Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

DIPLOMOVÁ PRÁCE

Umělá inteligence pro závodní hru Trackmania

2024

Bc. Ivana Jelínková

Vedoucí práce:
Mgr. Radek Janoštík, Ph.D.

Studijní program: Informatika,
Specializace: Umělá inteligence

Bibliografické údaje

Autor:	Bc. Ivana Jelínková
Název práce:	Umělá inteligence pro závodní hru Trackmania
Typ práce:	diplomová práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2024
Studijní program:	Informatika, Specializace: Umělá inteligence
Vedoucí práce:	Mgr. Radek Janoštík, Ph.D.
Počet stran:	32
Přílohy:	elektronická data v úložišti katedry informatiky
Jazyk práce:	český

Bibliographic info

Author:	Bc. Ivana Jelínková
Title:	Artificial intelligence for the racing game Trackmania
Thesis type:	master thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2024
Study program:	Computer Science, Specialization: Artificial Intelligence
Supervisor:	Mgr. Radek Janoštík, Ph.D.
Page count:	32
Supplements:	electronic data in the storage of department of computer science
Thesis language:	Czech

Anotace

Práce pojednává o vytvoření a naučení počítačem řízených hráčů založených na různých přístupech umělé inteligence pro závodní hru Trackmania. Dále se práce zabývá srovnáním použitých přístupů a jejich testováním.

Synopsis

The thesis is about the creation and learning of computer-controlled players based on different artificial intelligence approaches for the racing game Trackmania. Furthermore, the work compare the used approaches and their testing results.

Klíčová slova: Trackmania; umělá inteligence; umělé neuronové sítě; genetické algoritmy; Bayesovské sítě

Keywords: Trackmania; artificial intelligence; artificial neural network; genetic algoritms; Bayesian network

Děkuji panu Mgr. Radkovi Janoštíkovi, Ph.D. za vedení práce a investovaný čas. Také děkuji rodině a přátelům za velkou podporu.

Odevzdáním tohoto textu jeho autor/ka místopřísežně prohlašuje, že celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

Obsah

1 Trackmania **7**
1.1 Ohodnocení projetí tratě . . . 7
1.2 Tvorba závodní mapy . . . 7
1.3 Verifikace závodní mapy . . . 8
1.4 Fyzika . . . 8
1.5 Ovládání . . . 8

2 Komunikace hráče a hry **8**
2.1 Informace přímo ze hry . . . 8
2.2 Vize . . . 9
2.3 Řízení formule . . . 10

3 Typy umělé inteligence **11**
3.1 Umělá neuronová síť . . . 11
3.2 Genetické algoritmy . . . 13
3.2.1 Výběr ruletou . . . 13
3.3 Bayesovská síť . . . 14
3.3.1 Belief propagation . . . 15
3.4 Fuzzy řídící systém . . . 16

4 Tvorba umělé inteligence **17**
4.1 Umělá neuronová síť . . . 17
4.2 Genetické trénování . . . 18
4.3 Bayesovské sítě . . . 20
4.4 Fuzzy systém . . . 22

5 Trénování a experimentování **24**
5.1 Trénování AI . . . 24
5.2 Testování . . . 25

Závěr **29**

Conclusions **30**

A Obsah elektronických dat **31**

Literatura **32**

Seznam tabulek

1 Pravděpodobnost pozorování nejkrajnějších vzdáleností, pokud se zatáčí 22
2 Pravděpodobnosti pozorování krajních vzdáleností, pokud se zatáčí 22
3 Data z testování 27

1 Trackmania

Série závodních her Trackmania započala stejnojmenou hrou vydanou roku 2003. Od publikování prvního titulu byla série obohacena o dalších 18 her. Tato práce se zabývá pouze titulem Trackmania (2020) publikovaným 1. července roku 2020 pod vydavatelem Ubisoft. Cílem hry je dojet do cíle se svojí formulí v nejkratší možném čase. Je možné hrát v módu solo nebo v módu multiplayer, kdy může hráč soupeřit online s dalšími účastníky. Hráč však může nejen závodit, ale taktéž sám vytvářet závodní mapy. Trackmania (2020) je zdarma přístupná, přesto je zde placený segment jež dává přístup k oficiálním kampaním, které byly kdy vydány. Nabízí i možnost se každý den zůčastnit závodů na tratích od hráčů, které zaštiťuje Ubisoft Nadeo nebo je zde i příležitost se angažovat v oficiálních esportových událostech. Placená verze tedy není pro tuto práci nijak potřebná.

1.1 Ohodnocení projetí tratě

Každé projetí tratě za nějaký čas může být ohodnoceno medailí, pokud je splněno její časové rozpětí dojetí do cíle. Medaile jsou přístupné v typech bronzová, stříbrná, zlatá a autorská, v pořadí od nejméně hodnotné k nejhodnotnější. Celkově jsou všechna ohodnocení na online tratích veřejná a jezdí se tedy stylem, aby se hráč dostal do cíle v co nejkratším čase. Na online mapách platí striktní pravidla ohledně podpůrných programů pro hráče a jejich přispívání v žebříčcích vítáno není. Účty využívající podporu takového charakteru by mohly být smazány. Mezi toto omezení je zahrnuta i umělá inteligence. Hráči založeni na umělé inteligenci, dále jen umělí hráči, kteří jsou výsledkem této práce, nejsou k použití na online mapách.

1.2 Tvorba závodní mapy

Pro vytvoření mapy se nabízí možnost jednoduchého nebo pokročilého editoru.

V jednoduchém editoru jsou k dispozici 4 druhy cest: asfalt, štěrk, led a vypouklý asfalt. Každý druh má následující bloky ve svém provedení, které se rozdělují na bloky určující trasu a bloky ovlivňující chování. Bloky určující trasu jsou: start, cíl, bod pro povinné projetí, start a cíl v jednom, rovná cesta, zatáčka 90 stupňů na 1×1 bloku, zatáčka 90 stupňů na 2×2 bloky, zatáčka 90 stupňů na 3×3 bloky, bloky měnící výšku dráhy o různé rozsahy.

Bloky ovlivňující chování: zrychlení, zvednutí doletové vzdálenosti při skoku, vypnutí motoru, vypnutí zatáčení, zpomalení času, zkřehlá formule, uvedení formule do původního stavu.

Pokročilý editor nabízí jako první změnu nasvícení mapy, na výběr je východ slunce, den, západ slunce nebo noc. Nabídka stavebních bloků je velmi rozmanitá a pro přehled jsou rozděleny do podsložek. Je na výběr z dalších povrchů jako jsou tráva, voda a plast. Je zde i více bloků pro určení trasy, další bloky ovlivňující chování a celkem jich dohromady čítá přes 3000.

1.3 Verifikace závodní mapy

Každou mapu před zveřejněním je třeba ověřit, že existuje možnost ji dokončit, tedy dojet s formulí ze startu do cíle. Autor mapy musí trať projet a jeho čas projetí, se kterým je spojen, bude uveden jako meze pro autorskou mediali. Verifikace je nutná při každé změně mapy. Ověřování mapy se tváří jako klasické ježdění po trati, ale je mimo online prostor, proto je možné toto využít právě k trénování nebo testování umělé inteligence.

1.4 Fyzika

Chování formule podléhá základním principům fyziky. Fyzikální engine zahrnuje hybnost, setrvačnost a zachování energie podle principů Newtonovy mechaniky. Navíc má každý terén svou specifickou trakci, není to jen dekoračním prvkem. Příkladem je štěrk, který má větší sklony k prokluzům a driftu než asfalt. Dostředivá nebo odstředivá síla má své místo v zatáčkách. Srážka s překážkou nebo zdí mění rychlost a směr formule. Skoky berou v potaz rychlost, úhel a odpor vzduchu. Voda zde také hraje roli, pokud formule projede vodou, jsou její pneumatiky mokré a tato skutečnost se odráží na trakci, dokud nejsou postupně zbaveny vody třením o povrch.

1.5 Ovládání

Hráč může používat k ovládání formule šipky na klávenici, gamepad, joystick nebo volant. Formule má obecně možnost kromě akcelerace a změny směru i možnost brždění. Zastaví-li formule úplně, pak se brzda překlopí na zpáteční převod. Umělí hráči budou mít k dispozici právě šipky pro akceleraci a zatáčení.

2 Komunikace hráče a hry

Pro komunikaci hry s umělým hráčem je využita skriptovací platformu OpenPlanet [1] ve verzi 1.26.17 a pro další informace vyvolává umělý hráč snímek obrazovky, který je dále procesován. Umělý hráč komunikuje zpět s hrou pomocí vyvolávání událostí stisknutí různých tlačítek na klávesnici.

2.1 Informace přímo ze hry

OpenPlanet je nástroj dovolující spouštět skripty v jazyce AngelScript (AngelCode Scripting Library) přímo ve hře. Skripty, které jsou podepsané OpenPlanet lze spouštět bez omezení. Nepodepsané skripty mohou být spuštěny pouze pod módem Developer, který je přístupný jen účtům s členstvím Trackmania Club (placená verze), bohužel se jinak k módu Developer nedá dostat[1]. Pokud by byl

[1]Informace přímo od vývojáře OpenPlanet

případně k dispozici pro tuto práci mód Developer, bylo by nutné dát nově vytvořený skript ke kontrole a podepsání týmu OpenPlanet, aby bylo možné jej spouštět i mimo účty spadající pod členství. OpenPlanet tímto způsobem hlídají distribuci podpůrných programů při legitimních závodech a chrání kontrolou i další hráče. Východiskem bylo nalezení již podpsaného skriptu, který otevírá komunikaci na localhostu na portu 9000 a zasílá pole o 11 hodnotách, z nichž je pro tuto práci využito 6, výčtově: rychlost, pozice na ose x, pozice na ose y, pozice na ose z, vstupní hodnota zatočení, pravdivostní hodnota projetí cílem. Skript nese název TMRL_GrabData (verze z 5.3.2023, podpis 6.3.2023) a je využit v knihovně TMRL [2] (licence MIT). V rámci práce byla vytvořena třída GameGetter napojující se na otevřenou komunikaci s cílem získávat výše zmíněné informace. Ve třídě běží zvlášť vlákno, které je po celou dobu napojeno na hru, nedochází tak k prodlevám způsobeným opětovnými pokusy navázat komunikaci a získat hodnoty ze hry.

2.2 Vize

Získání informací přímo ze hry není úplné, aby bylo možné pro umělého hráče adekvátně reagovat na okolí. Počítačové vidění je rozsáhlý obor sám o sobě. Pro tuto práci je řešením využití analýzy obrazu – hledání černé a vzdálenost k ní od určitého bodu. Udělá se základní snímek obrazovky na mapě, která tvoří ideální scénu pro bázi. Jedná se o je jeden blok tratě a dál je ohraničen čistě černou viz. obrázek 1. Na tento bázový snímek je aplikována maska, propouštějící jen černou barvu, ostatní je nahrazeno bílou (obrázek 2). Upravený snímek dále projde procesem zjištění vzdáleností od bodu definujícího vizi formule, souřadnice tohoto bodu jsou vypočítány dle vzorců s ohledem na možnou proměnlivost velikosti snímaného okna hry. Pro výpočet souřadnice y centrového bodu je následující vzorec

$$y = Y \times 0.889$$

kde Y je velikost snímku obrazovky v ose y. Pro souřadnici x centrového bodu

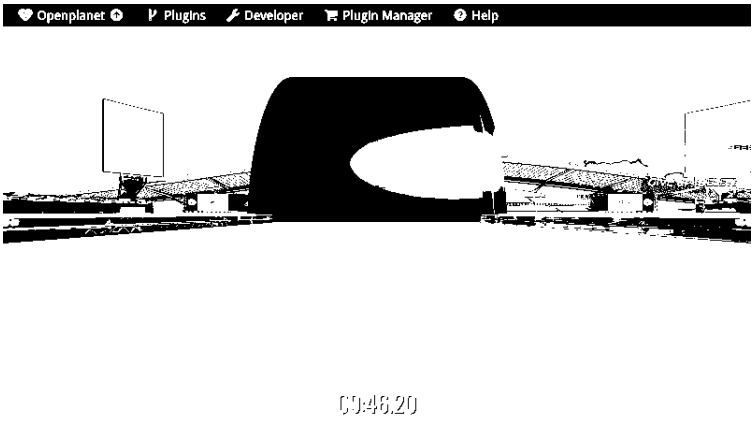
$$x = X \div 2$$

kde X je velikost snímku obrazovky v ose x.

Dále se vygeneruje 17 hodnot rovnoměrně rozprostřených na intervalu $\langle 0; \pi \rangle$, na tyto hodnoty je aplikována funkce tangens a výsledkem jsou směrnice přímek mapujících okolí formule. Tyto směrnice jsou využity v algoritmu DDA (digital differential analyzer), jenž se klasicky používá na rasterizaci úsečky. Zde plní fukci průchodu po směru úsečky s cílem najít první výskyt černé barvy a navrátit její velikost. Velikosti úseček můžeme nazvat bázové velikosti. Hledání černé barvy do daných směrů je zobrazeno na obrázku 3. Při každém snímku pro vizi hráče se pak program chová stejně, jen je ve finále velikost úsečky ještě vydělena bázovou velikostí úsečky stejné směrnice. Hráč by měl směřovat k největším velikostem prostředních bodů, neboť tímto směrem vede trať dál. Název point_x

je odvozen od hledání bodu černé barvy, ale je to vždy vzdálenost od bázového bodu k prvnímu bodu černé.

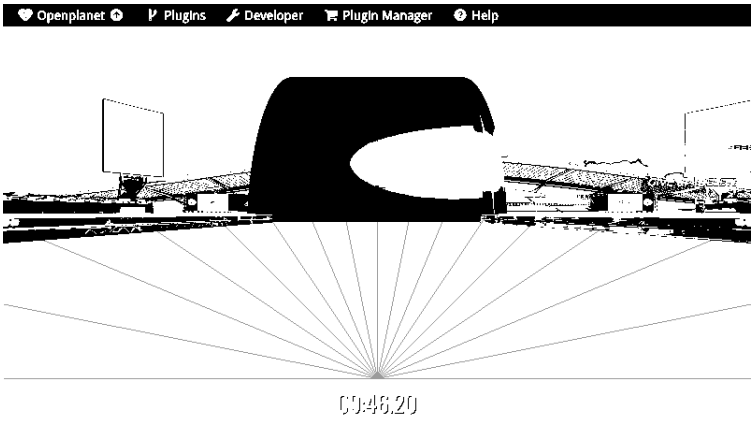
Obrázek 1: Základní snímek hry



Obrázek 2: Snímek s maskou

2.3 Řízení formule

Umělý hráč má přímé spojení se hrou ve směru ke hráči, ale není k dispozici skript opačného rázu. Řešením je tedy simulace vstupů jako od lidského hráče. Byly vytvořeny tedy dvě třídy k vyvolávání událostí stisknutí kláves, jedna pro operační systém Linux a jedna pro operační systém Windows. Každá třída využívá jinou knihovnu, ale názvy metod jsou totožné pro snadné přenášení mezi systémy. Klávesovými příkazy lze v Trackmanii i ukládat proběhlý závod nebo závod přerušit a spustit znovu. Třídy kromě ovládání formule disponují i metodami pro využití těchto možností.



Obrázek 3: Snímek s úsečkami od centrového bodu

3 Typy umělé inteligence

Umělá inteligence je systém nebo algoritmus vykonávající úlohy, které by jinak potřebovaly lidskou inteligenci. Existuje mnoho různých přístupů umělé inteligence, jenž mají silné a slabé stránky pro určité využití. Pro práci byly vybrány takové, které jsou často skloňovány, co se autonomního řízení týče a jsou tedy pro tuto problematiku použitelné.

3.1 Umělá neuronová síť

Umělé neuronové sítě [4] se skládají z umělých neuronů a ty se inspirovaly u biologických neuronů. Biologický neuron je buňka, která přijímá a vysílá elektrické signály. Umělý neuron se skládá ze vstupních signálů $x, x_i \in \mathbb{R}$, vah $w, w_i \in \mathbb{R}$, prahu $\theta \in \mathbb{R}$ a výstupního signálu $y \in \{0, 1\}$. Vstup přichází po spojích, které mají své váhy. Každý vstupní signál jde po svém spoji a je násoben jeho vahou. Následně jsou všechny sečteny a porovnány s prahem, zda neuron zahoří (vyšle signál dál).

$$y = \begin{cases} 1, & \Sigma(x_i w_i) \geq \theta \\ 0, & \text{jinak} \end{cases}$$

Tento jednoduchý model neuronu se nazývá perceptron a dají se jím modelovat například logické operace jako je konjunkce nebo implikace. Jeho omezením je schopnost se naučit pouze lineárně separabilní množiny. Omezení řeší sestavení sítě z perceptronů, zde se naráží na problém nenalezeného učícího algoritmu. Nejčastěji využívaná neuronová síť je vícevrstvá s dopředným šířením. Jejím základem je spojitý perceptron skládající se ze vstupních signálů $x, x_i \in \mathbb{R}$, vah $w, w_i \in \mathbb{R}$, prahu $\theta \in \mathbb{R}$, potenciálu neuronu $z \in \mathbb{R}$, aktivační funkce f a výstupního signálu $y \in \mathbb{R}$. Základem je výpočet potenciálu

$$z = (\Sigma(x_i w_i)) - \theta$$

který dále figuruje ve výpočtu výstupu

$$y = f(z)$$

Funkcí, které lze dosadit za f je na výběr mnoho. Velmi oblíbená je funkce ReLU [7]. Pro tuto práci byla brána v úvahu v začátcích, ve finále byl použit hyperbolický tangens (tahn)

$$ReLU : f(x) = \frac{x + |x|}{2}$$

$$Tahn : f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Síť má libovolný počet vstupů, výstupů, vrstev, počet neuronů ve vrstvách a výstup každého neuronu vrstvy $l - 1$ je vstupem každého neuronu vrstvy l, tedy propojení každého s každým. Obecně je síť definována počtem vrstev r, počtem výstupů n, počtem vstupů m, počtem neuronů ve vrstvě, a reptezentuje zobrazení

$$F : \mathbb{R}^m \to \mathbb{R}^n$$

Síť se učí úpravou vah a prahu. Na určení velikosti sítě[2], aby byla schopna se naučit požadované, neexistuje jednoduchý a univerzální vzorec či algoritmus. Jsou dvě metody přístupu zjištění optimální velikosti pro daný problém: od velké k malé nebo od malé k velké. Velká naučená sít se ořezává a je znovu podrobena učení, zda je schopna zvládnout úkol, toto se opakuje dokud je síť možné naučit. V opačném případě se vezme malá síť, pokud je schopna se naučit, pak je odpovědí, jinak je zvětšena a cyklus probíhá znovu.

Učení vícevrstvé sítě je nejčastěji asociováno s učícím algoritmem Backpropagation, pojmenovaném podle zpětného šíření chyby. Tento algoritmus je typu učení s učitelem[3]. Avšak toto není jediná možnost, jak naučit neuronovou síť požadované chování. Aktuálně je nejvíce řešeno zpětnovazební učení[4], jež funguje na základě odměny. V případě této práce, by byl umělý hráč odměňován za postup v trati a trestán například za ztrátu rychlosti nebo vyjetí mimo trať. Touto metodou přímo pro Trackmanii 2020 se již zabývali pan Bouteiller s panem Geze [2], kteří vytvořili knihovnu tmrl přímo pro trénování vlastních umělých hráčů. Právě pro tuto knihovnu byl skript, který je zmíněn v komunikaci, vytvořen. Proto se práce ubrala odlišným směrem a tím bylo pokusit se vytvořit neurovonou síť zvládající dojet do cíle pomocí genetických algoritmů a tedy naučit se bez učitele[5].

[2] počet vrstev a počet neuronů v každé vrtvě
[3] Supervised learning
[4] Reinforcement learning
[5] Unsupervised learning

3.2 Genetické algoritmy

Podobně jako dříve zmíněné umělé neuronové sítě i genetické algoritmy přejímají procesy z biologie. Biologická evoluce je na principu přirozeného výběru. Pokud má jedinec vlastnosti, které jsou výhodou k přežití, pak je upřednostněn. Navíc je možnost, že jedinec zmutuje a tím získá novou vlastnost, která může i nemusí být ku jeho prospěchu. Křížením jedinců vznikají noví jedinci mající různý poměr informací přejatých od rodičů. Tyto principy lze využít u umělé evoluce, kdy existuje představa řešení, například dojetí formule do cíle, a množina možných řešení prochází evolucí. V tomto případě populace umělých hráčů tvořených umělými neuronovými sítěmi a snaží se tomuto řešení přiblížit.

Pro genetický algoritmus je potřeba specifikovat kódování řešení. Klasicky jsou to řetězce bitů podle evoluce genů - chromozomů. Umělou neuronovou síť je snadné serializovat, od první vrstvy po poslední, v každé vrstvě od prvního neuronu po poslední, získáme řetězec neuronů reprezentující celou síť a takto je možné je mutovat a nebo křížit. Další potřebnou definicí je fitness funkce a metoda výběru. To umožňuje ohodnotit jak si dané řešení vedlo a přiřadit mu skóre, jenž rozhoduje o pravděpodobnosti výběru do další generace a taktéž do výběru ke křížení. Je nutné uvést jak se chová mutace a křížení a kdy má algoritmus skončit.

Jsou-li všechny tyto aspekty definovány, pak je možné spustit hledání řešení pomocí genetického algoritmu. V aktuální populaci se její jedinci ohodnotí, metodou výběru se zvolí jedinci, co postupují do populace další generace a při jejich postupu mohou zmutovat. Protože nemuseli být vybráni všichni jedinci, pak se ze všech jedinců z aktuální populace pomocí křížení vytvoří noví jedinci postupující do další generace, aby byla velikost populace stále stejná v každé generaci. Toto se opakuje dokud není splněna ukončovací podmínka. Ukončovací podmínka může být, že se jedinci dostatečně přiblížili k řešení nebo počet generací, jenž musí proběhnout.

3.2.1 Výběr ruletou

Výběr ruletou se využívá v genetických algoritmech pro výběr jedinců pro křížení. Ruleta se vytvoří ze všech ohodnocených jedinců, velikost zabraného místa na ruletě jedincem záleží na dosaženém skóre v generaci - čím lepší řešení, tím větší část rulety pokrývá. Suma celé rulety je 1. Výběr probíhá vygenerováním čísla z intervalu $\langle 0; 1\rangle$ a od místa, které bylo naposledy vybráno se postupuje po ruletě a přičítají se jejich pravděpodobnosti dokud je suma menší než vygenerované číslo. Ve chvíli kdy podmínka již neplatí, je vybrán jedinec, který podmínku svou hodnotou porušil.

3.3 Bayesovská síť

Bayesovská síť [5] je pravděpodobnostní orientovaný acyklický graf $G = (V, E)$, každý uzel $i \in V$ odpovídá proměnné X_i s konečným počtem navzájem disjunktních hodnot $\mathbb{X}_i$. Graf reprezentuje podmíněné nezávislostní vztahy mezi proměnnými pomocí orientovaných hran. $Parents(i)$ je označení pro množinu rodičů uzlu i a každému uzlu $i \in V$ odpovíná podmíněná pravděpodobnostní distribuce $P(X_i|(X_j)_{j \in Parents(i)})$. Výpočet podmíněné závislosti daných proměnných je založen na Bayesově větě.

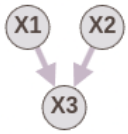
Věta 1 (Bayesova věta)
Máme-li dva náhodné jevy A a B, jejichž pravděpodobnosti jsou P(A) a P(B), a pokud P(B) > 0, potom platí

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Definice 2 (Bayesovská síť)
Nechť $X = (X_1, ..., X_n)$ jsou náhodné proměnné, x_i jsou hodnoty, kterých X_i nabývají

$$\mathbb{P}(X_1 = x_1, ..., X_n = x_n) = \prod_{i=1}^{n} p(x_i|x_{Parents(i)})$$

Graf je definován zápisem, pro $\mathbb{P}(X_1 = x_1, X_2 = x_2, X_3 = x_3) = p(x_1)p(x_2)p(x_3|x_1, x_2)$ odpovídá graf 4. Síť se tvoří na základě dat nebo za asistence experta.



Obrázek 4: Graf Bayesovské sítě

Máme-li data o pravděpodobnostech jednotlivých proměnných, pak položením dotazu (query) můžeme odvodit pravděpodobnosti př.: $\mathbb{P}(X_1)$?. V dotazu můžeme uvést i předpoklady př.: $\mathbb{P}(X_1|X_3 = 1)$?. Brute-force řešení dotazu: Pokud je přímo explicitně zadáno, že se proměnná rovná určité hodnotě, pak ji nazveme jako pozorovanou proměnnou. Pro každou proměnnou X_i pro kterou nebyla uvedena hodnota explicitně x_i a která figuruje v závislosti pro výpočet je nutné projít všechny možné hodnoty proměnné $\mathbb{X}_i$, kterých může nabývat a dosadit je do vzorce definujícího síť. Pro síť uvedenou jako příklad je výpočet $\Sigma_{x_1 \in \mathbb{X}_1, x_2 \in \mathbb{X}_2, x_3 \in \mathbb{X}_3} p(x_1)p(x_2)p(x_3|x_1, x_2)$, kde explicitně definovaná hodnota pro x_i nahradí množinu možných hodnot $\mathbb{X}_i$, například pro explicitně definovanou hodnotu $x_3 = y$ je výpočet $\Sigma_{x_1 \in \mathbb{X}_1, x_2 \in \mathbb{X}_2, x_3 = y} p(x_1)p(x_2)p(x_3|x_1, x_2)$. Časová složitost je $O(S^N)$, kde S je počet dvojic proměnných.

3.3.1 Belief propagation

Belief propagace je algoritmus předávající zprávy pro odvozování v grafických modelech jako je Bayesova síť. Algoritmus počítá marginální pravděpodobnosti rozdělení pro každou nepozorovanou proměnnou, závisle na pozorovaných proměnných.

Pravděpodobnost pozorování hodnoty x_i pro proměnnou X_i značíme L. Pro proměnnou, která nemá žádné potomky a není pozorována, je pravděpodobnost $L_{x_i} = 1$ pro $\forall x_i \in \mathbb{X}_i$. Pokud je pozorována jedna z hodnot, pak $L_{x_j} = 1$ pro pozorovanou hodnotu x_j a $L_{x_i} = 0$ pro $\forall x_i \in \mathbb{X}_i \backslash x_j$. Má-li proměnná potomky, pak po nich požaduje zprávy o jejich pravděpodobnosti L:

$$L(X) = \prod_{K} \lambda_{(K \to X)}$$

kde K je potomek, X aktuální proměnná a λ je zpráva. Prior pravdědopodobnosti proměnných nemající rodiče je mají dané od začátku. Jsou jimi jejich vlastní základní pravděpodobnosti, protože na jejich pravděpodobnost nemá žádný rodič vliv. Pro ostatní proměnné je třeba prior pravděpodobnosti dopočítat:

$$\pi(X) = \sum_{W} P(X|W) \prod_{K} \phi_{(K \to X)}$$

kde K je rodič, X aktuální proměnná a ϕ je zpráva. Suma pravděpodobností všech možných kombinací hodnot rodičů je násobena zprávami od rodičů. Víra, podle které je algoritmus nazván, je normalizovaná porsterior pravděpodobnost po pozorování proměnných.

$$\beta(X) = \alpha L(X)\pi(X)$$

α představuje normalizaci. Zpráva zasílaná rodičům je pro výpočet pravděpodobnosti L uzlu rodiče, který o ni požádal potomka. Potomek, aby mohl odeslat zprávu, se musí zeptat svých potomků na jejich pravděpodobnosti L, pokud nějaké má. Taktéž je třeba se otázat svých další rodičů, pokud nějaké má, na jejich pravděpodobnosti. Zasílání zprávy od potomka k rodiči definujeme:

$$\lambda_{(K \to X)} = \sum_{x \in X} L(x) \sum_{k \in K; k \in u} P(x|u) \prod_{i \neq k} \phi_{(Y \to X)i}$$

kde X je potomek zasílající zprávu, K je rodič, který o zprávu požádal, U jsou všichni rodiče X. Zpráva pro potomka má možnost dvou tvarů:

$$\kappa_{(X \to K)} = \alpha \prod_{C \backslash K} \lambda_{(C \to X)} \pi(X) = \alpha \frac{\beta(X)}{\lambda_{(K \to X)}}$$

kde X je rodič zasílající zprávu, K je potomek, α představuje normalizaci. Výpočet tedy spočívá v určení pozorovaných proměnných. A dále se dotážeme na „víru“ pravděpodobnosti požadované proměnné. Bayesovská síť se začne dotazovat na pravděpodobnost L a na prior pravděpodobnosti. V závislosti na velikosti a vazbách v síti se šíří zprávy a ve chvíli, kdy výpočet skončí je zodpovězen dotaz. Časová složitost je $O(S^2)$, kde S je počet dvojic proměnných.

3.4 Fuzzy řídící systém

Fuzzy řídící systém [6] je založen na fuzzy logice a fuzzy množinách. Oproti matematické logice namísto ohodnocení $\{0, 1\}$, disponuje funkcí příslušnosti k množinám v intervalu $\langle 0; 1\rangle$, kde 0 je absolutní nepříslušnost a 1 absolutní příslušnost. Fuzzy množina je množina prvků, které jsou výstupy funkce příslušnosti vůči množině a určují, nakolik prvek do množiny přísluší. Nad množinami lze provádět operace průnik, sjednocení a komplement.

$$\text{sjednocení: } m_{A\cup B}(p) = max\{m_A(p), m_B(p)\}$$

$$\text{průnik: } m_{A\cap B}(p) = min\{m_A(p), m_B(p)\}$$

$$\text{komplement: } m_{\neg A}(p) = 1 - m_A(p)$$

kde $m_X(p)$ je funkce příslušnosti prvku p k množině X.

Řídící systém kontroluje proces tím, že naslouchá na jeho výstupu, což je pro systém vstup a pomocí vnitřního výpočtu upraví vstup do kontrolovaného procesu, což je výstupem systému. Protože vstup do systému není fuzzy, musí být fuzzifikován. Pak je dále předán do fuzzy pravidlového systému, ke kterému náleží pravidlová báze a odvozovací procedura. Po dokončení je výpočet defuzzifikován a aplikován na vstupu do kontrolovaného procesu.

Fuzzifikace transformuje prvek $x_i \in X_i$ na fuzzy množinu $A_i : X_i \to [0, 1]$. Příkladem přístupů jsou singleton, triangular a další.

$$\text{Singleton: } m_X(p) = \begin{cases} 1, & p = x_i \\ 0, & \text{jinak} \end{cases}$$

Pro funkci příslušnosti typu Triangular je třeba určit a jako minimum, kdy začíná příslušnost, b jako bod absolutní příslušnosti a c jako maximum, kdy příslušnost končí.

$$\text{Triangular: } m_X(p) = \begin{cases} 0, & p \leq a \\ \frac{p-a}{b-a}, & a \leq p \leq b \\ \frac{c-p}{c-b}, & b \leq p \leq c \\ 0, & p \geq c \end{cases}$$

Pravidlová báze obsahuje pravidla ve formátu POKUD-TAK. Obecná forma pravidla je

$$R_i : \text{IF } x_1 \text{ is } A_1 \text{ \&... \& } x_n \text{ is } A_n \text{ THEN } y \text{ is } B_i$$

kde A_i a B_i jsou slovní výrazy pro příslušnost. Pravidlová báze je strategie řešení problému. Metoda pro odvození výstupu aplikuje všechna pravidla z báze na fuzzifikovaný vstup systému a následně jejich jednotlivé výsledky B_i sjednotí. Tímto opouští výsledek pravidlový systém. Je nutné jej nyní defuzzifikovat. Nejpoužívanější defuzzifikační funkcí je COG (center of gravity)

$$y = \frac{\sum_{k=1}^{K} B(y_k) y_k}{\sum_{k=1}^{K} B(y_k)}$$

kde y_j reprezentuje diskretizaci Y. Další defuzzifikační funkcí jsou například mean of maxima, center of maxima. Defuzzifikovaný výsledek je předán na výstup řídícího systému pro úpravu vstupu.

4 Tvorba umělé inteligence

Pro tvorbu všech uvedených umělých inteligencí byl využit programovací jazyk Python a jeho balíčky. Seznam všech balíčků je součástí README.

4.1 Umělá neuronová síť

Umělá neuronová síť vytvořena pro tuto práci se tvoří zadáním počtu vstupů a definicí vrstev s počty neuronů pro každou vrstvu. Všechny váhy jsou náhodně vygenerovány z intervalu $\langle -2; 2 \rangle$. Byla přidána i možnost síť ukládat do souboru a zpětně ji načítat. Vrstva se stará o sdružování svých neuronů, předává jim vstupy z předchozí vrstvy a jejich výstupy zasílá další vrstvě, část funkcionality lze vidět na kódu 1. Síť je situovaná na učení se genetickými algoritmy. Neuron disponuje aktivační funkcí a stará se o svou mutaci. V neuronu jsou uloženy váhy a bias. Vzhledem k nedostatku informací ohledně propojení umělých neuronových sítí a genetických algoritmů nebylo možné dohledat, jak přesně by se měly sítě chovat v souvislosti s mutací a křížením. Míra mutace je definována při zadávání trénování a určuje pravděpodobnost mutace. Každá hodnota v neuronu má malou procentuální šanci pro mutaci definovanou mírou mutace. K hodnotě určené k mutaci se přičte nově náhodně vygenerovaná hodnota z intervalu $\langle -2; 2 \rangle$. Genetické algoritmy jedince kódují jako řetězce bitů, ty lze reprezentova řetězci neuronů. Síť lze snadno serializovat pomocí pořadí, které mají vrstvy a neurony v nich. Do nové sítě se kopírují rovnou celé neurony a je to založeno na určení pivotu, který určí rozdělení sítě na 2 části. Každá část dědí z jiného rodiče. Proběhl pokus s náhodným děděním, kdy rodiče měli u každého neuronu 50% šanci, že právě z nich budou váhy a bias zděděny. Sítě se však ani po několika generacích příliš nelepšily.

```
class Layer:
   def __init__(self, num_input, num_neurons, front_layer):
      self.front_layer = front_layer
      self.next_layer = None
      self.neurons = np.empty(num_neurons, dtype=object)
      for i in range(num_neurons):
         self.neurons[i] = Neuron(num_input, random.uniform(-2, 2))
   def propagate_forward(self, inputs):
      output = []
      for n in self.neurons:
         output.append(n.activation_function(inputs))
      if self.next_layer is None:
         return output
      return self.next_layer.propagate_forward(output)
```

Zdrojový kód 1: Konstruktor třídy Layer a metoda propagate_forward

4.2 Genetické trénování

Metoda pro genetické trénování přijmá jako argumenty počet generací pro ukončovací podmínku, pravděpodobnost mutace, velikost populace, prvotní populaci a volitelné argumenty pro počátek v jiné generaci než je 0 s již vypočítanými skóre pro celou generaci. Před začátkem trénování je třeba trasu projet lidským hráčem za běhu promoDrive. PromoDrive je skupina funkcí, které se starají o sběr dat z trasy, jejich čištění a uložení pro použití ke trénování a případné srovnání. Zpracovaná data tvoří linii bodů tratě, které přesně kopírují trasu projetou lidským hráčem. Jedinci v trénovacím algoritmu jsou ohodnoceni na základě těchto bodů, které byli schopni posbírat svým okolím. Jeden díl trasy má přibližnou velikost od stěny ke stěně 23 bodů. Aby měla formule možnost posbírat veškeré body určující trať, pak poloměr okolí musí být přibližně tato hodnota. Tento poloměr je brán jako obecný a slouží jen pro informaci ohledně posunu na trati. Je zde zapracován hodnotící poloměr, který je menší a na základě počtu bodů sebraných tímto okolím se počítá skóre jedince. Hodnotící poloměr je ve výchozím stavu nastaven na 11 bodů a je možné jej měnit dle potřeby tréninku. Neuronová síť může pokračovat v jízdě na trati, dokud neprojela cílem a nebo dokud každé 4 sekundy sebere nový bod okolím velikosti obecného poloměru. Po zaseknutí se na místě nebo projetí cílem se odešle dosažené skóre. V hlavní trénovací metodě je vypočten i čas, jak dlouho umělá inteligence jela. Pokud došlo k zaseknutí, tak díky prodlevě se čas chová jako penalizace. Po dojetí celé generace začíná výpočet. Vypočítá se rychlost pro každou umělou inteligenci, pak se vypočítají pravděpodobnosti postupu do další generace:

$$p(AI) = \frac{\text{rychlost AI}}{\text{maximální rychlost AI v aktuální generaci}} * 0.3 + \frac{\text{skóre AI}}{\text{maximální dosažitelné skóre}} * 0.7$$

Vygeneruje se náhodné číslo z intervalu $\langle 0; 1\rangle$ a pokud je menší než p(AI), pak AI postupuje do další generace. Po protřídění přichází na řadu tvorba potomků. Pravděpodobnosti pro výběr do další generace se znormalizují a vytvoří se pole fungující jako ruleta. Přímo metodou color_wheel_pick se vybírá rodič, ukázka kódu 2. Potomci se tvoří dokud není populace na požadované kvantitě. Pak se přechází do další generace. Pro přehled o tréninku se vypisuje v jaké generaci se právě trénink nachází. Část metody pro trénování je zobrazen jako kód 3.

```
def color_wheel_pick(color_wheel, picked=-1):
   i=picked+1
   pick = random.uniform(0, 1)
   x=0
   while True:
      x=x+color_wheel[i%len(color_wheel)]
      if pick<=x:
         return i % len(color_wheel)
      i+=1
```

Zdrojový kód 2: Metoda color_wheel_pick

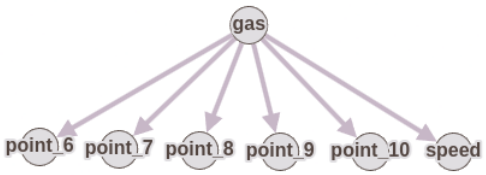
```
      speeds=np.divide(score,times)
      speed_q = np.divide(speeds, (np.repeat(np.max(speeds), len(
         speeds))))
      p_score = [i / top_trace for i in score]
      p_score_next = [(i / np.max(p_score)) * 0.7 for i in p_score]
      speed_q_norm = [i * 0.3 for i in speed_q]
      prob = np.add(p_score_next, speed_q_norm)
      new_population = []
      for i in range(len(prob)):
         if random.uniform(0,1) < prob[i]:
            new_population.append(copy.deepcopy(population[i]))
            new_population[-1].mutation(mutation_rate)
      color_wheel = [p/sum(prob) for p in prob]
      last_picked=-1
      for i in range(population_size - len(new_population)):
         last_picked = color_wheel_pick(color_wheel, last_picked)
         parent_one = population[last_picked]
         last_picked = color_wheel_pick(color_wheel, last_picked)
         parent_two = population[last_picked]
         new_population.append(parent_one.create_child(parent_two,
            mutation_rate))
      population = new_population
```

Zdrojový kód 3: Část metody pro trénování

4.3 Bayesovské sítě

Pro tuto práci byla vytvořena třída pro vytváření Bayesovských sítí a pro širší využití je zde možnost mít víc hodnot pro proměnnou, oproti běžným $\{True, False\}$. Síť funguje na základě názvu sloupců v tabulkách, které jí jsou poskytnuty jako základní pravděpodobnosti. Sloupec definující pravděpodobnost musí být nazván „prob“. Pokud jsou pravděpodobnosti závislé na jiné proměnné, pak název sloupce ponese stejné jméno jako ona proměnná a v jeho řádcích budou hodnoty, které mají na pravděpodobnost vliv. Celý výpočet pracuje na již popsaném algoritmu Belief. Je nutné brát zřetel na limity, které tato síť a algoritmus Belief obecně má. Pokud jsou alespoň dva nezávislé uzly a mají alespoň 2 společné potomky, pak dochází k cyklení požadavků o zprávy. Pro zabránění cyklení existuje LoopyBelief [8], který ale nemusí mít nejpřesnější výsledky. Řízení formule se bez sítě s cykly obejde. Výpočet pravděpodobnosti pozorování hodnot proměnné a výpočet prior pravděpodobností je v kódu 4.

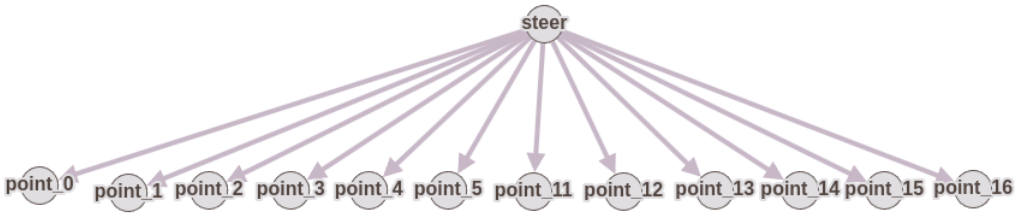
Přestože bylo možné dát vše do jedné sítě a ta by interně byla rozdělena na 2, nebyl v tom žádný benefit. Formuli tedy ovládají 2 sítě. Graf sítě starající se o přidání plynu je na obrázku 5 a graf sítě je na obrázku 6. Stisknutí plynu je pouze $\{0; 1\}$, u zatočení je to doleva = -1 a doprava = 1. Celé rozhodování je zapouzdřeno třídou BayesianCar, která při zavolání konstruktoru již očekává Bayesovské sítě. Tvorba sítě a především tvorba pravděpodobností pro její správné fungování je práce pro experta. Součástí práce tedy bylo i navrhnutí funkční sítě a vytvoření správných pravděpodobností na základě kterých by byl umělý hráč schopen dojet do cíle. Po mnoha neúspěšných pokusech o sběr dat a jejich následné upravení, byla vytvořena metoda get_preprocessed_car, která vrací již nachystané BayesianCar připraveno k použití. Použité pravděpodobnosti jsou založené na testech, pozorování a analýze dat. Pro zatáčení se využívají informace o bodech směřujících ke stranám a pro ovládání plynu se využívají body ve středu zároveň s informací o rychlosti.



Obrázek 5: Graf Bayesovské sítě pro přidání plynu

Je nutné zmínit v jakém formátu počítačová vize předává informace o vzdálenostech. Vize předává pole, kdy indexy jsou pořadí velikostí. Avšak vzhledem ke kódovému řešení došlo k zrcadlení a tak index 0 není dle očekávání nejlevější vzdálenost od bodu, ale je nejpravější.

Proto tedy pokud je velikost nejkrajnějších bodů 0 a 1 menší než 1.0, pak mají vysokou pravděpodobnost zatočení doleva, jak je viditelné v tabulce 1.



Obrázek 6: Graf Bayesovské sítě pro zatáčení

Pravděpodobnosti pro krajní body 2 až 5 v tabulce 2 více připomínají křivku oproti pravděpodobnostem pro body 0 a 1, které připomínají step funkci.

BayesianCar se stará o předání vstupů a ošetřuje výstup, aby mohla formule jet i rovně, namísto mikro-balancování. Vstupem jsou všechny body vize a rychlost, výstupem je stiknutí plynu $\{0; 1\}$ a zatočení $\{-1; 0; 1\}$.

	point_0-1	steer	prob
0	0.0	-1	0.95
1	0.1	-1	0.95
2	0.2	-1	0.95
3	0.3	-1	0.95
4	0.4	-1	0.95
5	0.5	-1	0.95
6	0.6	-1	0.95
7	0.7	-1	0.95
8	0.8	-1	0.95
9	0.9	-1	0.95
10	1.0	-1	0.50
11	1.1	-1	0.50
12	1.2	-1	0.50
13	1.3	-1	0.50
14	1.4	-1	0.50
15	0.0	1	0.05
16	0.1	1	0.05
17	0.2	1	0.05
18	0.3	1	0.05
19	0.4	1	0.05
20	0.5	1	0.05
21	0.6	1	0.05
22	0.7	1	0.05
23	0.8	1	0.05
24	0.9	1	0.05
25	1.0	1	0.50
26	1.1	1	0.50
27	1.2	1	0.50
28	1.3	1	0.50
29	1.4	1	0.50

Tabulka 1: Pravděpodobnost pozorování nejkrajnějších vzdáleností, pokud se zatáčí

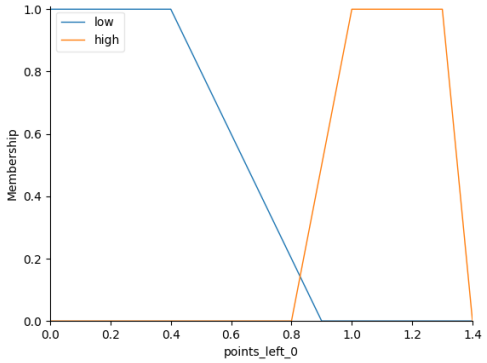
	point_2-5	steer	prob
0	0.0	-1	1.00
1	0.1	-1	0.95
2	0.2	-1	0.90
3	0.3	-1	0.85
4	0.4	-1	0.80
5	0.5	-1	0.75
6	0.6	-1	0.70
7	0.7	-1	0.65
8	0.8	-1	0.60
9	0.9	-1	0.53
10	1.0	-1	0.50
11	1.1	-1	0.50
12	1.2	-1	0.47
13	1.3	-1	0.20
14	1.4	-1	0.15
15	0.0	1	0.00
16	0.1	1	0.05
17	0.2	1	0.10
18	0.3	1	0.15
19	0.4	1	0.20
20	0.5	1	0.25
21	0.6	1	0.30
22	0.7	1	0.35
23	0.8	1	0.40
24	0.9	1	0.47
25	1.0	1	0.50
26	1.1	1	0.50
27	1.2	1	0.53
28	1.3	1	0.80
29	1.4	1	0.85

Tabulka 2: Pravděpodobnosti pozorování krajních vzdáleností, pokud se zatáčí

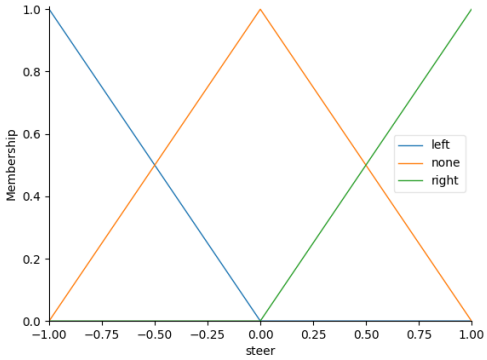
4.4 Fuzzy systém

Pro formuli založené na fuzzy systému byl využit balíček skfuzzy [3], který má již pro fuzzy systém předpřipravé metody. Jejich dokumentace bohužel není úplná a tak nějaké funkce je nutné zjišťovat z kódu. Pořadí programování se shoduje s osnovou práce a pravděpodobně Bayesovské sítě byly dobrým nástrojem pro nauku, jak expertního přístupu, tak zjištění chování formule s požadovanými reakcemi. Přesný návod pro tvorbu pravidel pro fuzzy systémy není, neboť problémy, které systémy řeší, jsou velmi specifické. Pro případ této práce, kdy je třeba kontrolovat

formuli dvěma nezávislými proměnnými. Nabízelo se obdobné řešení a tedy pro každou funkci mít oddělený řídící systém. Jeden řídící systém se stará o přidávání plynu a druhý o zatáčení, stejně jako u Bayesovských sítí. Bylo vytvořeno rozložení množin a definovány funkce příslušnosti. Proměnné se kterými se pracuje pro výpočet plynu jsou points_mid_i pro $i \in \{0, 1, 2, 3, 4, left, right\}$, speed a gas. Proměnné pro výpočet zatáčení jsou: points_left_i pro $i \in \{0, 1, 2, 3, 4, 5\}$, points_right_i pro $i \in \{0, 1, 2, 3, 4, 5\}$ a steer. Všechny proměnné point mají stejnou definici jako je znázorněno na obrázku 7. Proměnné gas a steer jsou výstupní. Definici proměnné steer je na obrázku 8.



Obrázek 7: rozdělení množiny point



Obrázek 8: rozdělení množiny steer

Pravidla jsou napsána na základě pozorování a chování formulí. Celkem se fuzzy formule řídí podle 12 pravidel, kdy 4 jsou pro přidávání plynu a 8 pro zatáčení. Obecně, pokud jsou velikosti vpravo vysoké, pak zatáčí vlevo a naopak. Pokud jsou všechny velikosti velké, nezatáčí. Pokud má před sebou malé vzdálenosti, nepřidává plyn a naopak. Přestože pravidla mají možnost se zlepšit, jedná se o plně funkční celek pro potřeby této práce. Pravidlová báze pro systém obsluhující přidávání rychlosti v celém znění je:

1. IF points_mid_0 is 'low' | points_mid_1 is 'low' | points_mid_2 is 'low' | points_mid_3 is 'low' | points_mid_4 is 'low' THEN gas is 'release'
2. IF points_mid_0 is 'high' & points_mid_1 is 'high' & points_mid_2 is 'high' & points_mid_3 is 'high' & points_mid_4 is 'high' & points_mid_left is 'high' & points_mid_right is 'high' THEN gas is 'press'
3. IF speed is 'low' THEN gas is 'press'
4. IF speed is 'high' | points_mid_left is 'low' | points_mid_right is 'low' THEN gas is 'release'

Pravidlová báze pro systém obsluhující zatáčení v celém znění je:

1. IF points_left_0 is 'low' | points_left_1 is 'low' THEN steer is 'right'

2. IF points_right_0 is 'low' | points_right_1 is 'low' THEN steer is 'left'

3. IF points_left_2 is 'low' & points_left_3 is 'low' & points_left_4 is 'low' & points_left_5 is 'low' THEN steer is 'right'

4. IF points_right_2 is 'low' & points_right_3 is 'low' & points_right_4 is 'low' & points_right_5 is 'low' THEN steer is 'left'

5. IF (points_right_2 is 'high' | points_right_3 is 'high' | points_right_4 is 'high' | points_right_5 is 'high') & (points_left_2 is 'high' | points_left_3 is 'high' | points_left_4 is 'high' | points_left_5 is 'high') THEN steer is 'none'

6. IF points_right_5 is 'high' & points_left_5 is 'high' THEN steer is 'none')

7. IF (points_right_2 is 'high' | points_right_3 is 'high' | points_right_4 is 'high' | points_right_5 is 'high') & points_left_2 is 'low' & points_left_3 is 'low' & points_left_4 is 'low' & points_left_5 is 'low' THEN steer is 'none'

8. IF points_right_2 is 'low' & points_right_3 is 'low' &points_right_4 is 'low' & points_right_5 is 'low' & (points_left_2 is 'high' | points_left_3 is 'high' | points_left_4 is 'high' | points_left_5 is 'high') THEN steer is 'none'

5 Trénování a experimentování

5.1 Trénování AI

Trénování bylo pouze pro umělou neuronovou síť, ostatní AI požadují pro uvedení do provozu data, nejlépe od experta na danou problematiku, kterou mají řešit. Umělé neuronové sítě se pomocí genetického algoritmu dokázaly naučit dojet do cíle během prvních 10 generací. Bohužel přes mnoho různých počátečních populací se vždy rozhodly, že optimálním řešením je jet podél zdi, což nemusí být obecně velmi optimální strategií. Velikost neuronové sítě, která se nejlépe osvědčila, aby bylo možné ji v roumném čase naučit a zároveň dojela do cíle pro 19 vstupů (rychlost, aktuální zatočení a 17 pro vizi) je 13 neuronů v první vrstvě, 9 neuronů v druhé a 6 ve třetí, která je i poslední. Každý neuron poslední vrstvy představuje jednu z následujících akcí, které zastupují stiknutí šipek na klávesnici: nejet/nezatáčet, nejet/zatočit vlevo, nejet/zatočit vpravo, jet/nezatáčet, jet/zatočt vlevo, jet/zatočit vpravo. Bylo provedeno i několik pokusů, kdy poslední vrstva měla jen 2 neurony, jeden ovládal zatáčení a jeden plyn. Trénování bylo velmi obtížné a pokud síť jela alespoň trochu, tak byla mnohem více závislá na zdi a bez zatáčení jezdit nezvládla. Mezi nepovedené pokusy natrénování patří i povolení jízdy vzad.

Tvorba dat pro Bayesovské sítě stála mnoho času. Byly napsány i metody pro sběr dat, které byly následně zpracovávány. Přímé vytažení pravděpodobností z dat výpočtem dávalo formuli velmi smíšené signály. Pravděpodobně se jednalo o příliš malý vzorek. Další možností bylo využít dat pro analýzu a na jejím základě přizpůsobit pravděpodobnosti přímo z nich získané. Ani tento způsob nebyl nejlepší a stále vznikaly v jízdě velké chyby. Poslední a použitý přístup bylo předpovídat chování. Na základě dat, která přichází bylo odhadováno jakým směrem by měla formule jet a jak. Na základě sepsání těchto odhadů byly vytvořeny pravděpodobnosti, které nyní řídí.

Data, neboli pravidla a funkce příslušnosti pro fuzzy systém, byla vytvořena na základě znalosti získané od Bayesovských sítí. Nejdelší čas byl stráven na vymyšlení pravidel, aby odpovídala představě o chování formule a zároveň aby bylo možné je vůbec aplikovat. Několikrát se stalo, že i vlivem funkcí příslušnosti, se výsledek ocitl v nedefinovaném prostoru použité knihovny. Nakonec se povedlo najít pravidla i funkce, co spolu příjemně fungují.

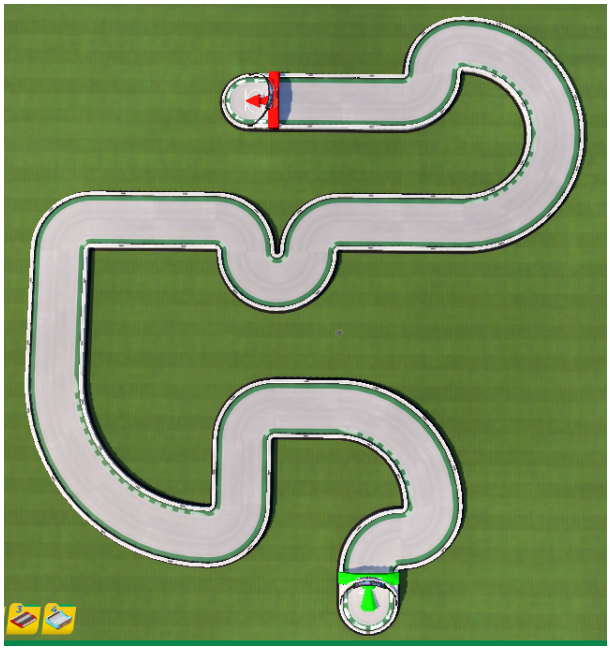
5.2 Testování

Umělá neuronová síť se snaží jezdit co nejrychleji a u pravé zdi. Narazit pod plným plynem do zdi tratě je strategie, která sice nikam nevede, ale některé ji mají v oblibě. Přestože byly některé schopny projet cílem a jsou tedy vedeny jako úspěšné, ne vždy své projetí zreplikují 1:1. Může se jednat o minimální prodlevy, které se ve výsledku nasčítají nebo lehce odlišný čas zpracování při trénování na jiném stroji.

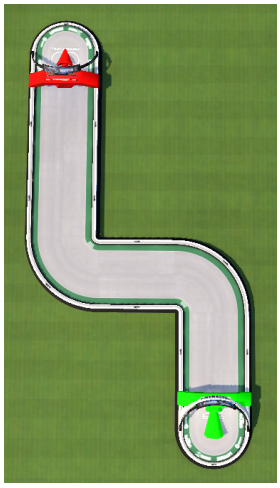
Formule založená na Bayesových sítích je velmi stálá svou rychlostí a velmi málo se k mantinelům vůbec přiblíží, což je chtěná vlastnost. Pravděpodobnosti ohledně přidávání plynu by nejspíše potřebovaly hlubší analýzu chování tratě a upravit schod na pozvolnější ubývání pravděpodobnosti. Na základní mapě Promomap jezdí ukázkově a velmi opatrně.

Formule založená na fuzzy řídícím systému už tak opatrná není, formule dosahuje v některých okamžicích tratě slušných rychlostí, ale díky tomu se občas dotkne stěny. Stále je schopna dostat se do cíle v relativně normální čas, jako závodní protivník použitelná není. Nebyla nalezena úprava, která by tento problém řešila, aby nebyla rychlost omezená příliš.

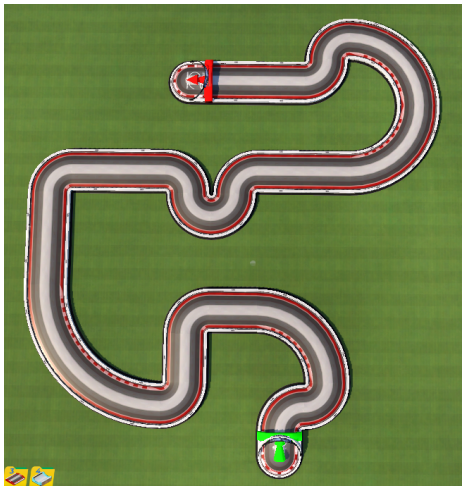
Bylo vytvořeno několik map pro účely testování. Promomap na obrázku 9 je základní trénovací mapa, která má všechny druhy zatáček ze základního editoru i rovinné úseky. Mapa r-l-map na obrázku 12 a l-r-map na obrázku 10 jsou mapy pro testování zatáčení. Mapa promomap-saus na obrázku 11 je obtížnější verzí trénovací mapy, obsahuje vypouklý asfalt a má na sobě černé prvky, což může být pro vizi matoucí.



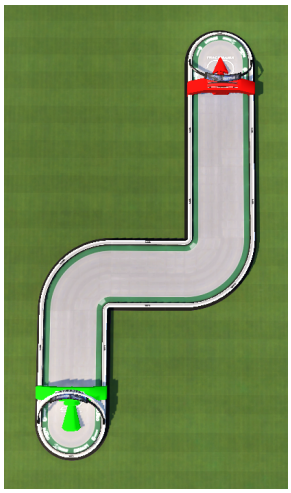
Obrázek 9: Mapa promomap



Obrázek 10: l-r-map



Obrázek 11: promomap-saus



Obrázek 12: r-l-map

Pro testování byly náhodně vybrány 4 umělé neuronové sítě z různých generací, které dojely do cíle a bral se jejich nejlepší výsledek. Data mohou být ovlivněna operačním systémem a hardwarem. Každý systém využívá pro kontrolu hry jinou knihovnu, takže mohou vznikat malé prodlevy, které se mohou odrazit na chování. Linuxová knihovna má v případech stisknutí prodlevy povinné. Slabší hardware by mohl zapříčinit pomalejší výpočet umělé inteligence. Je dobré zmínit, že zatímco se AI rozhoduje jaký bude její další krok, tak je neustále v běhu její poslední zvolená akce. Pokud se zpozdí výpočet, zpozdí se i změna akce, což může vyvolat rozdíl například v projetí zatáčky nebo zda narazí do zdi.

	NNC score	**NNC time**	**BC score**	**BC time**	**FC score**	**FC time**	**Author time**
promomap	62%	73s	96%	115s	89%	97s	24s
r_l_map	-	-	99%	25s	93%	19s	6s
l_r_map	41%	15s	100%	24s	89%	19s	6s
promomap_saus	-	-	44%	202s	85%	117s	30s

Tabulka 3: Data z testování

Názvy formulí podle ovládající umělé inteligence: NNC = Umělá neuronová síť, BC = Bayesovské sítě, FC = Fuzzy systém. Dle dat v tabulce 3 lze správně soudit, že stragie neuronových sítí není dobře přenositelná na další tratě. Formule ovládaná Bayesovými sítěmi je velmi přesná, co se držení ve středu tratě týče, trať promomap_saus má černé prvky, které může jejich vidění vidět jako okraj. Formule BC sklouzla ke kraji a již se od něj nedostala. Jedná se tedy spíš o problém vidění než BC. Formule řízená fuzzy sice nemá takové pokrytí jako BC, ale to vyrovná časem projetí trati.

Byla vytvořena i trať, která obahující převýšení a pokroucené okraje. Bohužel ji nebyla ani jedna umělá inteligence schopna dojet do konce. Podle sledování měla NNC stejný problém jako u jiných jí nedojetých tratí – zasekla se u zdi kolmo k ní. BC měla problém s vyjetím do kopce, protože její rychlost je omezená, na tomto případě je to velmi zřejmé. FC mělo problém s vlnitými okraji, spíš se jedná o vadu vidění než o vadu FC, ale je možné, že chyba je na obou stranách. Přesto se FC dostalo na trati nejdál. I za cenu ztráty přesnosti má FC dobrý čas dojetí.

```
1     def get_likelihood(
2           self):
3        if self.likelihood is not None:
4           return self.likelihood
5        tab = self.possibilities.copy()
6        tab['prob'] = 1
7        for child in self.children:
8           tab = pd.merge(tab, child.send_msg_to_parent(self), on=self
                .name, how='outer',
9                          suffixes=('', '_' + child.name))
10          tab['prob'] = tab['prob'] * tab['prob_' + child.name]
11          tab = tab.drop('prob_' + child.name, axis='columns')
12       return tab
13
14    def get_priors(self):
15       if self.priors is not None:
16          return self.priors
17       parents_priors = {}
18       for p in self.parents:
19          parents_priors[p.name] = p.send_msg_to_child(self)
20       ult_table = self.table.copy()
21       for pp in parents_priors:
22          ult_table = pd.merge(ult_table, parents_priors[pp], on=pp,
               how='outer', suffixes=('', '_' + pp))
23          ult_table['prob'] = ult_table['prob'] * ult_table['prob_' +
                pp]
24          ult_table = ult_table.drop('prob_' + pp, axis='columns')
25       ult_table = ult_table.groupby(by=self.name)['prob'].sum().
            reset_index(name='suma')
26       priors = self.possibilities.copy()
27       priors = pd.merge(priors, ult_table, on=self.name, how='left',
             suffixes=('', '_'))
28       priors['prob'] = priors['suma']
29       priors = priors.drop('suma', axis='columns')
30       return priors
```

Zdrojový kód 4: Metody ve třídě Node get_likelihood a get_priors

Závěr

Srovnání různých přístupů k umělé inteligenci na stejné problematice, kterým jsou autonomní hráči závodních her bylo zajímavé a práce by mohla být využita jako přehled funkcionality zkoumaných přístupů.

Překvapivá byla stabilita skrze všechny provedené testy formule řízené fuzzy systémem. Samotná konfigurace může být pro nezaškolené osoby složitá, ale díky využité knihovně je možné k těmto systémům snadno dostat. Bayesovské sítě s algoritmem Belief, které byly naprogramovány pro tuto práci jsou využitelné i mimo ni, zvláště díky možnosti většího množství hodnot pro proměnné. Umělé neuronové sítě v kombinaci s genetickými algoritmy byl obstojný pokus, jestli je možné naučit síť bez učitele. Takto malá neuronová síť naučená bez učitele, ale není lehce přenositelná na další mapy.

Formule založené na fuzzy či na Bayesovských sítích by v rukou odborníků mohly dosahovat vynikajících výsledků. Z práce vyplývá, že pro trénování umělých neuronových sítí nejsou genetické algoritmy nejvhodnějším způsobem. Tato práce může být brána jako nástin, jak různé umělé inteligence mohou řešit stejný problém, ale jiným způsobem a jinak dobře. Největší problém byl zaznamenán u počítačového vidění. Bylo by na místě se věnovat jeho rozvoji a přidání hloubky, ať už nějakým způsobem ze hry, pokročilejší analýzy obrazu nebo využitím další umělé inteligence pro rozpoznání trasy.

Conclusions

The comparison of different approaches to artificial intelligence on the same issue, which is the autonomous players of racing games, was interesting and the work could be used as an overview of the functionality of the investigated approaches.

The stability through all tests of the car formula controlled by the fuzzy system was surprising. The configuration itself can be complex for uninitiated people, but thanks to the library that was used, it is possible to get to these systems easily. Bayesian networks with the Belief algorithm, which were programmed for this work, can also be used outside of it, especially thanks to the possibility of a larger number of values for the variables. Artificial neural networks in combination with genetic algorithms was a passable attempt to see if it is possible to teach a network without a teacher. Such a small neural network learned without a teacher, but it is not easily transferable to other maps.

Car formulas based on fuzzy or Bayesian networks could achieve excellent results in the hands of experts. The work shows that genetic algorithms are not the most suitable method for training artificial neural networks. This work can be taken as an outline of how different artificial intelligences can solve the same problem, but in a different way and differently well. The biggest problem was seen with computer vision. It would be a good idea to continue development and add depth, either through game data, more advanced image analysis or the use of additional artificial intelligence for route recognition.

A Obsah elektronických dat

text/
: Adresář s textem práce ve formátu PDF, vytvořený s použitím závazného stylu KI PřF UP v Olomouci pro závěrečné práce, včetně všech (textových) příloh, a všechny soubory potřebné pro bezproblémové vytvoření PDF dokumentu textu (případně v ZIP archivu), tj. zdrojový text textu a příloh, vložené obrázky, apod.

README.txt
: Textový soubor obsahuje informace o požadavcích na hardware a software, postup pro zprovoznění všech potřebných programů, seznam použitých knihoven pro Python a návod k jejich instalaci, detailní postup k zacházení s prací.

maps/
: Adresář se všemi použitými mapami pro Trackmania, které byly využity.

skfuzzy/
: Adresář obsahuje upravený soubor pro knihovnu skfuzzy.

processedtraces/
: Adresář obsahující data o projetí lidským hráčem přiloženými mapami.

neuralnetworks/
: Adresář obsahující vybrané umělé neuronové sítě, které byly schopné dojet do cíle během trénování.

sourcecodes/
: Adresář obsahuje všechny zdrojové kódy v jazyce Python.

support/
: Adresář obsahuje základní obrázek pro výpočet báze a další soubory pro zprovoznění práce

Literatura

[1] GEELS, Melissa 'Miss'. OpenPlanet [online]. ş2016-2024. [cit. 2024-08-05]. Dostupné z: https://openplanet.dev/

[2] BOUTEILLER, Yann; GEZE Edouard. TMRL. GitHub [online]. 2014-05-12. [cit. 2024-08-05]. Dostupné z: https://github.com/trackmania-rl/tmrl

[3] THE SCIKIT-FUZZY TEAM. Scikit-fuzzy. GitHub [online]. (c) 2012. [cit. 2024-08-05]. Dostupné z: https://github.com/scikit-fuzzy/scikit-fuzzy

[4] GOODFELLOW, Ian; BENGIO Yoshua; COURVILLE Aaron. Deep Learning [online]. MIT Press, c2016. [cit. 2024-08-05]. Dostupné z: http://www.deeplearningbook.org

[5] KOLLER, Daphne; FRIEDMAN Nir. Probabilistic graphical models: principles and techniques. Adaptive computation and machine learning (MIT Press). Cambridge: MIT Press, c2009. ISBN 978-0-262-01319-2. [cit. 2024-08-05].

[6] KRUSE, Rudolf; BORGELT Christian; BRAUNE Christian; MOSTAGHIM Sanaz; STEINBRECHER Matthias; KLAWONN Frank; MOEWES Christian. Computational Intelligence: A Methodological Introduction. Springer London, c2016. ISBN 9781447172963. [cit. 2024-08-05].

[7] KUNC, Vladimír; KLÉMA Jiří. Three Decades of Activations: A Comprehensive Survey of 400 Activation Functions for Neural Networks. arXiv [online]. c2024. [cit. 2024-08-05]. Dostupné z: https://arxiv.org/abs/2402.09092

[8] MÉZARD Marc; MONTANARI Andrea. Information, Physics, and Computation. Oxford University Press, 2009. ISBN 978-0198570837. [cit. 2024-08-05].

[9] WENIG Phillip. Belief Propagation in Bayesian Networks. Towards Data Science, 2019-05-26. [cit. 2024-08-05]. Dostupné z: https://towardsdatascience.com/belief-propagation-in-bayesian-networks-29f51fdc839c