Inteligentní křižovatky

Matyáš Lamprecht
Univerzita Karlova, Matematicko-fyzikální fakulta, Praha, Česká republika

Abstract

Problém křižovatky spočívá v hledání co nejefektivnějšího průjezdu křižovatkou pro množinu nezávislých agentů, jejichž cesty se navzájem kříží. V této práci se zabýváme různými přístupy, které byly pro problém křižovatky navrženy. Ukazujeme provedené experimenty, zhodnocujeme dobré a špatné stránky daných přístupů a snažíme se navrhnout možná vylepšení.

Úvod

Křižovatky jsou jedním z nejvíce problematických míst na silnici. Dochází zde k velkému množství dopravních nehod – až ke 40% všech dopravních nehod a až k 50% všech vážných dopravních nehod. Problém je v tom, že zde dochází ke křížení cest jednotlivých vozidel a tedy všechna vozidla nemohou jet najednou. Navíc když doprava okolo křižovatky zhoustne, tak každé vozidlo musí čekat delší a delší dobu na průjezd křižovatkou.

Aby byly průjezdy křižovatkou efektivní a zároveň bezpečné, bylo navrženo větší množství typů křižovatek. Ty jsou navržené podle toho, jak je daná křižovatka vytížená. V málo vytížených křižovatkách stačí respektovat dopravní značky a předpisy. V rušnějších křižovatkách se používá světelná signalizace. Ta ovšem nemusí být vždy efektivní. Pokud například přijedeme na prázdnou křižovatku, ale máme červenou, tak nesmíme vjet do křižovatky, i když tam nikdo není a tedy nemusíme nikomu dávat přednost. V tomto případě je křižovatka velice neefektivní. Pokud bychom ovšem mohli používat autonomní vozidla a nějaké inteligentní řízení křižovatky, tak bychom mohli průjezdy křižovatkou zefektivnit a takovéto situace by nenastávaly. Navíc bychom mohli zamezit nehodám, které se dnes v křižovatkách stávají.

Věty, které jsou takto napsány kurzívou, vyjadřují názor autora.

Dnes již některá vozidla umí sama jezdit na silnicích s pruhy - např. na dálnici. Umí předjíždět, brzdit, drží si rychlost podle značek. Kde ale mají tato vozidla problém jsou právě křižovatky, protože zde žádné pruhy nejsou.

Online MAPF

Na problém křižovatky se můžeme dívat jako na online MAPF (multi-agent path finding) problém. Jedná se o generalizaci offline MAPF. Začínáme s problémem offline MAPF, agenti vykonávají rozvržený plán a noví agenti se můžou vyskytnout na mapě. i-tý nový agent je trojice $< \ t_i, s_i, g_i \ >$, kde t_i je čas, kdy se agent objevil, s_i je počáteční pozice agenta a g_i je cílová pozice agenta (Švancara et al. 2019). Řešením online MAPF je sekvence plánů $\Pi =< \pi^0, \pi^1, ..., \pi^m >$, kde m je počet nových agentů, kteří se postupně objevili; π^0 je plán pouze pro offline vstup; pro všechna $j > 0 : \pi^j$ je plán pro offline vstup, pro agenta j a pro všechny agenty, kteří se objevili dříve než agent j. Online MAPF má více variant:

- Nový agent se objeví:
  - v počáteční pozici – problém: může způsobit nepředvídatelné kolize – na problém křižovatky se nehodí
  - nový agent musí udělat pohyb, aby se dostal do počáteční pozice
- Agent dosáhne cíle:
  - zůstane v něm
  - zmizí – budeme používat pro problém křižovatky

Dále si musíme zadefinovat objektivní funkce (Švancara et al. 2019). Převezmeme objektivní funkci z offline algoritmu: $f_1(\Pi) = \sum_{i \in A} |Ex[\Pi]_i|$, kde $|Ex[\Pi]_i|$ je počet kroků i-tého agenta, než dosáhne cíle. Této objektivní funkci se říká sum-of-costs. Ovšem Makespan (délka celého plánu – poslední agent přijde do cíle) se nedá použít, protože se jedná o online problém a tedy poslední agent není. Pro online MAPF se tedy používají dvě nové objektivní funkce:

- $f_2(\Pi) = \sum_{t=1}^{\infty} NotAtGoal(t)$, což sčítá počty agentů, kteří nejsou v časovém kroku t v cíli.
- $f_3(\Pi) = \sum_{i \in A} |Ex[\Pi]_i| - o_i$, kde o_i je délka optimální (nejkratší) cesty pro i-tého agenta, když by se ostatní agenti nevyskytovali.

Nyní k online MAPF algoritmům (Švancara et al. 2019). Ty se liší podle toho, co se stane, když se objeví nový agent. *Replan Single* algoritmus hledá optimální cestu

pro všechny nové agenty postupně tak, že se vyhýbá již naplánovaným agentům. *Replan Single Grouped* algoritmus hledá optimální řešení pro všechny nové agenty najednou. Dalším algoritmem je *Replan All*, který vždy, když se objeví nový agent, naplánuje všechny agenty znova z jejich aktuálních pozic. Tím dostaneme nejlepší možné řešení tzv. snapshot optimální řešení, což je typ plánu, kde všichni agenti mají optimální trasu do cíle (suma přes ceny), kdy předpokládáme, že žádní noví agenti se v budoucnu nevyskytnou. Nicméně algoritmus *Replan All* není vždy úplně žádoucí, protože plýtvá časem. Navíc změna plánu agenta, který se již hýbe, také nemusí být žádoucí. Posledním přístupem, který si zmíníme je *Online Independent Detection*. Ten je založen na algoritmu *Independence Detection*, který povolí novým agentům cesty tak, že se ignorují ostatní agenti. Když při plánování vznikne konflikt s již naplánovaným agentem, tak vezme skupinu konfliktních agentů a plánuje všechny společně. Pokud by opět došlo ke konfliktům, tak vezme opět všechny konfliktní agenty a plánuje je dohromady. Takto iteruje dokud nejsou žádné konflikty. *ID* ovšem nemusí vrátit snapshot optimální řešení. *OID* vychází z *ID* a je upraveno tak, aby snapshot optimální řešení našlo.

Křižovatka jako multi-agentní systém s centrálním řízením

Dresner a Stone (Dresner a Stone 2008) navrhli rezervační systém pro průjezd autonomních vozidel, kde se snaží splnit následující body:

1. autonomní vozidla jsou nezávislí agenti
2. agenti si předávají pouze minimální množství zpráv
3. sensor model realism – každý agent má přístup pouze k senzorům, které jsou přístupné dnešními technologiemi. Mechanismus by neměl spoléhat na nějaké fiktivní technologie senzorů, které ještě nebyly zrealizovány.
4. agenti komunikují pomocí standardizovaného komunikačního protokolu
5. vyvarování se deadlockům – každý agent nakonec křižovatkou projede
6. celý systém by měl být rozšiřitelný – měl by být postupně použitelný pro všechny křižovatky a měl by být použitelný i v případě, že některá vozidla nebudou zcela autonomní
7. bezpečnost – pokud vozidla budou dodržovat protokol, tak by nikdy nemělo dojít ke strážce v křižovatce
8. efektivita – vozidla by měla projet křižovatkou v nejmenším možném čase

Dnešní křižovatky splňují vše kromě efektivity. I když se v nich stává velké množství nehod, tak se většinou nejedná o chybu křižovatky ale řidičů.

Účastníci komunikace v křižovatce jsou dvou typů:

- driver agent (DA) – zodpovídá za řízení vozidla a komunikaci s okolím

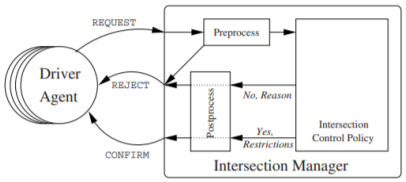


Figure 1: Komunikace DA s IM.

- intersection manager (IM) – nachází se v místě křižovatky a jeho úkolem je přidělovat jednotlivým DA povolení ke vjezdu

DA chce projet křižovatkou, tedy pošle při příjezdu do křižovatky příslušnému IM žádost s časem příjezdu, odhadovanou rychlostí a fyzickými parametry vozidla. IM následně na základě své strategie (ICP – intersection control policy) rozhodne, zda požadavku vyhoví a povolí DA vjet do křižovatky, nebo ho zamítne. Jestliže DA neobdrží kladné potvrzení požadavku, nesmí v žádném případě do křižovatky vjet. *To je celkem logické, jelikož např. kdyby došlo k problému v komunikaci mezi DA a IM a DA by mohlo vjet do křižovatky bez potvrzení, tak by mohlo dojít ke kolizím.* Celý proces je vyobrazen na Obrázku 1.

Strategie FCFS - first come first served

Vozidla jsou obsluhována podle časů příjezdů. FCFS umožňuje vozidlu si dopředu rezervovat čas a místo, které potřebuje pro průjezd křižovatkou. Plánování dopředu umožňuje vozidlům přijíždějícím ze všech stran projet křižovatkou současně s minimálním zpožděním. Strategie funguje následovně.

Křižovatka je rozdělena na mřížku s $n * n$ políčky, kde n se nazývá granularita křižovatky. Od přijíždějícího agenta dostane parametry vozidla. Strategie rozběhne interní simulaci trajektorie přes křižovatku za pomocí těchto parametrů. V každém kroku interní simulace strategie určí jaká políčka budou obsazena daným vozidlem. Když v jakémkoliv čase simulace požaduje políčko, které již je rezervováno jiným vozidlem, tak strategie žádost zamítne. Jinak strategie žádost přijme a rezervuje daná políčka pro časy, kdy jsou požadována.

Strategie FCFS pracuje dobře, ale v prvních implementacích bylo možné jet křižovatkou pouze rovně. Poté, co se dovolilo zatáčet bylo zřejmé, že by si vozidla sama neměla určovat pruh, ze kterého budou zatáčet. Místo toho IM, který má mnohem více informací o situaci, dělá toto rozhodnutí. DA určí, jakým směrem chce křižovatku opustit, a ICP poté rozhodne, jakým pruhem má jet. V experimentech se použilo celkem zjevné pravidlo, že ta vozidla, která jedou doleva, používají nejlevější pruh, ta, která jedou doprava, používají nejpravější pruh a ta, která jedou rovně, si vyberou pruh, ve kterém přijedou.

Další věcí, která se musí probrat je zrychlování v křižovatce. Pokud bychom povolili zrychlování v křižovatce, tak existuje nekonečně mnoho trajektorií, jakými může dané vozidlo projet křižovatkou. Tedy nějaké hodnoty zrychlení by mohly způsobit, že dané vozidlo

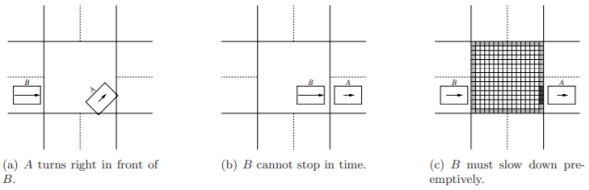
může vjet do křižovatky, zatímco rozdílné hodnoty můžou způsobit, že povolení pro vjezd bude zamítnuto. Proto o zrychlování vozidel rozhoduje ICP. První možností je, že se všechna vozidla budou pohybovat stejnou rychlostí. Tento přístup má ovšem určité nedostatky, největší z nich je, že může dojít k deadlocku, kdy vozidla jedou čím dále pomaleji a pomaleji. FCFS strategie tedy používá jinou strategii, kde se nejdříve snaží nasimulovat průjezd křižovatkou, kde vozidlo začne zrychlovat co nejrychleji – na maximální povolenou rychlost, jakmile přijede do křižovatky. Pokud tento přístup neuspěje, tak nasimuluje průjezd křižovatkou, kdy vozidlo udržuje stejnou rychlost. Pokud ani toto nejde, tak je žádost zamítnuta.

Celý algoritmus FCFS mi připadá celkem logický – vozidla jsou obsluhována podle příjezdu. Co bych upravil je podle čeho IM určuje dané zrychlení vozidla – to se může stářím vozidla, podle mě, měnit, navíc třeba i podle počasí – na mokré silnici se zrychluje hůře. Tím nechci říci, že by to byl úplně špatný nápad, zrychlování v křižovatce použít, ale chtělo by to drobnou úpravu. Nezrychloval bych, co nejrychleji to jde, ale vzal bych si parametry vozidla, určil podle nich nejvyšší možné zrychlení a zrychloval bych třeba ze 70% možného zrychlení, abych odstranil problém, že vozidlo nebude schopné zrychlit maximálním způsobem.

FCFS strategie také řeší příliš pomalá vozidla. Jestliže IM obdrží žádost, která byla odeslaná při velmi nízké rychlosti (menší než nějaká konkrétní minimální mez), provede simulaci pouze ve verzi se zrychlením. Tím zabrání deadlockům, které v prvním řešení mohly vznikat (viz *Příklad*) a zároveň zajistí, že vozidla nestráví v křižovatce příliš mnoho času.

Příklad (Deadlock při pomalé jízdě) (Škopková 2019). Pokud příliš pomalé vozidlo jede do křižovatky a není možné ho předjet, pak vozidlo za ním musí také zpomalit, zrušit svou rezervaci získanou při vysoké rychlosti a vytvořit novou rezervaci při nízké rychlosti. Další vozidlo přijíždějící ke křižovatce musí opět zpomalit a zažádat si o rezervaci při nízké rychlosti atd. Pokud by tato situace nastala v hustém provozu, tak by žádné nové vozidlo nemělo šanci projet křižovatku rychleji než původní pomalé vozidlo. Nezřídne-li provoz na nulu, křižovatka bude přeplněna.

Další problém, se kterým se musela FCFS strategie vypořádat, je, že když je žádost vozidla zamítnuta, tak DA vozidla ihned může poslat novou žádost. Tato nová žádost ale velice pravděpodobně nebude moc odlišná od minulé žádosti, a tak s největší pravděpodobností bude také zamítnuta. Pokud ovšem vozidlo zpomalí dostatečně anebo nějaká jiná žádost bude zrušena, tak vozidlo může dostat povolení pro průjezd křižovatkou. Z pohledu IM je ovšem každá zamítnutá žádost před úspěšnou žádostí ztráta času/úsilí. Jelikož FCFS strategie běží dvě simulace (zrychlení, stejná rychlost) pro jednu žádost, tak tyto žádosti můžou být poměrně výpočetně náročné, zvláště pokud se jedná o křižovatku s vysokou granularitou. Tedy, abychom nepřetížili IM mnoha žádostmi, strategie zahrnula timeouty. To znamená, že pokud je žádost daného vozidla zamítnuta, tak další jeho žádost nebudeme brát v úvahu do té doby, než vyprší timeout. Ve své implementaci se Dresner a Stone opřeli o výpočetní vzorec: $t + min(\frac{1}{2}, \frac{t_a - t}{2})$, kde t je čas nyní a t_a je čas příjezdu v žádosti. Tento proces velice zredukuje počet žádostí a také to dává možnost více žádostí vozidlům, které vstoupí do křižovatky dříve.



(a) A turns right in front of B. (b) B cannot stop in time. (c) B must slow down preemptively.

Figure 2: Možnost kolize za křižovatkou.

Figure 3: Vlevo časový buffer pro malou rychlost, vpravo časový buffer pro velkou rychlost.

Tento přístup se mi zdá velice dobrý, jelikož preferuje žádosti vozidel, která jsou blíže křižovatce.

Dalším problémem je možnost kolize těsně za křižovatkou, jak je ukázáno na Obrázku 2 – v a a b. Zde je problém v tom, že odbočující vozidlo A má malou rychlost, protože odbočuje, zatímco vozidlo B jedoucí rovně má velkou rychlost a tedy může dojít ke srážce. Na Obrázku 2 v sekci c si křižovatku rozdělíme na interní políčka a políčka na hraně. Na políčkách na hraně zvětšíme časový buffer – "roztáhneme" dané vozidlo podle aktuální rychlosti, jak můžeme vidět na Obrázku 3.

Tento přístup se mi zdá zcela v pořádku a nevidím v něm žádný problém.

S rychlostí, kterou vozidlo projede křižovatkou, úzce souvisí rychlost, kterou se vozidlo ke křižovatce dostaví. Jelikož DA informují IM o svém příjezdu dříve, než ke křižovatce dorazí, musí na základě své aktuální rychlosti odhadnout čas příjezdu do křižovatky. Pro výpočet tohoto času navrhli Dresner a Stone dva přístupy:

- optimistický agent: předpokládá, že ve chvíli odeslání žádosti vozidlo okamžitě zrychlí na maximální povolenou rychlost, a zahrne tento fakt do výpočtu očekávaného času příjezdu.
- pesimistický agent: neočekává, že zrychlí, předpokládá, že do křižovatky přijede aktuální rychlostí.

Každý agent střídá tyto dva přístupy podle své aktuální situace. Příkladem agenta, který začne být optimistický, je agent, který se právě dostal do pruhu, ve kterém přestal být blokován pomalu jedoucím vozidlem. Příkladem agenta, který začal být pesimistický, je agent, který byl nucen zrušit svou rezervaci, protože ji nemohl stihnout.

Lidští řidiči

Až budou v budoucnu fungovat inteligentní křižovatky, tak je možné, že stále budou existovat lidé, kteří milují řízení a budou chtít řídit sami. Navíc doba přechodu mezi dnešním

způsobem křižovatek a chytrými křižovatkami nebude ze dne na den, tedy je dobré v modelu počítat i s lidskými řidiči. Dále je potřeba počítat s cyklisty a chodci, kteří také chtějí projet/projít křižovatkou. V tzv. FCFS-Light modelu Dresner a Stone navrhují použít světelnou signalizaci. Funkci IM rozšiřují o schopnost ovládat světla v křižovatce, pomocí nichž má IM vytvářet rezervace pro lidmi řízená vozidla. Cyklisti a chodci jsou ošetření tím, že na křižovatce (jak již funguje dnes) bude tlačítko, které zmáčknou a tím oznámí svoji přítomnost.

Byly navrženy dva přístupy. První je založen na postupném střídání zelené pro různé směry. Tedy nejdříve mají zelenou vozidla z 1. směru, poté z 2. směru, následně ze 3. směru a nakonec ze 4. směru. A takto se zelená střídá pořád dokola. 2. přístup je také založen na postupném střídání zelené pro různé směry, ale je ještě rozvrstven. Tedy nejprve dá zelenou v 1. směru pro odbočování doleva, poté dá zelenou v 1. směru pro jízdu rovně a následně dá zelenou v 1. směru pro odbočování doprava. Stejným způsobem pokračuje pro ostatní směry. Pravidla pro průjezd takovouto křižovatkou platí následující:

1. Když svítí zelená, tak všechna vozidla v daném směru můžou jet.
2. Dát rezervaci DA kdykoliv je to možné. Autonomní vozidla tedy můžou jet i na červenou.

Tedy DA pošle žádost IM a IM se rozhoduje následovně:

- Když vozidlo přijede v pruhu, ve kterém bude zelená, tak vždy dostane povolení pro vjezd.
- Když přijede v pruhu, kde bude oranžová, tak povolení pro vjezd nikdy nedostane - z bezpečnostních důvodů.
- Pokud přijde v pruhu, kde je červená, tak spustí FCFS simulaci. Když v simulaci nejsou daná políčka rezervovaná jiným vozidlem nebo zeleným či oranžovým světlem, tak vjezd povolí. Jinak zamítne.

Tady mě napadá, že zejména 1. implementace není moc efektivní. Je to z toho důvodu, že kvůli zelené barvě z určitého směru vyblokuji průjezd ostatních směrů. Pro 1. přístup a granularitu křižovatky 1, by tedy daný přístup neměl žádné výhody. Pro větší granularitu např. 3 by alespoň autonomní vozidla mohla vždy odbočovat vpravo. V tom mi přijde 2. implementace lepší, že nedochází k až tak velkému vyblokování. Nicméně by asi záleželo na konkrétním použití 1. či 2. přístupu podle toho, jaký bude poměr autonomních a neautonomních vozidel. Protože 1. přístup se mi zdá lepší, když se bude vyskytovat stále mnoho neautonomních vozidel, zatímco 2. se mi zdá lepší, když bude více vozidel autonomních. Dále v čem je výhoda, že když by v křižovatce nikdo nebyl a IM vidí přijíždějící vozidlo, tak mu může dát zelenou. Co se může také stát je, že určitý DA stojí za vozidlem, které je řízeno řidičem a tedy daného DA blokuje. Co mě napadlo jako vylepšení je, že by na světelných křižovatkách byla defaultně červená. Tedy autonomní vozidla by mohla projíždět podle FCFS strategie a pouze pokud by se objevilo nějaké neautonomní vozidlo, tak by došlo v určitém pruhu k přepnutí na zelenou.

Záchranná vozidla

Když jedou záchranná vozidla – ambulance, hasiči, policie, tak v dnešní době ostatní vozidla zastaví na kraji silnice a pustí daná záchranná vozidla před sebe a poté pokračují v jízdě. Jak to naimplementovat? IM musí nejprve zaznamenat jejich přítomnost – to je poměrně snadné – přidáme do žádosti nové políčko, které bude označovat, jestli se jedná o záchranné vozidlo nebo ne. Poté ICP dává přednost záchranným vozidlům. FCFS-EMERG policy ví, ve kterých pruzích jedou přijíždějící záchranná vozidla, tak policy poskytuje rezervaci pouze pro vozidla v těchto pruzích, což znamená, že vozidla před záchrannými vozidly také dostanou pritoritu. Díky tomuto se pruhy, ve kterých jedou záchranná vozidla vyprázdní poměrně rychle a tedy záchranné vozy se moc nezpozdí.

Toto mi přide jako chytrá implementace a nic bych na ní neměnil.

Ošetření nehod

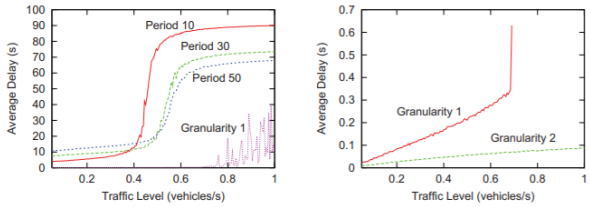
V dnešním provozu je naprostá většina nehod způsobena lidmi. Málo z nich je způsobeno technickou závadou, počasím či jiným faktorem. V době plně autonomních vozidel však nebudeme na silnicích potkávat řidiče nebo jen minimální množství, a tak by počet dopravních nehod měl zásadně klesnout. Nicméně bychom se měli zamyslet, co budeme dělat, pokud by v inteligentní křižovatce přeci jen došlo ke srážce. Abychom mohli vyřešit tuto situaci, tak IM by měl neustále monitorovat prostor křižovatky, detetekovat vozidla, která poruší pravidla své rezervace a okamžitě reagovat na vzniklou situaci. Dresner a Stone obohatili komunikační protokol o speciální nouzový signál informující o nehodě v křižovatce. Jakmile IM zjistí, že došlo k nehodě, tak hned začne zamítat nové žádosti. Nouzový signál je nutné vysílat všem vozidlům v křižovatce a v jejím okolí, a to opakovaně, aby ho měla šanci zachytit všechna vozidla. Chování IM a DA se změní následujícím způsobem:

- žádné další rezervace nebudou přijaty
- vozidla, která mohou zastavit před vjezdem do křižovatky, by tak měla učinit
- vozidla, která jsou v křižovatce, by neměla dále předpokládat, že těsné míjení nezpůsobí kolizi
- pokud se můžou vyskytovat i lidští řidiči, tak IM zapne na všech světlech červenou

Hlavním cílem nouzového signálu je ochránit co nejvíce vozidel před dalšími srážkami. Jelikož se však očekává, že nehody nebudou příliš časté, tak není prioritou IM obnovit provoz v křižovatce.

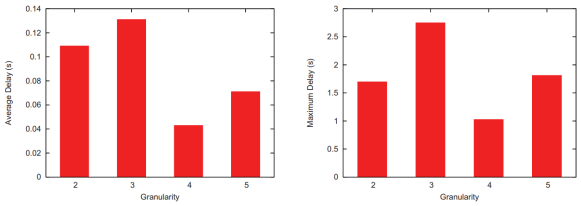
Výsledky provedených experimentů

Nejprve se porovnávaly průměrné časy zpoždění vozidel v křižovatce pro FCFS strategii a dnešní světelné křižovatky – Obrázek 4. Světelné křižovatky jsou zde porovnávány pro více period, což je doba, za jak dlouho se přepne světlo v křižovatce. Tedy vidíme, že pro méně vytížené křižovatky jsou lepší menší periody, zatímco pro více vytížené křižovatky jsou to vyšší periody. Dále na Obrázku



(a) Reservation vs. Traffic Light (b) Increasing Granularity

Figure 4: Porovnání průměrného zpoždění v křižovatce pro FCFS strategii a světelné křižovatky.



(a) Average delay (b) Maximum delay

Figure 5: Porovnání granularit křižovatek a jejich vliv na zpoždění.

4 můžeme vidět, že průměrné zpoždění pro FCFS strategii je mnohem menší než pro světelné křižovatky. Dále se zkoumalo, jaká granularita křižovatky je nejlepší. Mohlo by se zdát, že čím větší granularita tím lepší. Ale jak je vidět na Obrázku 5, není tomu tak.

Také se zkoušel experiment, kde bychom povolili odbočování z každého pruhu vs odbočovací pruhy jsou předem určeny – Obrázek 6. Obecně by se zdálo, že možnost odbočování z každého pruhu by mělo být lepší, protože se jedná o relaxaci odbočování z předem určených pruhů, ale jak ukazují experimenty, tak ke zlepšení nedošlo, naopak došlo k mírnému zhoršení.

Co kdyby nebyla všechna vozidla autonomní? Výsledky jsou vidět na Obrázku 7. Můžeme si všimnout, že pro 100% human a 10% human rozdíly nejsou moc velké.

Posledním experimentem, který uvedu, je, jak se liší doba zpoždění při průjezdu záchranných vozidel oproti normálním vozidlům – Obrázek 8. Záchranná vozidla se objevují v 0.1% případů. Z Obrázku 8 je vidět, že záchranná vozidla mají menší zpoždění při průjezdu křižovatkou než ostatní vozidla.

Kdybych měl celkově zhodnotit přístup Dresnera a Stona, tak se mi velice líbí. Je vidět, že to měli všechno hodně promyšlené.

Další přístupy k řešení

Jak jsme již zmínili v Úvodu, tak problém křižovatky můžeme chápat jako speciální problém online MAPF. Musíme být ale schopni problém křižovatky reprezentovat jako online MAPF problém (Škopková 2019). Předpokládejme, že každý agent se v každém časovém okamžiku nachází právě na jednom políčku křižovatky a že

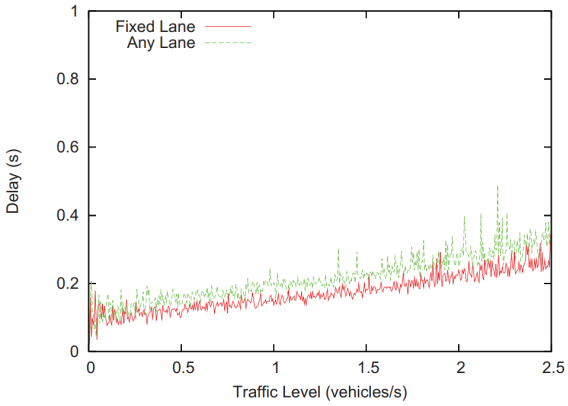


Figure 6: Odbočování z předem určených pruhů a možnost odbočování ze všech pruhů.

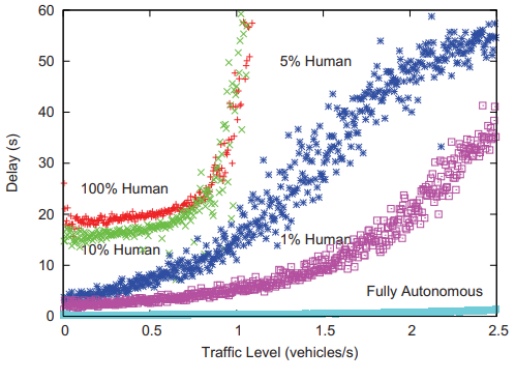


Figure 7: Porovnání zpoždění při průjezdu křižovatkou pro různé poměry autonomních a neautonomních vozidel.

se agenti pohybují synchronizovaně (všechna vozidla jedou stejnou rychlostí). Potom se nabízí křižovatku reprezentovat jako graf $G = (V, E)$, kde V je množina vrcholů grafu pokrývajících veškerý prostor křižovatky a E je množina dvojic $(u, v) \in V$ takových, že z místa u se lze přímo přemístit do místa v. Potom můžeme problém křižovatky chápat jako speciální typ online MAPF problému, kde pro agenta a_i odpovídá s_i ústí křižovatky v jeho příjezdovém směru, g_i ústí křižovatky ve směru odjezdu a plán pro agenta a_i popisuje akce, pomocí nichž se dostane z s_i do g_i. Jedná se tedy o online MAPF problém. Startovní a cílové vrcholy všech agentů jsou navzájem různé. Navíc agent, který křižovatku jednou opustí, se už do ní znovu nevrací, což znamená, že v určitém časovém kroku z plánu úplně zmizí. Poté pro řešení můžeme již použít nějaký z algoritmů, který řeší online MAPF problém – ty už jsme zmínili v Úvodu.

Tato generalizace problému je použitelná, nicméně v reálném případě si to moc nedokáži představit. Zdá se mi, že takováto křižovatka by nemusela být úplně efektivní a to z toho důvodu, že každá křižovatka by měla být připravena na to, že jí může projet nějaký dlouhý kamion. Nicméně většinu času ji projíždí osobní auta, která jsou mnohem

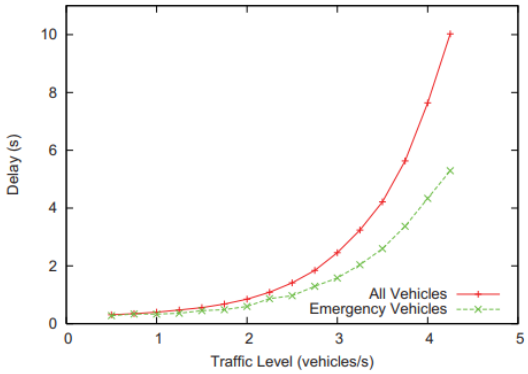


Figure 8: Zpoždění záchranných vozidel oproti normálním vozidlům.

kratší – třeba 3x, 4x kratší než kamion, a když v každém časovém okamžiku je vozidlo na jednom políčku křižovatky, tak ta políčka musí být dostatečně velká, aby se do nich vešel kamion. Pak ale když se v křižovatce objeví zástup osobních automobilů, tak ty pak mezi sebou budou mít obrovské rozestupy. Mohlo by se to nějak speciálně ošetřit, že bychom si určili rozměr políčka, aby se do nich vešla většina vozidel (to by se mohlo zjistit nějakých statistickým měřením), a pokud by se objevilo nějaké obnormálně velké vozidlo, tak by se mohlo reprezentovat jako více za sebou jedoucích menších vozidel. Tento přístup k MAPF, kde uvažujeme, že daný agent zabírá více políček, již existuje (Jiaoyang et al. 2019).

Nalezení optimálního řešení pro problém křižovatky je NP-těžký problém, jelikož se jedná o zobecnění NP-úplného problému Loydovy patnácky (Škopková 2019). Tak se nabízí jako řešení převést problém křižovatky na jiný NP-těžký problém, pro který již existují kvalitní řešiče a vyřešit tento problém. Jedná se o tzv. deklarativní přístup. Ten je založen na vytváření podmínek, které musí splňovat validní řešení MAPF, v nějakém deklarativním jazyce a ty se poté předají řešiči. Ten najde řešení splňující dané podmínky a tím najde řešení MAPF. Příkladem deklarativního jazyku, který se hodí pro modelování MAPF, je jazyk Picat.

Jelikož je typické kódování MAPF založené na grafu a přechody mezi jednotlivými vrcholy, tak se pro něj vztahují připomínky, které jsem uvedl v této sekci výše.

Křižovatka jako online MAPF

Nyní se budeme zabývat porovnáním různých typů křižovatek (Škopková et al. 2020). Pro jednoduchost předpokládáme, že DA jsou homogenní a jedou stejnou rychlostí. Opět se posílají žádosti, které se potvrzují nebo zamítají, a reprezentujeme křižovatku opět pomocí granularity. V každém časovém okamžiku jeden agent může být pouze na jednom políčku, agenti se pohybují synchronně. Každé políčko má maximálně 4 sousední políčka – diagonální políčka nejsou sousední. Prostor křižovatky můžeme popsat následovně: políčka jsou vrcholy grafu a sousední políčka jsou spojeny hranou. Reprezentace reálné

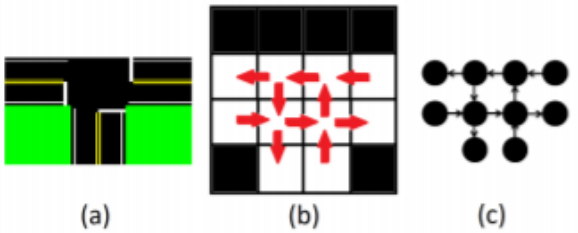


Figure 9: Ukázka převodu křižovatky do grafové podoby.

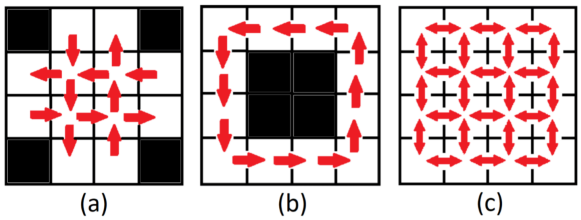


Figure 10: a) klasická křižovatka, b) kruhový objezd, c) free movement

křižovatky je ukázána na Obrázku 9. Poté můžeme aplikovat nějaký online MAPF algoritmus na tento graf. Zkoumaly se tři křižovatkové modely, kde dva se používají v reálném světě – klasická křižovatka a kruhový objezd, a jeden, který dává agentovi volnost pohybu (budeme ho označovat jako free movement) – viz Obrázek 10. Každá žádost se skládá z příchozího a odchozího směru a požadovaného času pro vstup do křižovatky.

Obecně u reprezentace křižovatky grafem, mám trochu problém s touto reprezentací, která je zobrazená na Obrázku 10 v části a. Znázornil jsem to na Obrázku 11, kde si představme, že by jelo více vozidel proti sobě a všechna by chtěla zabočit doleva. Tak v této reprezentaci křižovatky dojde ke křížení tras, zatímco v dnešním stylu křižovatky se tyto dvě trasy nekříží – v obrázku nejpravější část. Přemýšlel jsem, jak by to šlo udělat lépe.

Napadla mě reprezentace, která je zobrazena na Obrázku 12. Nicméně tato reprezentace je složitější a porušuje klasickou reprezentaci křižovatky, kde všechna políčka jsou stejně velká. To by ovšem šlo vyřešit podrozdělením velkých políček na menší. Abych ten Obrázek 12 trochu vysvětlil, tak jsem se inspiroval v dnešních křižovatkách a zakreslil jsem, jak zhruba fungují. Příjezd z každého směru je označen různou barvou a tato barva označuje, jak by dané vozidlo mohlo projíždět křižovatkou. Nicméně to jsem udělal z důvodu přehlednosti, pokud bychom se drželi značení, které používali ve článku, tak bychom mohli všechny šipky označit stejnou barvou a nahradili bychom více šipek různé barvy, které jdou ze stejného políčka do stejného políčka (celkově jsou čtyři takovéto přechody), za pouze jednu šipku. Co bych dále zavedl, je možnost swapování dvou vozidel mezi dvojicemi políček 1-5, 2-5, 3-5, 4-5. Tímto bych částečně vyřešil problém odbočování vlevo - všechna vozidla by samozřejmě nemohla být současně na políčku 5,

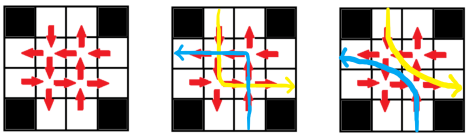


Figure 11: Porovnání průjezdů křižovatkou.

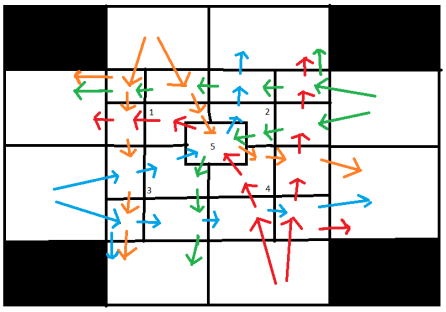


Figure 12: Návrh křižovatky.

ale kdyby jedno trochu zrychlilo nebo druhé trochu zpomalilo, tak by se mohla swapnout. Celý graf křižovatky by tedy vypadal jako na Obrázku 13.

Zpátky k článku. Před každým experimentem si autoři článku určili maximální zpoždění pro každý vjezd do křižovatky. Žádosti, pro které není žádná volná cesta mezi požadovaným časem a požadovaným časem + zpožděním, jsou zamítnuty. Ve většině případů používají adding-solver, který, jak jsme zmínili už v Úvodu, nenarušuje již nalezené cesty předchozích žádostí a snaží se najít takovou cestu, která s nimi nebude kolidovat. V nějakých experimentech též použili rescheduling-solver tedy takový, který může měnit i cesty, které již nalezl. Ovšem rescheduling-solver je použitený pouze pro případ free movement křižovatky, protože první dva typy křižovatky mají přesně danou cestu.

Experimenty

V první části experimentů porovnávají křižovatky z Obrázku 10. V každém experimentu vygenerovali pět náhodných sekvencí vstupních žádostí a zprocesovali je přes danou testovanou křižovatku. V experimentech je také použit termín *prostorový limit*, což je maximální přípustné zpoždění při vjezdu do křižovatky. Pro generovaní žádostí použili uniformní distribuci. Výsledky se pozorovali pro tři metriky:

- Délka plánu – počet diskrétních časových kroků potřebných k transportu všech agentů do jejich cílů podle plánu. Experiment začíná z času 1.
- Zpoždění – kumulativní počet časových kroků, které každý agent musí strávit v křižovatce (nebo časem čekáním před vstupem do křižovatky), děleno minimálním počtem časových kroků, které by to zabralo agentovi k dosažení cílové pozice, kdyby se nevyskytovali žádní další agenti.

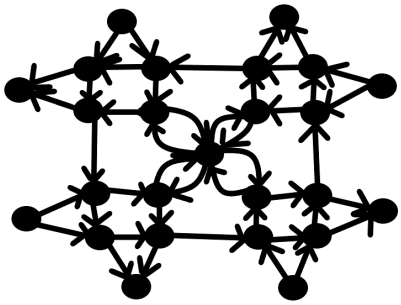


Figure 13: Návrh křižovatky - graf.

Typ křižovatky	délka plánu	zpoždění	zamítnuto
Kruhový objezd	15.6	1	**0**
Klasická křižovatka	**13**	2.4	0.2
Free - adding	**13**	**1.4**	**0**
Free - rescheduling	**13**	**0.8**	**0**

Table 1: Porovnání typů křižovatek pro málo hustou dopravu. Prostorový limit 1.

- Zamítnutí – počet žádostí, které nebylo možno provést v daném prostorovém limitu. Žádosti, které byly zamítnuté se vůbec přes křižovatku nepohybovaly.

V tabulkách 1, 2, 3, 4, 5, 6 vidíme výsledky. U každé tabulky je uvedeno, pro jak hustou dopravu daná čísla vyšla.

Z tabulky 1 vidíme, že nejlepší výsledky dává křižovatka free movement - rescheduling, kruhový objezd má nejdelší délku plánu a klasická křižovatka má jako jediná určitý počet zamítnutých žádostí.

V tabulce 2 si opět vede nejlépe free movement, ale tentokrát se jedná o adding. Je to způsobeno tím, že rescheduling nestačí počítat, jinak by samozřejmě nikdy nemohlo dostat horší výsledek než adding. Z toho důvodu se poté v dalších experimentech již rescheduling neobjevuje. Opět si můžeme všimnout, že délka plánu je pro kruhový objezd delší než pro ostatní přístupy, nicméně zpoždění je pro kruhový objezd vůbec nejmenší.

V tabulkách 3, 4 porovnáváme hustou dopravu pro různé prostorové limity. V obou případech free movement odmítla nejmenší množství žádostí. Můžeme si všimnout, že s rostoucím prostorovým limitem se zvyšuje zpoždění a snižuje počet zamítnutých žádostí, což je z definice prostorového limitu poměrně logické. Délka plánu zůstává zhruba stejná.

V tabulkách 5, 6 jsou zobrazeny výsledky pro velmi hustou dopravu pro různé prostorové limity. Opět můžeme vidět, že se zvyšujícím se prostorovým limitem se zvýšilo zpoždění a bylo zamítnuto méně žádostí. Dále kruhový objezd má opět nejdelší délku plánu a free movement nejmenší počet zamítnutých žádostí.

Posledním experimentem, který zde uvedu, je porovnání klasické křižovatky a free movement křižovatky pro granularitu 8. Pro klasickou křižovatkou s granularitou 8, se zkouší 3 možnosti, jak se agenti můžou pohybovat – zo-

Typ křižovatky	délka plánu	zpoždění	zamítnuto
Kruhový objezd	20.2	**2.2**	0.8
Klasická křižovatka	**17.2**	4.4	1.6
Free - adding	17.4	5.8	**0.4**
Free - rescheduling	18	4.75	1

Table 2: Porovnání typů křižovatek pro středně hustou dopravu. Prostorový limit 1.

Typ křižovatky	délka plánu	zpoždění	zamítnuto
Kruhový objezd	18.4	**5.2**	4.2
Klasická křižovatka	**14**	8.4	3.4
Free adding	15	10.8	**2**

Table 3: Porovnání typů křižovatek pro hustou dopravu. Prostorový limit 1.

Typ křižovatky	délka plánu	zpoždění	zamítnuto
Kruhový objezd	18.8	20	0.6
Klasická křižovatka	16.4	31.4	0.4
Free adding	**15.4**	**19.2**	**0**

Table 4: Porovnání typů křižovatek pro hustou dopravu. Prostorový limit 5.

Typ křižovatky	délka plánu	zpoždění	zamítnuto
Kruhový objezd	17.2	**15**	8.2
Klasická křižovatka	**14**	21.4	9.6
Free adding	14.8	36.2	**5.8**

Table 5: Porovnání typů křižovatek pro velmi hustou dopravu. Prosotorový limit 2.

Typ křižovatky	délka plánu	zpoždění	zamítnuto
Kruhový objezd	19	**47**	3.6
Klasická křižovatka	17	59.2	3.8
Free adding	**16.2**	63.6	**1.6**

Table 6: Porovnání typů křižovatek pro velmi hustou dopravu. Prostorový limit 5.

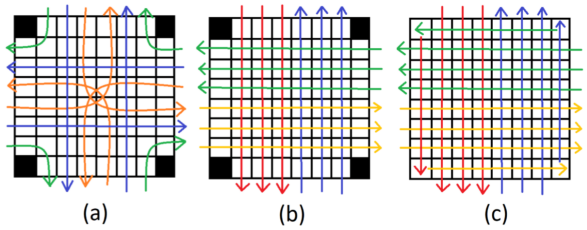


Figure 14: Nastavení koridoru v křižovatkách. a - striktní pruhy, b - volné pruhy, c - orientovaný graf.

Typ křižovatky	délka plánu	zpoždění	zamítnuto
Koridor - striktní	17.8	**12.4**	1.2
Koridor - volné	18.5	15.8	1.2
Koridor - graf	18.4	17	1
Free - adding	**17.6**	13	**0**

Table 7: Porovnání typů křižovatek pro granularitu 8.

brazeny na Obrázku 14. Výsledky jsou zobrazeny v tabulce 7. Nejlépe si opět vedla křižovatka free movement. Ovšem nejmenší zpoždění měla křižovatka se striktními pruhy a ze všech koridor křižovatek, bych řekl, že si vedla nejlépe.

Celkově bylo vyzkoumáno, že kruhový objezd nezpůsobuje velké zpoždění pro agenty, ale celková délka trasy je poměrně delší než u dalších přístupů. Free movement křižovatka má málo zamítnutých žádostí, ale za cenu vyššího zpoždění, což je způsobeno větší flexibilitou.

Můj celkový názor na tento článek je, že free movement křižovatky si moc nedokáži představit, že by někdy mohly fungovat v reálném světě, nicméně fungují jako odrazový můstek k porovnávání, jelikož se jedná o nejuniverzálnější přístup. Tedy můžeme zkoumat, o jak moc jsou přístupy, které se používají dnes – kruhový objezd a klasická křižovatka – horší. A můžeme vidět, že byť se free movement křižovatka projevuje jako nejlepší varianta, tak klasické přístupy nejsou nějak mnohonásobně horší.

Shrnutí

V tomto článku jsme si zadefinovali online MAPF problém a problém křižovatky, kde se objevují noví agenti již při rozvrženém plánu. Ukázali jsme různé přístupy k řešení problému křižovatky a přidali jsme naše názory na možná vylepšení. Porovnali jsme dnes používané typy křižovatek s univerzální free movement křižovatkou.

Reference

Dresner, K. M. and Stone, P. (2008). *A multiagent approach to autonomous intersection management. J. Artif. Intell. Res., 31:591–656*

Věra Škopková (2019). *Diplomová práce - Inteligentní křižovatka. MFF UK, Praha*

Jiří Švancara, Marek Vlk, Roni Stern, Dor Atzmon, Roman Barták (2019). *Online Multi-Agent Pathfinding, AAAI-19: 7732-7739*

Věra Škopková, Roman Barták, Jiří Švancara (2020). *What*

Does Multi-agent Path-finding Tell Us About Intelligent Intersections, ICAART 2020: 250-257
Jiaoyang Li, Pavel Surynek, Ariel Felner, Hang Ma T. K. Satish Kumar Sven Koenig (2019). *Multi-Agent Path Finding for Large Agents, AAAI-19: 7627-7634*