme na několik jednodušších podúloh a pro každou z nich napíšeme odměnu. Model MultiPPO2 při inicializaci vytvoří dílčí model pro každou úlohu a spáruje je s odpovídající funkcí odměny. Trénování modelu MultiPPO2 je popsáno v části Alg. 1 a probíhá podobně jako u modelu PPO2.

Algorithm 1 MultiPPO2

```
function TRAIN_MODEL(M, reward)
    M = "number of desired models"
    reward = "reward functions array"
    Model.submodels = [ ]
    for m in M do
        models[m] = PPO2_model()
    end for
    for i in epochs do
        id = decide(observation)
        id = ïd of appropriate submodel"
        train_batch(Model.submodels[id], reward[id])
    end for
    return Model
end function
```

Druhým novým krokem je zavedení predikčního modulu. Systém na základě aktuálního pozorování rozhodne, jakou dílčí úlohu se síť právě snaží splnit. Podle rozhodnutého úkolu se na této části dat cvičí odpovídající model s odměnou náležející danému úkolu. Při použití natrénovaného modelu MultiPPO2 pro testování se postupuje obdobně jako při trénování, jak je uvedeno v Alg. 2.

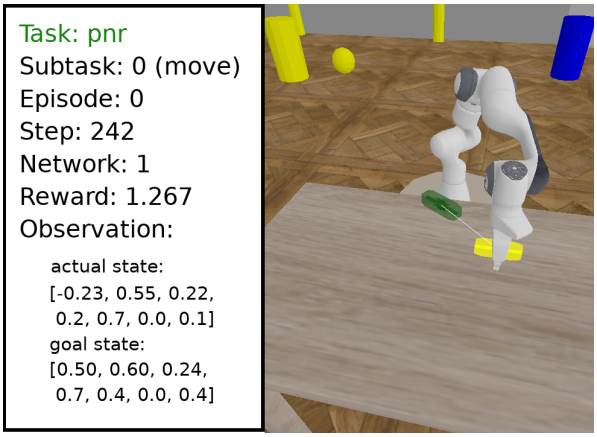
2.2 Úlohy

Vyvinuli jsme novou sadu úloh v rámci myGym toolboxu, abychom porovnali náš algoritmus MultiPPO2 s

Algorithm 2 MultiPPO2 predictions

```
function PREDICT(Model, observation)
    id = decide(observation)
    id = ïd of appropriate submodel"
    return Model.submodels[id].predict(observation)
end function
```

PPO, PPO2 Schulman a spol. (2017) a ACKTR Wu a spol. (2017). Tyto single-policy algoritmy již byly vyhodnoceny v rámci myGym toolboxu Vavrecka a spol. (2021) na úlohách bez dílčích cílů (reach, pick, push) s vysokou úspěšností.



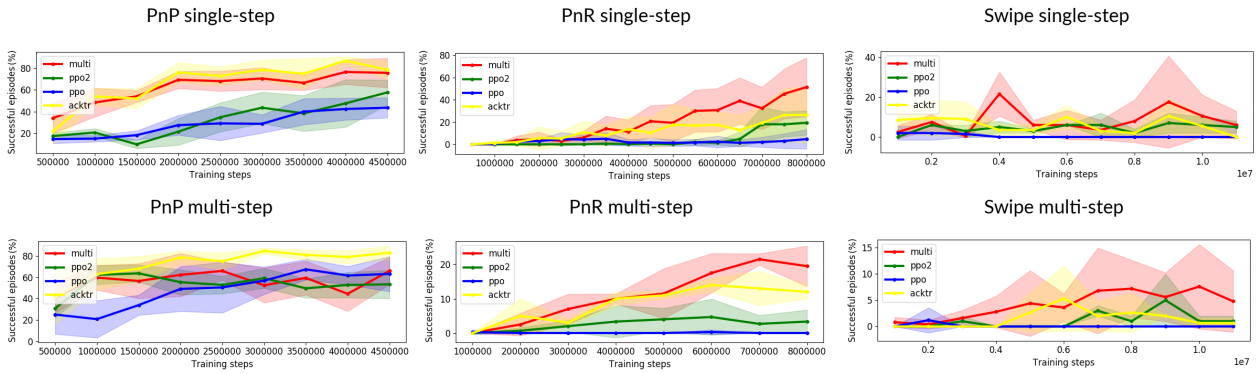
Obr. 1: Úloha Pick and Rotate je modifikací úlohy Pick and Place. Robot musí vybrat objekt a poté jej přesně umístit v prostoru podle požadavků cílového stavu. Na pravé straně jsou podrobné informace o průběhu tréninku.

První novou úlohou je Pick and Place s podúlohami (PnP), kde každá část procesu má samostatnou odměnu. Úloha Pick and Rotate (PnR) představuje rozšířenou verzi PnP (viz například obr. 1) . Tato úloha se týká specifického typu robotické manipulace, kdy je robot trénován k tomu, aby zvedl předmět a otočil jej do určité orientace. Úloha obecně zahrnuje následující kroky: První dva dílčí cíle jsou podobné jako u PnP, ale třetí dílčí cíl (otočit) vyžaduje, aby robot přesně otočil uchopený objekt do polohy 6DOF.

Třetí a nejobtížnější úlohou je Swipe. Oproti úlohám Pick a Rotate je zde prezentována nová dílčí úloha. Robot si musí osvojit nové dovednosti, aby v této úloze uspěl. Swipe vyžaduje, aby robot našel a přesunul houbu do požadované polohy a sledoval specifickou trajektorii po desce stolu.

2.3 Trénink a evaluace

Úlohy popsané v předchozí části bylo trénováno ve virtuálním robotickém simulátoru s robotickými ra-



Obr. 2: Srovnání jednokrokových a vícekrokových úloh s rostoucí složitostí zleva doprava v jednokrokovém (nahoře) a vícekrokovém (dole) scénáři

Tab. 1: Srovnání algoritmů pro úlohy s rostoucí složitostí

ALGORITHM	PNP SINGLE	PNP MULTI	PNR SINGLE	PNR MULTI	SWIPE SINGLE	SWIPE MULTI
PPO	43.6 (9.2)	63.1 (16.4)	4.6 (8.6)	0.8 (0.4)	0.0. (0.0)	0.0 (0.0)
PPO2	57.6 (11.3)	53.4 (13.2)	19.3 (10.8)	4.2 (2.9)	5.9 (3.0)	1.0 (1.0)
ACKTR	**78.4 (8.2)**	**83.0 (6.4)**	26.4 (25.0)	12.4 (3.1)	0.0 (0.0)	0.6 (0.9)
MULTIPPO2 (OURS)	75.6 (13.5)	66.0 (12.7)	**51.3 (26.4)**	**21.6 (5.2)**	**6.0 (6.9)**	**4.8 (5.8)**

meny Panda i Kuka (oba roboti dosáhli podobných výsledků). Poloha a orientace objektů při inicializaci byla náhodná. Jednokrokové úlohy jsme trénovali po 512 tréninkových krocích a vícekrokové úlohy po 1024 krocích.

Přesnost algoritmu opakovaně vyhodnocujeme po určitém počtu tréninkových kroků. Robotovi je předloženo 50 nových konfigurací a sečetli úspěšné epizody. Vyhodnocujeme také úspěšnost dílčích úkolů, protože nám může napovědět, který podúkol nebyl naučen správně. Počítá se také průměrný počet kroků na každý dílčí cíl a průměrná vzdálenost od cílového stavu.

3 Výsledky

Výsledek pro úlohu PnP prokázaly (**??**, že jak single-policy, tak multi-policy sítě jsou schopny se úlohu naučit. Algoritmy MultiPPO2 a ACKTR mírně překonávají algoritmy PPO a PPO2 a dosahují 80 % přesnosti. Vzhledem k tomu, že tato úloha nevyžaduje různorodé dovednosti, algoritmy se výrazně neliší. Podobné výsledky jsou i u vícekrokového scénáře (graf vlevo dole). Mezi single-policy PPO2 a multi-policy MultiPPO2 není žádný rozdíl. Celkový výkon pro vícekrokový scénář je překvapivě lepší než pro jednokrokový, přestože je úloha obtížnější kvůli náhodné poloze paže po dokončení první dílčí úlohy. Můžeme konstatovat, že všechny algoritmy vykazují v úlohách PnP téměř podobný výkon.

Ve složitějším scénáři, kde jsou prezentovány různé dovednosti (Pick and Rotate), můžeme vidět větší rozdíl mezi sítěmi s jednou a více policy. Jelikož je jako jeden z dílčích cílů prezentováno otáčení 6DOF, jednopolitické sítě si vedou mnohem hůře ve srovnání s MultiPPO2. Naše metoda dosahuje 51 % přesnosti ve srovnání s 26 % u PPO2 a 19 % u ACKTR. PPO v této úloze selhává, protože dosahuje přibližně 4 % přesnosti. Ve vícekrokovém scénáři metoda MultiPPO2 opět překonala algoritmy s jednou sítí. Jak PPO, tak PPO2 nebyly schopny se tuto úlohu naučit. Výkonnost MultiPPO2 klesá mezi jedno- a vícekrokovými úlohami z 51 na 21 %.

Analýza nejobtížnější úlohy ukázala, že algoritmus s více policy si vede lépe, ale výsledky nejsou uspokojivé. MultiPPO2 dosahuje 20 % přesnosti v polovině tréninku, ale poté výkonnost klesá. Podobný pokles můžeme pozorovat i u vícekrokového scénáře. To lze přičíst složité funkci odměny, kdy funkce odměny pravděpodobně narušuje proces trénování a je třeba jej dále zlepšovat. Jak je vidět na grafech vpravo, algoritmy s jednou policy nebyly schopny tuto úlohu vyřešit.

Výsledky pro všechny úlohy jsme shrnuli v tab. **??**. Můžeme konstatovat, že MultiPPO2 překonává všechny single-policy algoritmy v jedno- i vícekrokových úlohách, kde jsou vyžadovány různorodé dovednosti.

4 Závěr

Potvrdili jsme hypotézu, že různorodé dovednosti (různé dílčí cíle s různými funkcemi odměny)

způsobují problémy single-policy algoritmů. Ty nejsou schopny zachytit variabilitu stavového prostoru úlohy. Představili jsme metodu pro tréninkový proces RL, která je schopna se s tímto problémem vypořádat. Náš algoritmus MultiPPO2 překonal single-policy algoritmy reprezentované PPO, PPO2 a ACKTR ve složitých úlohách vyžadujících různorodé soubory dovedností. Pomocí robotické simulace jsme ukázali, že algoritmus MultiPPO2 je efektivní při řešení problémů, které lze rozdělit na menší části. Domníváme se, že strategii více policy lze aplikovat i na další algoritmy RL. Tuto možnost budeme zkoumat v budocnu. Další možné pokračování zahrnuje automatické rozdělení úlohy na dílčí úlohy. Rádi bychom také prozkoumali kompozici dílčích cílů v komplexních úlohách.

Poděkování

Tento příspěvěk vznikl za podpory projektu GAČR č. 23-04080L a č. 21-31000S).

Literatúra

Alet, F., Lozano-Pérez, T. a Kaelbling, L. P. (2018). Modular meta-learning. V *Conference on Robot Learning*, str. 856–868. PMLR.

Andreas, J., Klein, D. a Levine, S. (2017). Modular multitask reinforcement learning with policy sketches. V *International Conference on Machine Learning*, str. 166–175. PMLR.

Clegg, A., Yu, W., Tan, J., Liu, C. K. a Turk, G. (2018). Learning to dress: Synthesizing human dressing motion via deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 37(6):1–10.

Dalal, M., Pathak, D. a Salakhutdinov, R. R. (2021). Accelerating robotic reinforcement learning via parameterized action primitives. *Advances in Neural Information Processing Systems*, 34:21847–21859.

Devin, C., Gupta, A., Darrell, T., Abbeel, P. a Levine, S. (2017). Learning modular neural network policies for multi-task and multi-robot transfer. V *2017 IEEE international conference on robotics and automation (ICRA)*, str. 2169–2176. IEEE.

Hill, A., Raffin, A., Ernestus, M., Gleave, A., Kanervisto, A., Traore, R., Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S. a Wu, Y. (2018). Stable Baselines. `https://github.com/hill-a/stable-baselines`.

Marzari, L., Pore, A., Dall'Alba, D., Aragon-Camarasa, G., Farinelli, A. a Fiorini, P. (2021). Towards hierarchical task decomposition using deep reinforcement learning for pick and place subtasks. V *2021 20th International Conference on Advanced Robotics (ICAR)*, str. 640–645. IEEE.

Plappert, M., Andrychowicz, M., Ray, A., McGrew, B., Baker, B., Powell, G., Schneider, J., Tobin, J., Chociej, M., Welinder, P. a spol. (2018). Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*.

Pore, A. a Aragon-Camarasa, G. (2020). On simple reactive neural networks for behaviour-based reinforcement learning. V *2020 IEEE International Conference on Robotics and Automation (ICRA)*, str. 7477–7483. IEEE.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A. a Klimov, O. (2017). Proximal policy optimization algorithms.

Vavrecka, M., Sokovnin, N., Mejdrechova, M. a Sejnova, G. (2021). Mygym: Modular toolkit for visuomotor robotic tasks. V *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*, str. 279–283, Los Alamitos, CA, USA. IEEE Computer Society.

Wu, Y., Mansimov, E., Liao, S., Grosse, R. a Ba, J. (2017). Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation.