Problém hledání cest pro více agentů s různými tvary

Kamil Závorka
kamil.zavorka@seznam.cz

Abstrakt

Hledání cest pro více agentů (MAPF) je jedním z problémů umělá inteligence. Tento problém má svou typickou reprezentaci, na jejíž téma bylo provedeno mnoho výzkumů. Praktické použití nicméně vyžaduje, aby byla tato klasická reprezentace upravena vzhledem ke konkrétní situaci, kterou se snažíme řešit. Tento referát se zabývá modifikací MAPF problému, ve které předpokládáme, že jednotliví agenti mají vlastní tvar, který zabírá určitý prostor. V referátu se pokusíme sumarizovat různé přístupy a práce, které se tímto tématem zabývají.

1 MAPF problém

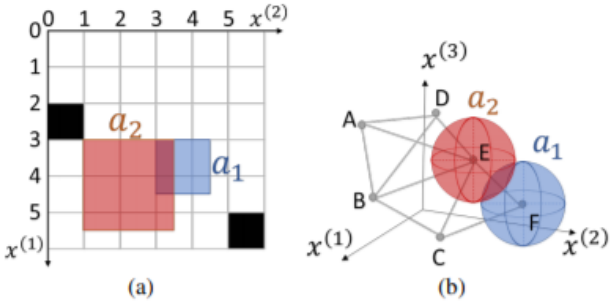
Základní problém hledání cesty pro více agentů můžeme definovat tak, že máme libovolný graf G = (V, E) a množinu k agentů $\{a_1,\ldots,a_k\}$. Každý z nich má svůj počáteční a cílový vrchol v grafu G. Všichni agenti jsou umístěni ve vrcholech grafu G, přičemž jejich pozice jsou navzájem různé. Cílem je nalézt cesty pro všechny agenty z jejich počátečních stavů do jejich koncových stavů, kde k přesunu agentů mezi vrcholy grafu využíváme hrany. Tyto cesty, které pro agenty nalezneme také musí splňovat to, aby neobsahovaly konflikt mezi agenty. Konfliktem označíme situaci, kdy jsou 2 agenti ve stejném vrcholu, nebo přecházejí po stejné hraně proti sobě (dojde k jejich srážce).

Typicky nás zajímá řešení, které je nějakým způsobem optimální např. tak, že celková doba nutná k vykonání plánu je minimální. Tuto základní definici problému pak lze podle konkrétní situace rozšířit o další vlastnosti, jako přiřazení doby trvání pohybu agenta po hraně nebo zvážení kinematických omezení jako akcelerace. Jedním z takových rozšíření je i myšlenka toho, že agenti nejsou pouze jeden bod, ale celý útvar, který zabírá určitý prostor.

Velcí agenti

Pod pojmem velcí agenti označujeme agenty, kteří nejsou omezeni pouze na jediný bod, ale mají vlastní geometrický tvar, kterým zabírají určitý prostor. Dále předpokládáme, že tento tvar zůstává stále stejný a agent není schopen se jakkoliv otáčet. Přestože agenti mají svůj tvar, musíme stále uvažovat nějakou pozici, která reprezentuje vrchol, ve kterém se nacházejí. Volba této pozice je individuální (u kruhu můžeme použít střed, u obdélníku levý dolní roh), nicméně tato pozice musí zůstat stejná po celou dobu plánování.

Rozšíření agentů o jejich geometrický tvar přináší do problému řadu situací, které musíme vyřešit. Hlavní změna je v definici kolizí. V základním MAPF problému jsou nejtypičtější kolize dvou typů – vrcholová (2 agenti se nacházejí ve stejném vrcholu) a hranová (2 agenti využívají stejnou hranu). V závislosti na konkrétní úloze mohou existovat i další druhy kolizí (např. nechceme, aby agent navštívil vrchol, který jiný agent v daném kroku uvolní – toto omezení je může být motivováno právě rozměry agentů a kinematickými omezeními). Pro velké agenty však může dojít ke kolizi i v dalších situacích, protože kvůli svému tvaru zabírají větší prostor. V tomto prostoru se mohou nacházet i jiné, nepoužívané hrany a vrcholy grafu. Obrázek 1 tuto situaci ukazuje.



Obrázek 1: Rozmístění velkých agentů v (a) dvourozměrné mřížce, (b) trojrozměrném grafu. V obou případech jsou agenti ve vzájemné kolizi, přestože nesdílí stejný vrchol, ani nepoužívají stejnou hranu.
Zdroj: (Li a kol. 2019)

2 CBS (Conflict-based search)

Jedním z řešících algoritmů pro LA-MAPF (problém hledání cest pro více velkých agentů) je upravený algoritmus CBS (Li a kol. 2019).

Základní verze CBS pro řešení MAPF (Sharon a kol. 2015) je založena na dvojitém prohledávání. Prohledávání na vyšší úrovni hledá optimální řešení v prostoru množin cest pro množinu agentů. Tento prostor je reprezentovaný stromem, ve kterém každý uzel obsahuje konkrétní množinu cest pro jednotlivé agenty. Pokud jsou tyto cesty v daném uzlu přípustné (nedochází mezi nimi k žádnému konfliktu), tato množina je hledané řešení problému. Pokud cesty nejsou přípustné, daný uzel se rozdělí na dva potomky s jinými cestami a prohledávání pokračuje. V základní variantě se zde používá prohledávání do šířky.

Každý uzel tohoto stromu také obsahuje množinu podmínek, které musí splňovat všechny cesty v daném uzlu. Pokud jsou cesty navštíveného uzlu odmítnuty kvůli konfliktu, vygenerujeme z tohoto uzlu 2 potomky, do nichž přidáme takové nové podmínky, aby nalezenému konfliktu předešly.

Podmínky mohou být tvaru:

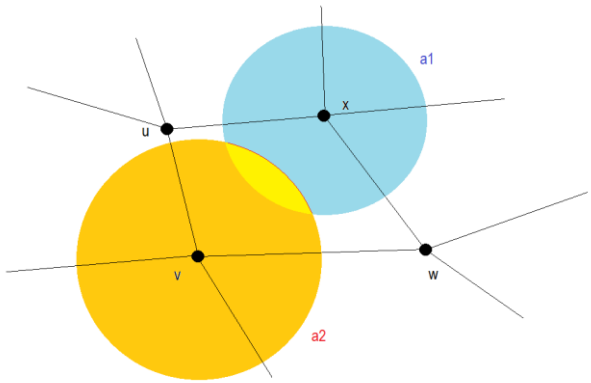
- $\langle a,v,t \rangle$ - vrcholová podmínka – zakazuje přítomnost agenta a ve vrcholu v v čase t.
- $\langle a,u,v,t \rangle$ - hranová podmínka – zakazuje pohyb agenta a z vrcholu u do vrcholu v v čase t.

Druhé prohledávání, které CBS dělá, je prohledávání na nižší úrovni, které hledá konkrétní cesty pro agenty v grafu G. Toto prohledávání hledá cestu pro každého agenta individuálně (ignoruje cesty ostatních agentů) a bere do úvahy množinu podmínek přítomných v daném uzlu stromu.

Algoritmus CBS začne s prohledávacím stromem na vyšší úrovni o jednom uzlu. Tento uzel je kořen, který obsahuje prázdnou množinu podmínek C = {} a množinu cest pro všechny agenty. Tyto cesty jsou nalezeny algoritmem prohledávání na nižší úrovni a pro kořen platí, že jsou to individuální nejkratší cesty. U cest ověříme, zda mezi nimi nedochází ke konfliktu. Pokud konflikt není přítomen, množina cest je optimální řešení problému. Pokud konflikt existuje (předpokládejme např. vrcholový konflikt agenta a_i a agenta a_j ve vrcholu v v čase t), vytvoříme 2 potomky v prohledávacím stromě. Oba potomci zdědí z rodiče všechny podmínky a všechny cesty agentů. V obou potomcích pak zakážeme přítomnost jednoho z agentů v konfliktním místě v daném čase ($C_1 = C \cup \{\langle a_i,v,t \rangle\}$, $C_2 = C \cup \{\langle a_j,v,t \rangle\}$) a znovu vygenerujeme cesty s ohledem na nové podmínky. Jelikož jsme do každého potomka přidali podmínku, která se týká pouze jednoho agenta, stačí v daném potomkovi přegenerovat cestu pomocí prohledávání na nižší úrovni pouze pro tohoto ovlivněného agenta a cesty ostatních agentů nechat stejné, jaké byly zděděny z rodiče.

CBS pro LA-MAPF

CBS algoritmus (Sharon a kol. 2015) může být použit pro velké agenty velice jednoduše. Stačí místo klasických hranových a vrcholových kolizí detekovat také kolize geometrických tvarů agentů a pro takového kolize též zakazovat přítomnost jednoho, nebo druhého agenta v daném bodě. Toto upravení je sice validní, nicméně výsledná efektivita algoritmu není příliš dobrá. To je způsobeno hlavně tím, že při generování potomků ve stromu na vyšší úrovni vždy přibyde pouze jedna podmínka, zakazující přítomnost agenta v jednom vrcholu. Pro agenty, kteří však zabírají větší prostor existuje takových pozic, které chceme zakázat, větší množství.



Obrázek 2: Agenti *a1* a *a2* kolidující svými geometrickými tvary v čase *t*.

Předpokládejme situaci jako na Obrázku 2 – standardní CBS by nejprve přidalo podmínku $\langle a2,v,t \rangle$, poté $\langle a2,u,t \rangle$ a ve třetí vrstvě $\langle a2,w,t \rangle$. Jelikož přidá pouze jednu podmínku v každé úrovni stromu, toto postupné přidávání prodlouží prohledávaný strom o 3 vrstvy.

Vícepodmínkový CBS

Pokud víme že můžeme pro daného agenta zakázat více pozic najednou (jako na obrázku 2), mohli bychom zkusit tyto další podmínky přidat do množiny při jediném generování potomků. Toto přidání ušetří část prohledávané hloubky stromu, díky čemuž se zbavíme několika běhů hledání cest na nižší úrovni.

Přidávat podmínky lze více způsoby, klasický způsob je ten, že pro agenta zakážeme všechny body a hrany, ve kterých se dostává do kolize s druhým agentem. Alternativa k tomuto řešení je ta, že zakážeme některé tyto pozice i pro různé časy. Podmínky pro různé časy jsou však komplikovanější, protože je těžké přesně rozhodnout, zda nám takováto podmínka opravdu pomůže předejít kolizi a nezablokuje nějaké vhodné řešení.

To, jaké podmínky přidáváme do potomků se může řídit dvěma přístupy:

- Asymetrický přístup – Množinu podmínek přidáváme pouze do jednoho potomka, zatímco v druhém pouze zakážeme jeho přítomnost v kolizním vrcholu. Uvažme příklad na obrázku 2. Pokud bychom tvořili potomky asymetrickým přístupem, výsledné množiny podmínek pro oba potomky by vypadaly takto:

  $C_1 = C \cup \{<a2,v,t>, <a2,u,t>, <a2,w,t>, <a2,x,t>\}$
  $C_2 = C \cup \{<a1,x,t>\}$

  Tedy pro jednoho potomka jsme zakázali přítomnost agenta a2 ve všech vrcholech, ve kterých dochází ke kolizi s agentem a1, zatímco pro druhého potomka jsme pouze zakázali přítomnost agenta a1 ve vrcholu, ve kterém dochází ke kolizi.

- Symetrický přístup – Přidává do obou vytvořených potomků více podmínek. Myšlenka toho přidávání je taková, že v oblasti, která je průnik tvarů agentů a která je tudíž kolizní, vytvoříme umělý bod P. Podmínky, které pak do obou potomků přidáváme, zakazují agentům použít vrcholy a hrany takové, že v nich jejich geometrický tvar obsahuje bod P (takto vytvořená množiny podmínek C_1' se liší od množiny C_1 vytvořené asymetrickým přístupem.

Oba přístupy přidávání podmínek jsou validní. Existují případy, kde asymetrický přístup dokonce vygeneruje více podmínek než přístup symetrický. Nicméně autoři článku (Li a kol. 2019) experimentálně prokázali, že symetrický přístup je v průměru výkonnostně lepší, než asymetrický.

Řízené prohledávání

Vylepšené CBS (Boyarsky a kol. 2015) vylepšuje základní CBS definováním hlavních konfliktů. Hlavní konflikt je takový konflikt, u kterého zjistíme, že rozdělením uzlu získáme 2 potomky, kteří oba mají celkovou cenu svého plánu větší, než rodič (délka nově nalezené cesty pro ovlivněného agenta je v obou případech delší, než nejdelší z cest v rodiči). Pro tyto hlavní konflikty platí, že jejich přednostní vyřešení zmenšuje prohledávaný prostor. Vylepšené CBS je tedy založeno na použití heuristiky využívající právě detekci hlavních konfliktů k řízení prohledávání.

Jelikož je jasně, že můžeme v rámci CBS pro velké agenty při dělení vrcholu navrhnout více způsobů, jak přidávat podmínky, chtěli bychom umět vybrat nejvýhodnější množinu podmínek. Tedy takovou množinu, která maximalizuje ceny potomků a využít tak optimalizačního principu použitého u vylepšeného CBS.

Pro počítání vah množin podmínek navrhli (Li a kol. 2019) použití vícehodnotového rozhodovacího diagram (MDD). MDD_i^n je orientovaný acyklický graf, který obsahuje všechny cesty délky n pro agenta a_i vedoucí od startu do cíle. Pokud nějaká množina podmínek pro tohoto agenta blokuje všechny vrcholy v nějaké hloubce h, tato množina podmínek zároveň blokuje všechny cesty délky n a tudíž nejkratší cesta musí mít minimálně délku n+1. Pro agenta použijeme MDD_i^{n+d} graf, kde d nazýváme hloubka pohledu dopředu. Z tohoto grafu pak můžeme předpovědět, jaký nárůst v ceně cesty agenta očekáváme a tento nárůst prohlásíme za váhu množiny podmínek. Na základě těchto vah získaných z pohledu dopředu, dokážeme říci, jak dobrá je množina podmínek, kterou se chystáme do potomka přidat (ve skutečnosti se zohledňuje součet vah množin podmínek vytvářených pro oba potomky).

Kromě zvolení vhodných podmínek lze tyto váhy použít také k tvorbě heuristiky. Zatím jsme pro prohledávání na vyšší úrovni uvažovali pouze algoritmus prohledávání do šířky. Váhy, které umíme počítat pro množiny podmínek, lze však využít i k tvorbě heuristiky, díky níž můžeme pomoci řídit prohledávání tak, aby se důležitější konflikty (konflikty, které prodlouží celkovou cenu řešení) řešily přednostně, obdobně jako u vylepšeného CBS.

Tvar agentů

Zatím jsme pro CBS používali pouze podmínky, které zakazovaly konkrétnímu agentovi konkrétní vrchol v konkrétním čase. Tímto způsobem ovšem můžeme generovat poměrně velké množství podmínek, které musíme udržovat a kontrolovat. Toto množství bude tím větší, čím více vrcholů geometrický tvar agenta obsadí (pokud máme veliké agenty na drobné mřížce, jejich geometrický tvar může pro ostatní agenty zakázat velké množství vrcholů a hran). Zakazování jednotlivých vrcholů, ale nemusí být jediný způsob zadání podmínky. Pokud pracujeme s agenty majícími pravidelný tvar, můžeme například vytvořit podmínky, které pro nějakého agenta zablokují celou oblast (například pro kruhové agenty můžeme zakázat okruh konfliktního vrcholu). Takové podmínky jsou stejně silné, jako výčet zakázaných vrcholů a zároveň se lépe a rychleji kontrolují při hledání konkrétní cesty prohledáváním na nižší úrovni. Tímto zlepšením můžeme ušetřit trochu výkonu během řešení problému.

Zároveň se můžeme zamyslet, zda v některých případech, ve kterých máme nepravidelné agenty, nepřinese zlepšení a zjednodušení celého algoritmu aproximace tvaru agenta pomocí nějakého pravidelného tvaru.

3 Použití meta-agentů

CBS pro velké agenty umožňuje nalézt optimální řešení pro problém hledání cest pro velké agenty. Jako všechny z technik, které se zabývají řešením NP-těžkých problémů, je však na některé instance problému lepší a na některé horší. Pro CBS, jako ostatní techniky, které jsou založené na řešení konfliktů cest, roste složitost s počtem kolizí. Čím více kolizí mezi jednotlivými agenty je, tím více se zvětšuje prohledávaný prostor. Pro velké agenty je tento problém ještě znatelnější, protože kvůli svým rozměrům dochází častěji ke kolizím, než u jednobodových agentů. Naopak algoritmy založené na prohledávání pomocí např. A* se mohou v případech, kde je mnoho kolizí, chovat lépe. Právě pro tyto případy bylo navrženo použití meta-agentů pro CBS (Sharon a kol. 2012).

Myšlenka tohoto zlepšení je taková, že v CBS definujeme mez pro kolize. Zároveň si pro každou dvojici agentů ukládáme, ke kolika kolizím mezi nimi došlo během prohledávání. Pokud tento počet kolizí pro určitou dvojici přesáhne nastavenou mez, rozhodneme se tyto dva agenty spojit do meta-agenta. Tento meta-agent je struktura, která agenty sdružuje a přestane řešit vzájemné kolize mezi nimi. Toto spojení pomáhá zlepšovat výkon CBS tím, že agenty, kteří hodně kolidují, uzavřeme do skupiny, čímž redukujeme množství přidávaných podmínek a vytvářených a prohledávaných větví. Takto sdružené agenty pak speciálně řešíme prohledáváním na nižší úrovni tak, že optimální cestu hledáme pro všechny agenty v meta-agentovi najednou na nižší úrovni pomocí algoritmu založeném na A*, který se může s kolizemi lépe vypořádat.

Abychom to mohli realizovat, přidáme do CBS schopnost tyto meta-agenty vytvářet a to tak, že pokud při prohledávání na vyšší úrovni zjistíme kolizi, můžeme ji buď vyřešit klasicky rozdělením vrcholu na 2 potomky, nebo spojením agentů do meta-agenta.

Meta-agent se chová jako každý jiný agent – kolizi s meta-agentem definujeme jako kolizi s libovolným agentem, který je v něm obsažen. Podmínka, která zakazuje meta-agentovi přítomnost ve vrcholu v daném čase tuto přítomnost zakazuje všem agentům, kteří jsou v něm zahrnuti.

Dále musíme definovat strategii spojování. Hlavní věc, kterou musíme vyřešit je to, jak naložíme s již zadanými podmínkami pro oba spojované agenty (nebo meta-agenty). Pro spojované agenty a_i a a_j a jejich podmínky mohou nastat následující situace:

- Vnitřní – podmínka řešící konflikt mezi a_i a a_j
- Vnější – podmínka řeší konflikt mezi a_i nebo a_j a vrcholem, který nebude součástí nového meta-agenta.

Vnitřní konflikty řešit nemusíme, protože kolize mezi agenty v meta-agentovi budeme nadále řešit v prohledávání na nižší úrovni. Podmínky, které tedy budeme řešit se týkají ostatních agentů. Tyto podmínky ošetříme tak, že je rozšíříme na celého meta-agenta (pokud existuje podmínka $<a_i,v,t>$, uděláme z ní podmínku $<a_{ij},v,t>$, která zakazuje přítomnost ve vrcholu v v čase t pro všechny agenty, tvořící nového meta-agenta a_{ij}.

Implementací tohoto konceptu snížíme počet konfliktů, které musí CBS řešit a tím zmenšíme jeho prohledávaný prostor. Tyto časté konflikty naopak necháme řešit nižší vrstvu, která je může řešit efektivněji.

Přestože tato technika nebyla původně navržena pro velké agenty, myslím, že tento koncept by mohl pomoci vylepšit CBS pro velké agenty. V něm obecně očekáváme více kolizí. Tento počet bychom tedy mohli zredukovat tím, že rozdělíme všechny agenty na skupiny, jejichž agenti spolu hodně kolidují, ale skupiny mezi sebou mají kolizí málo. Kolize cest skupin pak vyřešíme prohledáváním na vyšší úrovni, pro nějž se výrazně zmenší větvení stromu, a kolize v samotných skupinách prohledáváním na úrovni nižší.

4 Řešení pomocí stromu s rostoucími cenami

Další algoritmus, který lze použít pro řešení problému plánování pro více velkých agentů je prohledávání stromu s rostoucími cenami (Increasing cost tree search), který je navíc navržen i pro počítání s různými délkami hran a různou dobou trvání akcí (Walker a kol. 2018). Základní předpoklad uvedeného algoritmu je, že máme nějaký algoritmus PTO (partial time overlap), který dokáže detekovat kolize agentů v prostoru (Ericson, 2004) v prostředí se spojitým časem.

ICTS algoritmus má podobně jako CBS 2 úrovně prohledávání. Na vyšší úrovni je prohledáván strom, jehož uzly obsahují k-tice $<C_1,\ldots,C_k>$. Tato k-tice reprezentuje ceny cest pro jednotlivé agenty. Základní otázka, kterou si klademe u vyššího prohledávání je, zda existuje přípustné řešení takové, že ceny cest pro jednotlivé agenty jsou $<C_1,\ldots,C_k>$. Pokud takové řešení neexistuje, vrchol rozdělíme tak, že v jednotlivých potomcích zkusíme různé ceny zvětšit (v základním ICTS zvětšujeme o hodnotu d = 1, pro spojité případy volíme reálné nezáporné d).

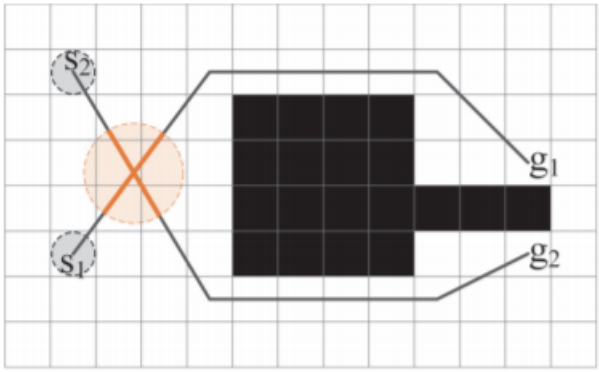
Na nižší úrovni prohledávání (prohledávání v každém jednotlivém uzlu) hledáme konkrétní cesty délky C_i. Ty sestrojíme tak, že pro agenta a_i, jehož hodnota C_i byla v nově vygenerovaném potomkovi změněna, vytvoříme orientovaný acyklický graf, který všechny tyto cesty délky C_i obsahuje. Důležité je, aby v tomto grafu nikde nebyl cyklus a žádná cesta tedy neobsahovala stejný vrchol vícekrát. Cesty z těchto grafů pro jednotlivé agenty jsou pak pomocí PTO vzájemně porovnávané s cílem najít nekonfliktní množinu cest.

Zde můžeme navrhnout možnost prořezání prohledávaného stromu na vyšší úrovni. Algoritmus nemusí nutně testovat všechny dvojice cest, zda jsou vzájemně nekonfliktní. Pokud existuje nějaká podmnožina m agentů, kde m < k, pro níž neexistuje přípustné řešení, můžeme tento uzel rovnou zamítnout a víme, že nám stačí vygenerovat následníky pouze takové, kteří upravují ceny pro onu m-tici agentů.

Tento algoritmus lze přímo použít pro velké agenty, jelikož jsme hned na začátku přímo zavedli použití PTO detekce. (Walker T. T. a kol. 2018) tento algoritmus dále rozšířili o myšlenky umožňující použití při spojitém čase a to v optimální variantě, nebo neoptimálních variantách, ve kterých můžeme určit, jak velkou neoptimalitu tolerujeme. Dále také popisují, jak lze v takovýchto případech řídit poměr mezi výkonnostní náročností a kvalitou řešení, nicméně takováto rozšíření už přímo nesouvisí s velkými agenty.

5 Další řešící přístupy

Kromě zmíněných algoritmů existují i další práce, které nabízejí jiný pohled na řešení problému hledání cest pro velké agenty. Jeden takový pohled je například využití podobnosti mezi CBS a Satisfiability modulo theories (SMT). Na základě této podobnosti Surynek (2019) navrhl algoritmus, který řeší MAPF problém pomocí rozhodovacích proměnných z SMT. Tento algoritmus je také rozšiřitelný pro velké agenty a spojitý čas.



Obrázek 3: Příklad mřížky a agentů s cestami pro prioritizované plánování.
Zdroj: (Yakovlev, Andreychuk 2017)

Yakovlev a Andreychuk (2017) zase navrhli algoritmus využívající prioritizované plánování. Prioritizované plánování používá trochu jiné prostředí než standardní MAPF problém. Uvažuje všechny agenty kruhového tvaru o stejném poloměru r. Prostředí je pevně ohraničená čtvercová mřížka, jejíž každá buňka má šířku 2r a tudíž dokáže přesně pojmout jednoho agenta. Agenti nemají přesně dané hrany, po kterých se pohybují. Mohou se v mřížce pohybovat z jedné buňky do libovolné jiné buňky (tyto buňky nemusí být nutně sousední) po přímé trase a každá buňka je volná pouze pokud do ní svým tvarem nezasahuje žádný agent. Příklad takového prostředí je znázorněn na Obrázku 4.

Prioritizované plánování vychází z konceptu, že každému agentovi je přiřazena priorita. Podle této priority přistupujeme k plánování. Nejdříve jsou naplánované cesty agentů s vyšší prioritou. Po té jsou naplánované cesty agentů s nižší prioritou tak, aby se vyhýbali jak překážkám v prostředí, tak kolizím s agenty vyšší priority. Samotný plánovací algoritmus je založený na A* prohledávacím algoritmu.

Další instance problému

Mimo uvedené přístupy a algoritmy existují i další články, které se věnují různým instancím tohoto problému. Jedna taková instance problému je například plánování pro pohyb skupiny quadroptér (Honig a kol. 2018). Zajímavá definice problému je také plánování pro různé konvoje agentů (Thomas a kol. 2015). Konvoj je množina jednobodových agentů (samotní agenti nemají geometrický tvar), nicméně pro tuto množinu platí omezení, že agenti se musí přesně následovat, tedy že žádné dva konvoje se nesmí protnout (obdoba oblíbené hry snake). Pro takovéto konvoje se pak snažíme najít trasy.

Závěr a další práce

Na téma MAPF pro velké agenty existuje několik přístupů. Podle jejich experimentálních výsledků je patrné, že různé přístupy jsou efektivnější pro různé situace, tedy neexistuje algoritmus, který by všechny ostatní překonával ve všech případech. Navíc, jelikož jde o NP-těžký problem (Yu a LaValle 2013), velkou překážkou je zde výkon, který nás může velmi omezovat, pokud vyžadujeme výsledky v reálném čase. Z těchto důvodů je tedy důležité dobře analyzovat situaci, kterou chceme řešit a podle ní se rozhodnout pro konkrétní algoritmus, případně zvážit, zda potřebujeme zcela optimální řešení, nebo si vystačíme s dobrou aproximací.

Mezi zmíněnými algoritmy však nejsou zastoupeny všechny metody pro řešení MAPF. Jedna taková metoda, která by mohla stát za podrobnější průzkum, je použití řešení CSP (problému splňování podmínek). Tato metoda by podle mně mohla být velice snadno adaptovatelná pro velké agenty, jelikož je přímo založená na podmínkách, obdobně jako CBS. Konkrétně jde o podmínky, pomocí nichž se filtrují domény proměnných, pomocí kterých získáváme cesty pro agenty (doména např. může obsahovat vrcholy, do nichž agent může přejít). Přidávání takových podmínek by mohlo být inspirováno např. symetrickým přístupem ze CBS pro velké agenty.

Mimo návrhu nového algoritmu je také důležité zlepšovat výkon. Stran výkonu je zajímavý např. článek o

použití rychlých náhodných restartů (Cohen a kol. 2018) pro MAPF problémy. Tento článek experimentálně ukazuje jejich příznivý vliv na výkon u algoritmů založených na CBS a A*. Je postaven na snaze rozbít velký celistvý problém hledání cest na vícero menších instancí, pomocí kterých by bylo možné složit celkové řešení. Myslím si, že tento koncept by mohl být snadno rozšiřitelný i pro velké agenty a tudíž by to mohl být další možný směr výzkumu toho, jak zlepšit výkon.

Reference

Li, J.; Surynek, P.; Felner, A.; Ma, H.; Kumar, T. K. S.; and Koenig, S. 2019. Multi-Agent Path Finding for Large Agents. AAAI conference on Artificial Intelligence.

Sharon, G.; Stern, R.; Felner, A.; and Sturtevant, N. R. 2015. Conflict-based search for optimal multi-agent pathfinding. Artificial Intelligence 219:40-66.

Sharon, G.; Stern, R.; Felner, A.; and Sturtevant, N. 2012. Meta-Agent Conflict-Based Search For Optimal Multi-Agent Path Finding. AAAI conference on Artificial Intelligence.

Stern, R.; Sturtevant, N. R.; Felner, A.; Koenig, S.; Ma, H.; Walker, T. T.; Li, J.; Atzmon, D.; Cohen, L.; Kumar, T. K. S.; Boyarski, E.; and Barták, R. 2019. Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks. AAAI conference on Artificial Intelligence.

Walker, T. T.; Sturtevand, N. R.; and Felner, A. 2018. Extended Increasing Cost Tree Search for Non-Unit Cost Domains. AAAI conference on Artificial Intelligence.

Surynek, P. 2019. Multi-Agent Path Finding with Continuous Time and Geometric Agents Viewed through Satisfiability Modulo Theories (SMT). AAAI conference on Artificial Intelligence.

Yakovlev, K.; and Andreychuk, A. 2017. Any-Angle Pathfinding for Multiple Agents Based on SIPP Algorithm. AAAI conference on Artificial Intelligence.

W. Hönig, J. A. Preiss, T. K. S. Kumar, G. S. Sukhatme and N. Ayanian, "Trajectory Planning for Quadrotor Swarms," in IEEE Transactions on Robotics, vol. 34, no. 4, pp. 856-869, Aug. 2018.

Thomas, S.; Deodhare, D.; and Murty, M. N. 2015. Extended conflict-based search for the convoy movement problem. IEEE Intelligent Systems 30(6):60–70.

Cohen, L.; Wagner, G.; Kumar, T. K. S.; Choset, H.; and Koenig, S. 2017. Rapid Randomized Restarts for Multi-Agent Path Finding Solvers. ArXiv eprints

Yu, J., and LaValle, S. M. 2013. Structure and intractability of optimal multi-robot path planning on graphs. In AAAI, 1444–1449.