Erőforrások skálázása megerősítéses tanulással

Ph.D. tézisfüzet

Nguyen Tuan Hai

Hálózati Rendszerek és Szolgáltatások Tanszék
Villamosmérnöki és Informatikai Kar
Budapesti Műszaki és Gazdaságtudományi Egyetem

Témavezető:
Dr. Do Van Tien
Budapesti Műszaki és Gazdaságtudományi Egyetem

Budapest, Magyarország
2022

1. Bevezető

Lehet, sokaknak ismerős az a szituáció, amikor a reggeli előadás előtt még gyorsan beugrik a boltba egy jegeskávéért, de azt veszi észre, hogy a három kasszából csak az egyik működik és az oda vezető sor végeláthatatlan. A hosszú sor láttán az embernek ilyenkor elmegy a kedve a vásárlástól és inkább kihagyja a reggeli kávéját. Vagy képzeljük el azt, amikor egy egyetem meghirdeti a tárgyfelvételt a következő félévre az online oktatási rendszerén keresztül, hétfő este 6-ra. Egész délután a tökéletes órarend összerakásával fáradozik többezer diák, és amikor az óra 6 órát üt, mindenki egyszerre próbál belépni a rendszerbe, ezzel túlterhelve az egyetemi szervereket.

A két fent vázolt eset problémái hasonló okokból gyökereznek: nincs elég erőforrás az adott méretű igény kiszolgálására. Ez a probléma számos más helyen is előfordulhat napi szinten. Hány repülőjáratra van szükségünk egy vonal üzemeltetésére? Hány mobilhálózati cellára van szükségünk egy sűrűn lakott területen? Hány szervergépre van szükségünk egy új online játék elindulásakor? Ezek a kérdések messziről ugyanannak tűnhetnek, ugyanakkor minden eset más és egyedi – az ördög a részletekben rejlik. Ezért a gyakorlatban nehéz olyan módszert találni, ami általánosan képes megoldani minden ilyen típusú problémat. Tanulmányim során erőforráselosztással foglalkoztam információs és kommunikációs technológiai (information and communication technology – ICT) rendszerekben. Konkrétabban, felhő alapú 5G hálózatokban és aktor alapú elosztott rendszerekben.

Az erőforrások pontos meghatározása nem triviális feladat. Túl sok erőforrás kiosztása túlellátottsághoz vezethet. Ez azt jelenti, hogy túl sok felesleges erőforrás lesz kiosztva, amely növelni fogja az üzemeltetési költségeket. Például, ha túl sok pénztárost alkalmazunk vagy ha túl sok szervert bérlünk. Ugyanakkor túl kevés erőforrás alulellátottságot hozhat létre. Ilyenkor a szolgáltatás minősége (quality of service – QoS) romolhat, ami elégedetlen ügyfeleket és profitkiesést okozhat. Továbbá figyelembe kell vennünk azt, hogy az igény nem biztos, hogy időben állandó. Csúcsidőben valószínűleg több kasszásra van szükségünk a boltban vagy több szerver kell a felhőben. Ezért olyan megoldásokra van szükségünk, amelyek dinamikusan képesek állítani az erőforrásokat az igény függvényében. Ehhez gépi tanulásos módszereket, konkrétabban megerősítéses tanulást (reinforcement learning – RL) és mely megerősítéses tanulást (deep reinforcement learning – DRL) alkalmaztam.

2. Célok és motivációk

A hálózati funkciók virtualizációja (network function virtualization – NFV) során az egyes hálózati szolgáltatások (network service – NS) fizikai (physical network function – PNF) és virtuális funkciókból (virtual network function – VNF) tevődnek össze. A koncepciót az *European Telecommunications Standard Institute* (ETSI) vezette be [4, 5]. Az NFV az NS-ek gyors és rugalmas telepítését teszik lehetővé, mindezt úgy, hogy közben képes megbízhatóságot és

skálázhatóságot is nyújtani. A gyakorlatban egy hálózati funkciót akár több VNF komponens (VNF component – VNFC) párhuzamos futtatása is megvalósíthatja melyek számát a VNF üzemeltetője a forgalom függvényében állíthatja. Például képzeljünk el egy olyan hálózati funkciót, mely egy videó folyam filmkockáit dolgozza fel. A felhasználók számától és a filmkocka számtól függően az üzemeltetőnek több VNFC-t is el kell indítania, hogy az igényeket ki tudja szolgálni.

Az ún. aktor modellben az aktorok jelentik a számítás egységét. Az aktorok üzeneteken keresztül kommunikálnak. Egy aktor egy beérkezett üzenet hatására újabb üzenetet küldhet egy vagy több másik aktornak; létrehozhat további aktorokat; vagy végrehajthat bizonyos műveleteket. Az aktorok aszinkron működnek, ezzel lehetővé téve az egyidejű működést, mely kifejezetten alkalmas olyan elosztott rendszerek, mint például az IoT rendszerek implementálására [3, 6]. Az architektúra elrejti a fejlesztők elől az olyan alacsony szintű elemeket, mint például a zárakat vagy a szálakat, ezáltal kényelmesebbé válik az alkalmazásfejlesztés. Azonban a használatban levő szálkészlet mérete nagyban befolyásolhatja a rendszer működését, ugyanis a kihasználatlan szálok memóriát foglalhatnak fölöslegesen, valamint a szálak közötti kontextusváltás lassíthatja a rendszert.

Az 5G maghálózat (5G Core – 5GC) egyik eleme a felhasználói *User Plane Function* (UPF). A felhasználói készülékek (user equipment – UE) csomagjainak az adathálózat (data network – DN) felé való továbbításáért felelős. Azaz az UE-k kiszolgálása nagyban függ a futó UPF példányok számától [11]. Egy felhő alapú rendszerben az UPF példányok konténerekben vagy Podokban futnának. A Podok számának szabályzására és magának a felhőnek a vezénylésére az egyik legelterjedtebb eszköz pedig a Kubernetes [2]. A Kubernetes-ben az alapértelmezett skálázó algoritmus az ún. *Horizontal Pod Autoscaler* (HPA), melyre a DRL egy jó alternatíva lehetne. Azonban a DRL a döntéshozatalának sztochasztikus jellege miatt időnként rossz döntéseket hozhat, és ez teljesítményromláshoz vezethet. Ezt a problémát klasszifikációs módszerekkel lehetne orvosolni, amely a skálázási akciókat determinisztikusan rendelné a rendszer egyes állapotaihoz.

Tanulmányaim során a következő célokat tűztem ki:

- Egy DRL alapú módszer kifejlesztése VNF példányok skálázására konstans és változó forgalom alatt. A módszer legyen rugalmas abban az értelemben, hogy a QoS szintet az üzemeltető igényeinek megfelelően tudja állítani.

- Egy DRL alapú módszer kifejlesztése egy aktor alapú rendszer szálkészletének méretezésére. A módszer vegye figyelembe, hogy a forgalom időben változhat, valamint a szálak elindítása sem elhanyagolható. További cél a módszer más meglévő algoritmusokkal való összehasonlítása, mint pl. a Java `ThreadPoolExecutor` által alkalmazott *"time out"* szabály.

- Megmutatni, hogy a DRL jobban képes skálázni az UPF Podokat a HPA-nál, valamint a DRL ötvözése olyan klasszifikációs módszerekkel, mint a szupport vektor gépek (support vector machines – SVM), hogy kiküszöböl-

jük a DRL sztochasztikus döntéshozatalából származó teljesítményromlásokat.

3. Módszertan

Az autoskálázási módszerek általában az alábbi öt kategória valamelyikébe sorolhatóak: sorbanállási elmélet, küszöbérték alapú szabályok, szabályzástechnika, idősoros elemzések [1], és megerősítéses tanulás [7]. Az RL egyik előnye, hogy alkalmazása nem igényel az adott szakterülethez köthető tudást. Egy RL ágens a környezetével interakcióba lépve képes tapasztalatokat gyűjteni és a saját döntéseit értékelni. Az RL e tulajdonságát szokás modellfüggetlenségnek is nevezni, mivel az RL-nek nincs szüksége egy a környezetéről alkotott előzetes modellre.

Ahhoz, hogy alkalmazhassuk az RL-t egy problémára, előbb szükségünk van azt egy megfelelő Markov döntési problémaként (MDP) definiálni. Egy MDP az $\langle \mathcal{S}, \mathcal{A}, p, r, \gamma \rangle$ ötössel írható le, ahol $\mathcal{S}$ a rendszer állapotainak halmaza; $\mathcal{A}$ a végrehajtható akciók halmaza; $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ az állapotátmeneti valószínűségek függvénye; $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ a közvetlen jutalom függvénye egy adott állapotátmenetre; és $\gamma \in [0, 1]$ az ún. diszkontáló faktor [13].

Egy tanulási folyamat i-edik ciklusa során egy RL ágens a következő tevékenységeket hajtja végre: az ágens megfigyeli az $s_i \in \mathcal{S}$ állapotot; az ágens a döntéshozatali függvénye, $\pi : \mathcal{S} \to \mathcal{P}(\mathcal{A})$, alapján meghatározza az a_i akciót, (itt $\mathcal{P}(\cdot)$ egy adott halmaz feletti valószínűség-eloszlást jelöli); végrehajtja az a_i akciót a környezetében; az ágens egy r_i jutalmat kap a környezetétől, majd megfigyeli a következő, s_{i+1} állapotot; a megfigyelt (s_i, a_i, r_i, s_{i+1}) értékek alapján az ágens javít a döntéshozatali függvényén, π-n.

Az $\mathcal{S}$ és $\mathcal{A}$ halmazok jó definiálása egy nagyon fontos lépés. Túl nagy dimenziójú állapot- vagy akciótér ún. állapottér-robbanáshoz vezethet. Ez azt jelenti, hogy az $\mathcal{S}$ és $\mathcal{A}$ által definiált terek olyan nagyra nőhetnek, hogy egy RL ágens számára lehetetlenné válna minden állapot meglátogatása és a teljes tér feltérképezése. Azonban ha az állapottér túl kicsi, előfordulhat, hogy a rendszer két különböző állapota között nem fogunk tudni különbséget tenni, vagy az is megtörténhet, hogy jelentősen lekorlátozódik a lehetséges akciók száma egy adott állapotban. Az r jutalomfüggvény definiálása is nagyon fontos, hiszen ez határozza meg a célt, ami alapján az RL ágens optimalizálni fog. Rosszul definiált r esetén előfordulhat, hogy az RL ágens lokális minimumba reked. A p állapotátmeneti függvény írja le a modellt, azonban ennek ismeretére nincs szüksége az RL ágensnek, mert az közvetlenül lép interakcióba a környezetével. Az RL ágens az állapotátmenetekből, a döntésekből és a jutalmakból tanul csupán, modelfüggetlenül.

A szakirodalomban több különféle RL módszer is megtalálható. Az egyik legrégebbi és legismertebb ilyen módszer a Q-tanulás [14], mely minden állapot-akció párosra megpróbálja maximalizálni a $Q(s, a)$ értékfüggvényt (q-faktorokat) minden $s \in \mathcal{S}$ és $a \in \mathcal{A}$-ra. Általában $\mathcal{S}$ olyan nagy, hogy az egész állapottér nem fér bele egy számítógép memóriájába. Ennek érdekében szokás neurális hálókkal közelíteni a $Q(s, a)$ értékeket, ezt a módszert mély Q-tanulásnak (deep

Q learning) vagy mély Q hálózatoknak (deep Q network – DQN) is szokás nevezni [9]. Egy másik népszerű módszercsalád az akcióválasztási függvények gradiensét használja optimalizálásra. Ilyen például az aktor-kritikus (actor-critic – AC) [8] módszer vagy az ún. *proximal policy optimization* (PPO) [12, 10].

4. Tézisek

4.1. VNFC skálázás DRL-lel

Képzeljünk el egy olyan VNFC-kből álló NS-t mely mobil felhasználók videóinak adatfolyamait dolgozza fel filmkockánként. Tegyük fel, hogy ezek a filmkockák érkezése leírható egy konstans λ rátájú Poisson-folyamattal, valamint hogy a filmkockák feldolgozásának az ideje exponenciális eloszlást követ. Az üzemeltető ΔT időközönként dönthet arról, hogy kifele (*scale out*), befele (*scale in*) skálázik, vagy nem csinál semmit. Az igények érkezésekor előfordulhat, hogy nincs szabad VNFC a rendszerben, amely feldolgozná. Ezek alapján két esetet különböztettem meg: az első esetben a kérések egy végtelen bufferbe kerülnek és *first-in first-out* (FIFO) jelleggel kerülnek sorra; a második esetben nincs buffer, és ilyenkor a kéréseket elutasítja, azaz blokkolja a rendszer. Az operátor célja a VNFC minimalizálása úgy, hogy az átlagos várakozási időt vagy az átlagos blokkolási rátát egy bizonyos küszöbérték alatt tartsa (T_{th} valamint $p_{b,th}$).

Legyen $\{I(t), J'(t), K'(t)\}$ az állapotokat leíró hármas, ahol $I(t)$ a VNFC-k száma, $J'(t)$ jelzi, ha van szabad VNFC példány, valamint $K'(t)$ jelzi, ha van igény a sorban a t időpillanatban. Legyen az akciótér $\mathcal{A} = \{\texttt{ScaleIn}, \texttt{ScaleOut}, \texttt{None}\}$, továbbá legyenek a jutalomfüggvények:

$$r_{m,b} = \begin{cases} \frac{I(t_m^+)-n}{n} & \hat{p}_{b,m} > p_{b,th} \\ \frac{-I(t_m^+)}{n} & \hat{p}_{b,m} \leq p_{b,th} \end{cases}$$

és

$$r_{m,T} = \begin{cases} \frac{I(t_m^+)-n}{n} & \hat{T}_m > T_{th} \\ \frac{-I(t_m^+)}{n} & \hat{T}_m \leq T_{th} \end{cases}$$

blokkolásra és várakozási időre, ahol a $p_{b,th}$ és T_{th} értékeket a QoS határozza meg.

Javaslatot teszek a VNFC-k DRL-lel való skálázására. Tanulmányaim során azt találtam, hogy az aktor-kritikus módszer gyorsan és megbízhatóan képes megtalálni az optimális VFNC számot.

1. Téziscsoport. *[P1] Egy aktor-kritikus módszerrel tanított DRL ágens képes minimalizálni a VNFC-k számát egy NS-ben.*

Formálisan definiáltam egy MDP-t erre a VNF skálázási problémára és szimulációkkal megmutattam, hogy a DRL egy megfelelő eszköz a szabályzásra.

1.1. Tézis. *A fent leírt MDP esetén az aktor-kritikus algoritmus képes megtalálni az optimális számú VNFC-t a VNF skálázási problémában szakértői tudás*

nélkül, adott, Poisson-érkezésekkel és fix érkezési rátával jellemzett forgalom mellett.

Az 1.1 Tézis során kísérleteket végeztem el konstans λ érkezési ráta mellett. Időben változó érkezési ráták esetén több DRL ágenst tanítottam be különböző λ értékekre. Javaslatot teszek egy többágenses módszerre, ahol a mért érkezési ráta függvényében kiválasztjuk a megfelelő ágenst és azzal szabályozzuk a rendszert.

1.2. Tézis. *Változó forgalom esetén a többágenses módszer képes minimalizálni a VNFC-k számát, miközben a QoS-t megfelelő szinten tartja.*

4.2. Szálkészletek skálázása aktor alapú rendszerekben

Vegyük egy aktor alapú IoT rendszer MDP formalizációját, ahol adott t időben *session*-ök érkeznek $\lambda(t)$ intenzitással és a session-ök hossza exponenciális eloszlású μ rátával. Minden egyes session közben fix, azonos méretű csomagok érkeznek a rendszerbe meghatározott időközönként. Ezeknek a csomagoknak a feldolgozását szálak végzik el. Ha a szálkészlet minden szála foglalt, a csomagok egy véges várakozási sorba kerülnek. Ha a várakozási sor megtelt, az újonnan érkező csomagokat a rendszer elutasítja, blokkolja.

Legyen $\{X(t), Y(t), B(t), \hat{\lambda}(t)\}$ a rendszer állapotterét leíró négyes, ahol $X(t)$ az aktív session-ök száma, $Y(t)$ a szálkészlet mérete, $B(t)$ a foglalt szálak száma, valamint $\hat{\lambda}(t)$ a mért érkezési intenzitás. Két fajta akcióteret vizsgáltam. Az első esetben három fajta akció mellett dönthet az ágens: elindít egy új szálat, leállít egy szabad szálat, vagy nem csinál semmit, azaz $\mathcal{A}_1 = \{\texttt{START\_THREAD}, \texttt{STOP\_THREAD}, \texttt{NoOp}\}$. A második esetben az ágens egyszerre több szálat is elindíthat vagy leállíthat, azaz $\mathcal{A}_2(t) = \{B(t), B(t)+1, \ldots, N\}$, ahol N a szálkészlet maximális mérete. Legyen tovább a a jutalomfüggvény

$$r_i = \begin{cases} -\kappa\hat{\delta}_i & \text{if } \delta_{th} < \hat{\delta}_i \text{ or } p_{b,th} < \hat{p}_{b,i} \\ -Y(T_i) & \text{if } \delta_{th} \geq \hat{\delta}_i \text{ and } p_{b,th} \geq \hat{p}_{b,i} \end{cases},$$

ahol δ_{th} és $p_{b,th}$ a QoS által meghatározott várakozási idő és a blokkolási ráta küszöbértékei, $\hat{\delta}_i$ és $p_{b,i}$ a mért várakozási idők és blokkolási ráták az i-edik döntési pillanatban, valamint κ egy állítható hiperparaméter.

Javaslatot teszek az aktor alapú rendszerekben található szálkészletek skálázására DRL-lel.

2. Téziscsoport. *[P2] A DRL képes optimalizálni a szálkészlet méretét egy aktor alapú rendszerben.*

Formalizáltam egy szálkészlet MDP-jét és PPO segítségével betanítottam egy DRL ágenst. Szimulációkon keresztül megmutattam, hogy a DRL jobban teljesít, mint a szálkészletek által gyakran használt időzítéses módszer.

2.1. Tézis. *A fent leírt MDP alapján egy DRL képes minimalizálni a szálak számát egy aktor alapú rendszer szálkészletében, miközben az átlagos blokkolási*

rátát egy adott küszöbérték alatt tartja. A DRL κ és ξ hiperparamétereit a populáció alapú tanítás (population-based training – PBT) és a grid keresés ötvözésével kaphatjuk meg.

Ha a szálak indítási ideje kicsi, akkor a PPO hasonlóan teljesít az időzítéses módszerhez. Ha az indítási idő nagy, a DRL ágens jobban teljesít, mint az időzítéses módszer.

2.2. Tézis. *Több szál együttes indítása és leállítása nem javítja a működést.*

4.3. UPF Podok skálázása 5G/6G maghálózatban

Vegyük az MDP formalizációját egy felhőn futó 5G maghálózatnak. A felhőben található szerverek Podokba csoportosulnak. Az üzemeltető a skálázás során Podokat indíthat el vagy állíthat le. Egy adott t időpillanatban a PDU session-ök érkezési intenzitása $\lambda(t)$ és a kiszolgálási idejük exponenciális eloszlású μ rátával.

Legyen $\{d_{\text{on}}(t), d_{\text{boot}}(t), l_{\text{sess}}(t), l_{\text{free}}(t), \hat{\lambda}(t)\}$ az állapotteret leíró ötös, ahol $d_{\text{on}}(t)$ a futó Podok száma; $d_{\text{boot}}(t)$ az éppen inicializációs fázisban levő Podok száma; $l_{\text{sess}}(t)$ a PDU session-ök száma; $l_{\text{free}}(t)$ a szabad session kapacitás mértéke; és $\hat{\lambda}(t)$ a mért érkezési intenzitás. Legyen az akciótér $\mathcal{A} = \{\texttt{start}, \texttt{stop}, \texttt{noaction}\}$, azaz egy Pod elindítása, egy Pod leállítása, vagy semmilyen akció. Legyen a jutalomfüggvény

$$r_i = \begin{cases} -\kappa \hat{p}_{b,i} & \text{if } \hat{p}_{b,i} > p_{b,th} \\ -\hat{d}_{\text{on}}(T_i) & \text{if } \hat{p}_{b,i} \leq p_{b,th} \end{cases}$$

, ahol $p_{b,th}$ a blokkolás küszöbértéke, $\hat{p}_{b,i}$ a mért blokkolási ráta az i-edik döntési pontban, és κ az algoritmus egy hiperparamétere.

A [P3]-ban formalizáltam az MDP-t a felhő alapú 5G maghálózatbeli UPF Podok szabályzására. Továbbá javaslatot tettem a DRL alkalmazására UPF Podok skálázására. Megmutattam, hogy a DRL jobban teljesít, mint a Kubernetes HPA.

3. Téziscsoport. *[P3] A DRL vagy SVM által tanított ágensek képesek optimalizálni az UPF Podokat egy 5G maghálózatban.*

3.1. Tézis. *A fent leírt MDP formalizációval (állapottér, akciótér, jutalomfüggvény) az RL ágens képes minimalizálni az UPF Podok számát miközben az átlagos blokkolási rátát egy adott $p_{b,th}$ küszöbérték alatt tartja.*

3.2. Tézis. *A PPO által az átlagos jutalom módszerével betanított RL ágens jobban teljesít, mint a HPA.*

A PPO sztochasztikus módon választ akciókat. Ez azt jelenti, hogy a kimenete nem egy konkrét akció, hanem egy valószínűségi eloszlás a lehetséges akciók felett, és akcióválsztás során e szerint az eloszlás szerint mintavételezünk. Ez tanítás során egy nagyon hasznos tulajdonság, ugyanis így minden akciónak

van egy minimális esélye a kiválasztódásra, és így könnyebben fel lehet deríteni az akcióteret. Azonban ez egyúttal azt is jelenti, hogy üzemeltetés közben előfordulhat, hogy az RL ágens egy hibás döntést hoz valamilyen nagyon kicsi valószínűséggel. Ez a jelenség gyakran előfordul, ha a forgalom hirtelen nagyot változik. A sztochasztikus akcióválasztás helyett egy lehetséges megoldás az, ha az RL segítségével állapot-akció párokat generálunk és a kapott adathalmazra egy klasszifikáló algoritmust alkalmazunk.

3.3. Tézis. *Ha egy betanított RL ágens által generált adathalmazon alkalmazzuk a lineáris SVM klasszifikációt, akkor az így keletkezett akcióválasztásos módszer jobban teljesít az RL ágensnél hirtelen nagy változások esetén. Az átlagos esetben nem teljesít jobban az RL ágensnél, de még ekkor is jobban működik a HPA-nál.*

Az SVM alapú ágens egy hátránya, hogy nem működik *online*. Ezért csak olyan esetekre javaslom, amikor a forgalomban gyakran történik hirtelen változás és nincs szükség online tanításra. Egyébként a DRL ágens használatát javaslom.

5. Eredmények alkalmazhatósága

Az 1 téziscsoport eredményei a VNF kontextusába helyezhetőek, és VNFC-k skálázására alkalmasak. A javasolt DRL ágens külön elemként implementálható, amely az NFV vezénylővel (NFV Management and Orchestration – NFV-MANO) kommunikál. Ebben a felállásban az NFV-MANO adott időközönként adatot küld a DRL ágens felé, mely cserébe javaslatot tesz skálázási akcióra a kapott információ függvényében. A skálázási akciót vagy a VNF menedzser (VNFM) vagy egy vezérlő (network function virtual orchestrator – NFVO) hajtja végre.

A 2 téziscsoport eredményei az Akka, és ezzel együtt a JVM szálkészlet szálaira alkalmazhatók. Ebben az esetben a DRL ágens a Java `ExecutorService`-be lenne integrálható. Az `ExecutorService` feladata lenne a szálkészlet monitorozása és a DRL ágens futtatása. Azaz, feladata lenne új szálak indítása, vagy meglévő szabad szálak leállítása a DRL ágens javaslatai alapján.

A 3 téziscsoport eredményei felhő alapú vezénylőrendszerekben, mint például a Kubernetes, alkalmazhatóak UPF Podok kezelésére. Ebben az esetben az UPF példányok Kubernetes Podokban lennének csomagolva, azaz egy Pod egy UPF példányt futtat. Egy testreszabott metrikaszerver monitorozná az 5G maghálózatban lévő UPF Podok és PDU session-ök számát. Ezt az információt pedig továbbküldené egy DRL alapú autoskálázó felé. Az autoskálázó pedig javaslatot tesz skálázási akcióra a Kubernetes Scale interfészen keresztül. Egy DRL ágens képes lenne online tanulni. Egy SVM alapú skálázóhoz előbb szükség lenne egy betanított DRL ágensre, amely először generálná a szükséges adathalmazt és tanítás után tudna csak integrálódni az autoskálázóba.

Publikációk

[P1] Hai T. Nguyen, Tien Van Do, Attila Hegyi, and Csaba Rotter. An approach to apply reinforcement learning for a vnf scaling problem. In *2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, pages 94–99, 2019.

[P2] Hai T. Nguyen, Tien V. Do, and Csaba Rotter. Optimizing the resource usage of actor-based systems. *Journal of Network and Computer Applications*, 190:103143, 2021.

[P3] Hai T. Nguyen, Tien Van Do, and Csaba Rotter. Scaling upf instances in 5g/6g core with deep reinforcement learning. *IEEE Access*, 9:165892–165906, 2021.

További munkák

[B1] Yang Yuan Li, Tien Van Do, and Hai T. Nguyen. A comparison of forecasting models for the resource usage of mapreduce applications. *Neurocomputing*, 418:36–55, 2020.

[B2] Tien Van Do, N.H. Do, H.T. Nguyen, Csaba Rotter, Attila Hegyi, and Peter Hegyi. Comparison of scheduling algorithms for multiple mobile computing edge clouds. *Simulation Modelling Practice and Theory*, 93:104–118, 2019. Modeling and Simulation of Cloud Computing and Big Data.

[B3] Hai T. Nguyen and T. V. Do. A model for a computing cluster with two asynchronous servers. In Nguyen-Thinh Le, Tien van Do, Ngoc Thanh Nguyen, and Hoai An Le Thi, editors, *Advanced Computational Methods for Knowledge Engineering*, pages 197–211, Cham, 2018. Springer International Publishing.

Hivatkozások

[1] M. Alazab, S. Khan, S. S. R. Krishnan, Q. Pham, M. P. K. Reddy, and T. R. Gadekallu. A multidirectional lstm model for predicting the stability of a smart grid. *IEEE Access*, 8:85454–85463, 2020.

[2] Cloud Native Computing Foundation. Kubernetes. `https://kubernetes.io/`, 2021. Accessed: 2021-06-10.

[3] D. Diaz Sánchez, R. Simon Sherratt, P. Arias, F. Almenarez, and A. Marín. Enabling actor model for crowd sensing and iot. In *2015 International Symposium on Consumer Electronics (ISCE)*, pages 1–2, June 2015.

[4] ETSI Group Specification. ETSI GS NFV 001 V1.1.1. Network Functions Virtualisation (NFV): Use Case, 2013, 2013.

[5] ETSI Group Specification. ETSI GS NFV 003 V1.2.1. Network Functions Virtualisation (NFV): Terminology for Main Concepts in NFV. 2014., 2014.

[6] Andreas Moregård Haubenwaller and Konstantinos Vandikas. Computations on the edge in the internet of things. *Procedia Computer Science*, 52:29 – 34, 2015. The 6th International Conference on Ambient Systems, Networks and Technologies (ANT-2015), the 5th International Conference on Sustainable Energy Information Technology (SEIT-2015).

[7] Tania Lorido-Botran, José Miguel-Alonso, and José Antonio Lozano. A review of auto-scaling techniques for elastic applications in cloud environments. *Journal of Grid Computing*, 12:559–592, 2014.

[8] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1928–1937, New York, New York, USA, 20–22 Jun 2016. PMLR.

[9] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529 EP –, Feb 2015.

[10] OpenAI, :, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique P. d. O. Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning, 2019.

[11] Csaba Rotter and Tien Van Do. A queueing model for threshold-based scaling of UPF instances in 5G core. *IEEE Access*, 9:81443–81453, 2021.

[12] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[13] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[14] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.