AGS - prohledávání prostoru TileWorld pomocí agentů

Michal Glos (xglosm01)
Tomáš Plachý (xplach08)

19. prosince 2021

1 Úvod

V tomto projektu jsme řešili problém prohledávání mapy **TileWord** pomocí autonomních agentů. Celkem jsme naimplementovali 4 agenty blíže popsané v sekci 2. S těmito agenty jsme pak provedli sérii experimentů na různých mapách v různých počtech, viz sekkce 5. Součástí tohoto projektu je také grafické zobrazení mapy a agentů ve dvou obrázcích, kdy nalevo je celá mapa s překážkami a agenty a napravo je část mapy, která již byla agenty prohledána. Součástí je také prostředí pro vytváření map, blíže popsáno v sekci 4. Tento program také nabízí možnost proces prohledávání uložit jako video ve formátu mp4.

Funknce implementující chování agentů byly optimalizovány pro rychlost, avšak některé kombinace typu agenta, velikosti prohledávané mapy a počtu agentů jsou vypočítávány velice dlouho a pro prohledání mapy je potřeba velikých výpočetních zdrojů. Pro spuštění programu je potřebný `Python` v minimální verzi 3.9. Pro ukládání videí je navíc potřeba *ffmpeg*.

2 Typy agentů

Celkem jsme naimplementovali 4 různé druhy agentů. Tito agenti jsou popsání v této sekci, seřazeni podle svých schopností vzestupně.

2.1 Random

Tento agent se pohybuje na poli pomocí náhodných kroků. Pomocí argumentů předaných na příkazovém řádku u něj lze upravit pouze možnost pohybovat se po poli i diagonálně, stejně jako u všech ostatních agentů. Protože tento agent nevykazuje jakoukoli známku inteligence a nemůže poznat, jestli již byla prohledána veškerá dostupná políčka, jeho kroky také zahrnují kontrolu dokončení prohledávání.

2.2 Naive

Tento agent již vykazuje určité známky inteligence, jeho přístup k prohledávání je však velice naivní. Agent si zvolí nejbližší neobjevené políčko (do vzdálenosti nebere v potaz obcházení překážek) jako cíl, kterého chce dosáhnout. Poté na svou aktuální pozici zkouší aplikovat všechny možné kroky, které nevedou do překážky nebo jiného agenta a vybere ten, který jej nejvíce přiblíží k cíli. Tento agent má slabinu v tom, že nedokáže naplánovat cestu tak, aby obešel případné překážky, proto je jeho součástí detekce zacyklení pohybu. Pokud je detekován cyklus, agent udělá náhodný krok a se 75% pravděpodobností v každém kroku bude pokračovat dalším náhodným krokem. Tento agent již dokáže náhodnou mapu prohledat v poměrně krátké době, avšak narazí-li agent na pozici, která vyžaduje pro dosažení cíle několik kroků vedoucích do pozic, které se od cíle vzdalují, pravděpodobnost obejití překážky u každého pokusu klesá nepřímo uměrně k délce cesty potřebné pro její obejití. Tento agent také nedokáže určit, jestli již prohledal veškerá dostupná políčka, proto je opatřen funkcí kontrolující dokončení prohledávání.

2.3 Smart

Tento agent funguje na principu BFS prohledávání již objevené oblasti. Funkce implementující BFS vrátí agentovi seznam nejbližších neobjevených cílů (nejbližší políčka, na které svým vstupem objeví jiná políčká). Tenta agent již vzdálenost počítá včetně obcházení překážek (při prohledávání algoritmus BFS nebere v potaz pozice ostatních agentů). Z tohoto seznamu pak agent vybere jako cíl to políčko, na které svým vstupem objeví nejvíce políček neobjevených. Dále agent pomocí algoritmu A* najde nejkratší cestu k nalezenému cíli, včetně obcházení agentů. Pokud jsou všechna políčka, která by agent našel stoupnutím si na cíl objevena, agent si najde cíl jiný. Tento agent již dokáže sám zjistit, že již nemůže objevit další políčka, která jsou uzavřena překážkami, a sám skončit.

2.4 Smart_coop

Tento agent funguje na velice podobném principu jako smart, avšak algoritmus BFS nenalezne pouze nejbližší políčka, na které agent svým stoupnutím objeví políčko nové, ale vrátí políčka, které mají v dohledu agenta nějaké políčka neobjevené. Tím vytvoří de facto obrys objevené oblasti, ve které se agent nachází. Cíl si agent potom vybírá tak, že se všech ostatních agentů zeptá na jejich cíl. Agent zvolí svůj cíl tak, aby se případné zorné pole agenta v cíli neprotínalo se zorným polem jiného agenta v jeho cíli. Pokud žádné takové políčko neexistuje, agent uvažuje všechna možná políčka získaná algoritmem BFS. Z této množiny možných cílů si pak vybere ten nejbližší s nejvíce možnými objevenými políčky v cíli.

3 Spuštění

Skript lze spustit pomocí příkazu `python3 agentsearch.py` z kořenového adresáře a lze ovládat pomocí následujících argumentů předávaných skriptu při spuštění:

- `-c`: Otevře před samotným spuštěním prohledávání editor mapy, pomocí kterého lze na pole umístit překážky a agenty ve vlastním uspořádání. Při spuštění bez tohoto argumentu se vygeneruje mapa náhodně (což je ale ovlivnitelné dalšími argumenty).
  - `letft-click` Kliknutím levým tlačítkem myši lze na mapě menit obsazení políčka - z prázdného na překážku, z překážky na agenta a z agenta opět na prázdné políčko.
  - `"j"`: Stisknutím klávesy "j"uložíte vytvořenou mapu do adresáře `./saved_maps` a spustíte prohledávání.
  - `"Enter"`: Stisknutím klávesy Enter spustíte prohledávání na d vytvořenou mapou, ale mapa se neuloží.
- `-l [path/to/map]`: Načte mapu z JSON souboru na specifikované adrese.
- `---diagonally`: Umožní agentům pohyb po diagonálách.
- `--debug`: Spustí slriút v debug módu.
- `-m`: Spustí skript v módu pro měření výkonnosti agentů.
- `-d`: Číslo až čísla následující za tímto argumentem nastavují jeden (pro čtvercovou mapu) až dva rozměry mapy.
- `-a`: Číslo následující za tímto argumentem nastaví počet agentů (má vliv pouze při náhodném generování mapy).
- `-o`: Číslo následující za tímto argumentem nastaví počet agentů (má vliv pouze při náhodném generování mapy).
- `-v`: Číslo následující za tímto argumentem nastaví dosah zorného pole agentů.
- `-t`: Řetězec za tímto argumentem nastaví typ agentů (je na výběr z naive, random, smart a smart_coop)
- `-s`: Řetězec za tímto argumentem nastaví cestu pro uložení mapy.
- `-r`: Řetězec za tímto argumentem nastaví cestu pro uložení záznamu prohledávání. Pokud použijeme tento argument, dojde k uložení pvidea prohledávání na specifikovanou adresu.
- `--dpi`: Číslo za tímto argumentem nastaví DPI nahrávaného videa prohledávání.
- `--animation-speed`: Číslo za tímto argumentem nastaví rychlost animace prohledávání.
- `--frames`: Číslo nastaví počet snímků ukládaného videa. tímto argumentem nastaví počet snímků ukládaného videa.
- `-i`: Řetězec za tímto argumentem (z množiny small, medium, large) vybere odpovídající neprohledatelnou mapu a spustí nad ní prohledávání.

3.1 Příklady spuštění

- `python3 agentsearch.py -c` Pro spuštění editoru mapy (Po stisknutí klávesy Enter se spustí prohledávání).
- `python3 agentsearch.py -l ./saved_maps/experiment1_16_2.json -t smart_coop` Pro spuštění prohledávání mapy prvního experimentu se dvěma agenty typu smart_coop.
- `python3 agentsearch.py -d 10 20 -a 10 -o 50` Pro spuštění prohledávání náhodně vygenerované obdélníkové mapy o rozměrech 10x20 s deseti agenty typu 'naive' a padesáti překážkami.

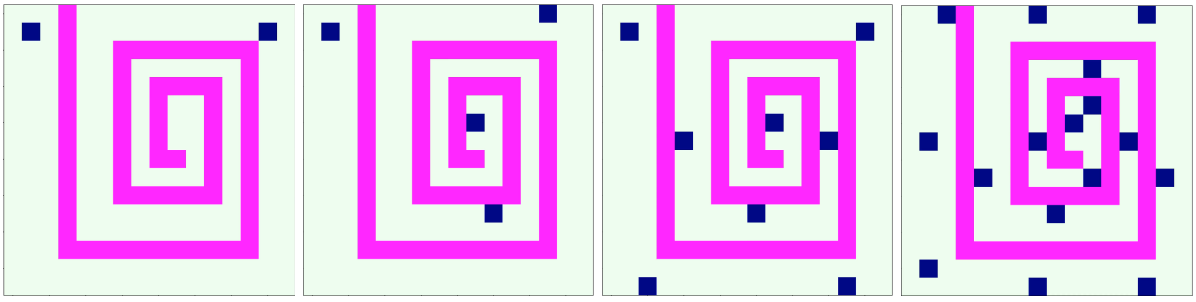
4 Vytváření mapy

Pokud se rozhodnete spustit skript příkazem `python3 agentsearch -c`, otevře se vám před spuštěním prohledávání editor mapy. Kliknutím levým tlačítkem myši na libovolné políčko na grafu změníte jeho obsazení z prázdného na překážku. Opětovným kliknutím ho změníte z překážky na agenta a dalším kliknutím z agenta opět na prázdné políčko. Stiskem klávesy "Enter"poté spustíte prohledávání této mapy. Stiskem klávesy "j"dojde k uložení mapy do složky `./saved_maps` a následnému spuštění prohledávání.

5 Experimenty

Naivní a náhodní agenti nemají deterministické chování, proto budou experimenty s nimi spuštěny 5x a výsledkem pak bude aritmetický průměr všech úspěšných prohledávání. Pokud alespoň jedno z prohledávání bude neúspěšné, za průměrem se v tabulce objeví hodnota, která v procentech udává úspěšnost agenta. Agenti smart a smart_coop mají deterministické chování, proto jsou experimenty s nimi spouštěny pouze jednou. Cílem experimentů je zjištění, kolik kroků agenti potřebují pro prozkoumání celé mapy nebo pro zjištění, že mapu nelze celou prozkoumat. Za núspěšné prohledávání mapy je považováno překročení počtu kroků rovnému trojnásobku celkového počtu políček na mapě.

5.1 Experiment 1.



Obrázek 1: Mapy použité pro 1. experiment s rozdílnými počty agentů.

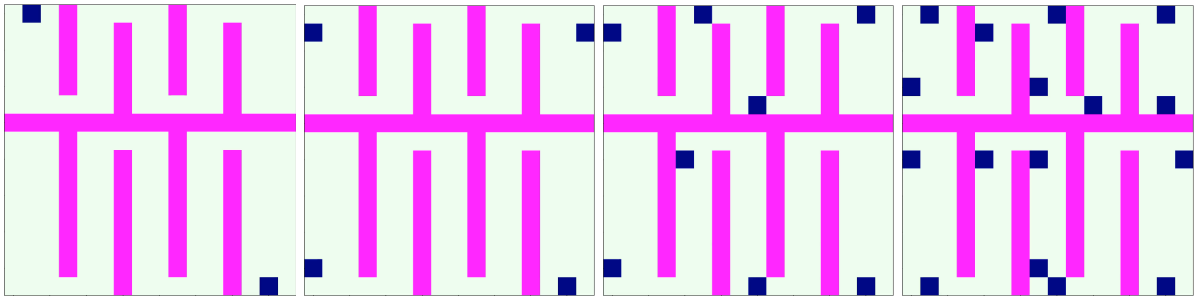
počet agentů	random	naive	smart	smart_coop
2	(0 %)	(0 %)	114	44
4	525.5 (40 %)	145.6	30	35
8	258.75 (80 %)	27.6	14	18
16	43.8	7	6	7

5.2 Experiment 2.

Obrázek 2: Mapy použité pro 2. experiment s rozdílnými počty agentů.

počet agentů	random	naive	smart	smart_coop
2	(0 %)	(0 %)	78	94
4	540 (60 %)	(0 %)	28	35
8	447.67 (60 %)	15.2	13	13
16	45.4	6	6	6

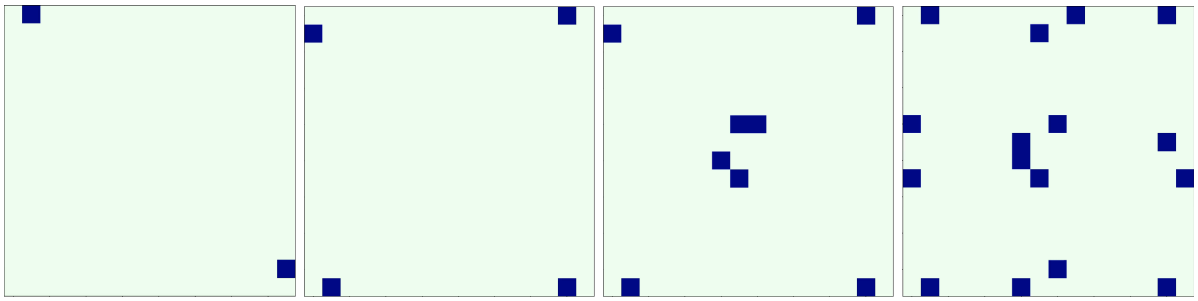
5.3 Experiment 3.



Obrázek 3: Mapy použité pro 3. experiment s rozdílnými počty agentů.

počet agentů	random	naive	smart	smart_coop
2	(0 %)	(0 %)	51	52
4	501(60 %)	(0 %)	23	23
8	73	18.8	12	11
16	22.2	4.6	4	4

5.4 Experiment 4.



Obrázek 4: Mapy použité pro 4. experiment s rozdílnými počty agentů.

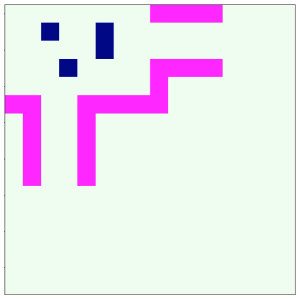
počet agentů	random	naive	smart	smart_coop
2	521.33 (60 %)	53.8	38	45
4	255.8	26.4	19	19
8	95.6	11.4	10	10
16	22.2	4	4	4

Z experimentů 1-4 lze vyvodit, že chování náhodného agenta příliš neovlivňuje koncepce mapy, avšak téměř ve všech případech dosahuje zdaleka nejhorších výsledků. Avšak především v experimentech 2. a 3. dokáže prohledat pole rychleji, než agent naivní, a to proto, že naivnímu agentovi dělá velký problém obcházení dlouhých překážek. Tento problém lze eliminovat přidáním více agentů tak, že obcházení překážek již téměř není potřeba, protože se za každou překážkou nachází jiný agent, který oblast prozkoumá. Také si lze povšimnout, že naivní agent v počtu 16 dosahoval téměř totožných výsledků jako agenti inteligentní, a to proto, že v tomto počtu agenti dokázali prohledat celou mapu téměř pouze za použití přímých pohybů.

Inteligentní agenti (smart a smart_coop) byli v každé instanci experimentu úspěšní, a to především proto, že dokáží efektivně prohledat jakoukoli oblast, která není ze všech stran obehnána stěnami. Zajímavé však je, že v drtivé většině případů agenti komunikující neporazili agenty nekomunikující. Důvodem je pravděpodobně fakt, že u map, které byly součástí experimentu se vyplatí maximalizovat lokální prohledávání a ne se příliš věnovat rovnoměrnému prohledávání celé mapy. V experimentu 1. však kooperující agenti v počtu 2 značně předčili agenty nekooperující, a to právě proto, že agent nekooperující, nacházející se v pravém horním rohu se rozhodl prohledávat stejnou část mapy, jako agent v levém horním rohu. To vedlo k tomu, že se oba agenti setkali na vnější straně spirály a museli celou spirálou projít až do středu, aby prohledali celý zbytek mapy. Agent kooperující však znal pozici i cíl agenta druhého a proto se vydal druhým směrem. Tento jediný případ je tak markantní, že ikdyž agenti nekooperující většinou dokázali mapu prohledat rychleji, agentni kooperující dokázali všechny experimenty dokončit v méně krocích. Kooperující agenti jsou celkově imunní vůči situacím, kdy si více agentů zvolí stejný cíl a proto kopírují tu samou trasu.

5.5 Experiment 5.

Pro ilustraci chování, při kterém komunikující agenti předčí ty nekomunikující jsme vytvořili ještě pátou mapu, kde (v mlaých počtech agentů) více nekooperativních agentů opět zbytečně kopíruje tu samou trasu, kdežto agenti kooperativní si práci rozdělí a prohledají mapu efektivněji.



Obrázek 5: Mapa použitá pro 5. experiment se čtyřmi agenty.

počet agentů	smart	smart_coop
2	86	59
4	45	31
8	42	43
16	25	30

Z toho vyplývá, že ačkoli jsou agenti nekooperující mnohdy o něco rychlejší, v některých případech (zejména s omezeným počtem agentů) jejich chování může vést k volbě velice neefektivní strategie, kdy je agenti kooperující předčí ve své konzistentnosti.

6 Závěr

Výsledkem práce je několik různých agentů, z toho dva vykazující jistou míru iteligence. Při jejich porovnání však není jasné, který z nich dosahuje lepšího výkonu. Pravděpodobně by jejich výkon bylo možné optimalizovat určitou kombinací jejich chování. Další možnost optimalizace kooperujícího agenta by bylo nastavení parametru, jak vzdálené musí být cíle agentů od jeho možných cílů, aby nad nimi uvažoval. Aktuální implementace momentálně počítá se vzdáleností 2x vzdálenost dohledu agenta. Další možné vylepšení agentů by bylo plánování cesty delší, než pouze na jediné políčko. U toho by však byla problematická neznalost neobjevených políček a plánování za objevená políčka by muselo být do jisté míry stochastické. Jistě by bylo možné aplikovat na rozhodovací proces agenta neuronové sítě, avšak zde by byl jejich přínost velice diskutabilní.