MAPF / MAPD - irodalomkutatás összefoglaló

Hegyi Péter
immsrb@inf.elte.hu

Kiss Alex
fwco0f@inf.elte.hu

Michelisz Máté
ljzr1u@inf.elte.hu

Miskolczi Miklós
psbdho@inf.elte.hu

2020 október

Kivonat

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

1. Bevezetés

A klasszikus MAPF-probléma esetében adott egy $G = (V, E)$ gráf, amely az ágensek által bejárható területet, például egy raktárt reprezentál. Ezen gráf két csúcsa között pontosan akkor megy él, ha a két csúcs a raktárban szomszédos cella és egyik sem tartalmaz akadályokat. Adottak továbbá az $s\colon [1, \ldots, k] \to V$ és $t\colon [1, \ldots, k] \to V$ függvények, melyek a k ágens mindegyikéhez egy-egy kezdő- és végpontot rendelnek. Az ágensek egy diszkrét időegység alatt egy műveletet képesek végrehajtani; átlépnek egy szomszédos cellába, vagy várakoznak. Egy műveletet akkor tekintünk konfliktus-mentesnek, ha nincs két ágens, amelyek ugyanazon cellában tartózkodnak egy adott időegységen belül. A cél az, hogy minden k_i ágens számára olyan konfliktus-mentes útvonalakat találjunk s_i és t_i között, amelyek valamilyen szempont, például az útvonalak összesített hossza szerint optimálisak. [4]

A klasszikus MAPF-problémának számos kiterjesztése létezik, amelyekben az ágensek száma dinamikusan változhat, vagy akár egy-egy ágenshez több feladatot is rendelhetünk.

Anonymous MAPF A cél MAPF-probléma ezen változatában is a konfliktus-mentes útvonalak biztosítása az összes ágens számára, viszont lényegtelen, hogy melyik ágenshez rendeljük hozzá az egyes célpontokat.

Combined target-assignment and path-finding (TAPF) A TAPF-probléma az eredeti MAPF-probléma egy általánosítása, amelyben az ágenseket csoportokként kezeljük és ezekhez a csoportokhoz rendeljük hozzá a feladatokat. [3]

Online / Lifelong MAPF Az online változat esetében MAPF-problémák sorozatát kell megoldanunk, azaz az ágensekhez több célpontot rendelünk, melyeket a konfliktus-mentes útvonalak mentén sorozatban kell érinteniük.

A Multi-Agent Pickup and Delivery (MAPD) problémák,például a **raktárrendszer modell** esetében a feladatok, avagy a termékek kiszállítása, csak a futás során, ismeretlen időben derülnek ki. Egy MAPD-problémát akkor tekintünk megoldottnak, ha az összes feladatot megoldottuk azok beszúrását követő T időn belül. [2]

Az online probléma egy másik változatában nem a feladatok, hanem az ágensek száma változik a futás

során. A problémát a közúti csomópontok ihlették, amelybe ismeretlen időközönként újabb ágensek lépnek be és kívánnak áthaladni. Ebben az esetben mint megoldás, például a raktárrendszerektől eltérően nem konfliktus-mentes útvonalak halmazát keressük, hanem egy szempontot figyelembe véve minimalizálni próbáljuk azt, például a csomópont áteresztőképessége szerint. [1]

Természetesen tekinthetjük a probléma egy hibrid változatát is, amelyben az ágensek és a feladatok száma is dinamikusan változik. [5]

2. Létező megoldások

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

2.1. Centralizált algoritmusok

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

2.2. Decentralizált algoritmusok

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

2.2.1. Prioritásos Tervezés

A prioritásos tervezés algoritmusa során minden ágens kap egy prioritást, majd a prioritás alapján sorban minden ágensnek készítünk egy tervet.[6] Az tervezés során a már útvonallal rendelkező ágenseket dinamikus akadálynak tekintjük, amivel a következő robotoknak tervezniük kell. Ha végeztük, minden robotnak egy ütközésmentes útja lesz, amelyet n tervezési ciklus alatt generált le az algoritmus. Az egysze-

rű prioritásos tervezés még centralizált rendszerekre adott megoldást.

Elosztott Prioritásos Tervezés: Egyszerű decentralizált változata az előbb ismertetett *prioritásos tervezés* algoritmusának. Az egyetlen további követelmény, hogy az ágensek meg tudják állapítani a statikus prioritásukat. A tervezés részt több iterációra osztjuk, amely során minden ágens egyszerre és függetlenül készíti el a tervét, a többi ágenstől kapott korábbi tervek alapján az előző iterációból. Minden iteráció során végén elküldi az aktuális tervét és prioritását az összes többi ágensnek, hogy azok a következő tervezési szakaszban felhasználják.[6]

Csökkentett Elosztott Prioritásos Tervezés: Az elosztott prioritásos tervezés algoritmusán javít, méghozzá azzal, hogy jelentősen csökkenti a kommunikáció során szükséges üzenetek számát. Az algoritmus során az ágenseknek statikus prioritása lesz minden egyes iteráció során. Az első iteráció a megszokott módon fog lefutni, de utána az egyes ágensek csak azoknak fognak további üzeneteket küldeni, melyeknek alacsonyabb a prioritása, mint az övék. Ezen felül, ha egy ágens nem változtatja a tervét egy iteráció során, akkor semmilyen üzenetet nem kell küldenie, hiszen az aktuális terv megtalálható már a többi ágensnél.[6]

Ritka Elosztott Prioritásos Tervezés: Lehetséges ennél jobban is csökkenteni a szükséges kommunikációt az ágensek között decentralizált megoldásnál. Ehhez egy statisztikai jelenséget használ ki az algoritmus, az úgynevezett *„születésnapi paradoxont”*[1]. Mivel kooperatív ágensekkel dolgozunk, ezért egy ágens ütközéseinek felfedését és feloldását a többi ágensre is rábízhatjuk. Ha minden robot elküldi a tervezett útvonalát a többi ágens véletlenszerűen választott részhalmazának, akkor minden ágens egy véletlenszerű halmazát kapja meg az összes útnak. Ezekben kereshetünk ütközést. A valószínűsége, hogy találunk, nagyon gyorsan nő ahogy növeljük azt a számot, ahány ágensnek kiküldik az egyes ágensek a tervezett útjukat.[6]

Eredmények a Prioritásos Tervezés változatai között: Különböző tesztek során a különböző prioritásos tervezésen alapuló algoritmusok hasonlóan hatékony megoldást készítenek, és érdekes módon az ágensek számának növelésével az algoritmusok egyre rövidebb relatív utat készítettek, tehát a tervezők egyre hatékonyabb megoldást találtak.

A különbség elsősorban a tervezési időben mutatkozott meg. Az előzetes elvárásokkal ellentétben az elosztott prioritásos tervezések általánosságban lassabban dolgoztak, mint a centralizált verziók. Érdekes eredmény az is, hogy minél kevesebb ágensnek kellett együtt dolgoznia, annál jobban teljesítettek a decentralizált algoritmusok. Kevés ágens során egyes esetekben még gyorsabbak is voltak mint a centralizált algoritmus, ami alapján kijelenthető, hogy a skálázhatóság egy nagy probléma a decentralizált prioritásos tervezés típusú algoritmusoknál.[6]

2.2.2. Local Repair A*

A Local Repair A* (LRA*) algoritmus család nem egy mai megközelítése a problémának. Az elve nagyon egyszerű.

Minden ágens függetlenül a többitől, a klasszikus A* algoritmus segítségével keres magának útvonalat, ami alapján halad majd előre. Utóbbit addig teszik, míg nem észlelnek feltéten ütközést. Ekkor újratervezik a maradék útvonalukat az említett módszerrel, viszont módosított heurisztikával. Minden ágens rendelkezik úgynevezett *nyugtalansági értékkel*, ami a konfliktusok esetén megnövelődik egyel. Ez az érték fogja meghatározni, hogy amikor újra kell számolni az ágens maradék útvonalát, milyen mennyiségű *véletlenszerű zajt* adjon hozzá az A* heurisztikájához. Az ideológiája ennek, hogy minél több konfliktus keletkezik, az ágensek annál véletlenszerűbben oldják fel a konfliktusokat. Ezáltal újabb útvonalakat próbálnak meg, csökkentve a további ütközés esélyeit.

[1] A születésnapi paradoxon lényege, hogy véletlenszerűen választott emberek között annak a valószínűsége, hogy két ember ugyanazon a napon született jelentősen nő, minél több embert választunk.

Az LRA* elfogadható, megfelelő eredményt produkál egyszerűbb hálózaton, illetve kis ágens szám esetén. Bonyolultabbakra viszont nem érdemes használni, nem skálázódik megfelelően.

2.2.3. Kooperatív útvonalkeresés decentralizált megközelítéssel

Kooperatív algoritmus esetén az ágensek közös döntés alapján oldják fel a lehetséges konfliktust. Ahhoz, hogy ez megtörténjen decentralizáltan, minden ágensnek az összes többivel szükséges kommunikálnia, ami jelentős költséggel jár. Ennek kiküszöbölésére minden ágens csak a közelükben lévőkkel kommunikálnak.

Az útvonalkeresés első fázisa a többi ágenstől független optimális útvonalkeresésben merül ki. A második, a kooperatív útvonalkeresés akkor jön elő, mikor konfliktus detektálódik a kommunikáció során. Ekkor a konfliktusban lévő ágensek a különböző célokra bevezetett algoritmusok alapján szerint feloldják közöttük a lehetséges ütközést.

A konfliktus feloldó algoritmusok az alábbiak: *várakozás, előrehaladás, kitérés, hátrálás, elfordulás. Várakozás* esetén az önkéntes ágens megvárja míg a másik elhalad. *Előrehaladás* akkor történik, míg a konfliktusban lévő ágensek egy ideig ugyanazt az utat követik. Ekkor haladnak előre egy irányba egymás után. Amikor egyik ágens elállja a másik útját, és a blokkoló körül található szabad hely, akkor alkalmazza a *kitérést.* Félreáll, hogy a másik elhaladhasson, majd folytatja útját. A *hátrálás* a jól ismert folyosó szituációban alkalmazandó, mikor az egyik ágens kihátrál belőle, hogy a másik tovább tudjon haladni. Az *elfordulás* az utóbbinak egy speciális esete, mikor egy folyosón több ágens található. Az elforduló ágens jelzi az ő hátrálását blokkoló ágensnek a konfliktust.

Az említett konfliktus feloldó algoritmusok lefedik az összes létező esetet. Ezáltal ágens nem ragad be, illetve igen jól skálázódik az ágensek számával.

2.2.4. A WSCaS módszer

A WSCaS (Walk, Stop, Count and Swap), azaz (séta, megállás, számlálás és csere) egy decentralizált algoritmus. Elméleti analízis megmutatta, hogy a módszer alkalmazásával legrosszabb esetben is O(1) távolságoptimális megoldás található a négyzetrácsos elrendezésű, szűk átjáró[2] nélküli MAPF problémára. Utóbbi feltételre a *cserélhetőség* tulajdonsága miatt van szükség, hiszen ilyenkor 2 ágens konstans lépésben helyet tud cserélni.

Az algoritmus menete nagyon hasonló az előző fejezetben taglalt kooperatív útvonalkereséshez. Az ágensek először egymástól függetlenül meghatározzák a saját útjaikat. Egy a_i ágens egy lépésénél öt lehetséges állapot fordulhat elő:

- type-0: Ha a következő cella szabad, és nincs olyan másik ágens, aki ide lépne és az $\mathbb{R}^2$-beli lexikális pozíciója kisebb. Ekkor a_i *sétál*, oda léphet.
- type-1: A következő cella szabad, de egy másik ágens is oda szeretne lépni, akinek a $\mathbb{R}^2$-beli lexikális pozíciója kisebb. Ekkor a_i-nek meg kell *állnia.*
- type-2: Két ágens pozíciót szeretne cserélni, vagy az vár egy másik ágensre, aki már megérkezett a céljához. Ez egy holtpont, amit a két ágens egy *cserével* fog megoldani.
- type-3: Az ügynök egy sorozatban található, ami nem kör. Ekkor meg kell *állnia.*
- type-4: Az ügynök egy körben található. Ez holtpont, *cserékkel* megoldható.

Egyes esetekben a type-4 kör olyan hosszú lehet, hogy az a_i ágens a helyi kommunikációjával nem fogja tudni, hogy körben van-e vagy egy sorozatban. Az eset kikerülése érdekében van a *számlálás* protokol: Minden ágens tárol egy α számot, amit a szomszédainak állandóan továbbít. Ha a_i nem blokkol másik utat, akkor 0-t küld, ha viszont blokkolja a_j utját, akkor $\alpha_j + 1$-et küld tovább. Így egy körnél a végtelenségig fog növekedni, egy sorozatnál pedig legfeljebb $|V|$-ig.

Az algoritmus költségei függetlenek az ágensek méretétől a számítás, a memória és a kommunikáció

[2]Szűk átjáró nélküli négyzetrács: a legszűkebb átjáró is megengedi, hogy 2 ágens párhuzamosan áthaladjon rajta.

komplexitására vonatkozóan. E tulajdonság következtében jól skálázhatóvá válik az ágensek számával a gyakorlatban.[7]

2.3. A saját megoldásunk

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Hivatkozások

[1] Kurt Dresner and Peter Stone. A multiagent approach to autonomous intersection management. *J. Artif. Intell. Res. (JAIR)*, 31:591–656, 01 2008.

[2] Hang Ma, Wolfgang Hönig, T. K. Satish Kumar, Nora Ayanian, and Sven Koenig. Lifelong path planning with kinematic constraints for multi-agent pickup and delivery, 2018.

[3] Hang Ma and Sven Koenig. Optimal target assignment and path finding for teams of agents, 2016.

[4] Oren Salzman and Roni Stern. Research challenges and opportunities in multi-agent path finding and multi-agent pickup and delivery problems. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '20, page 1711–1715, Richland, SC, 2020. International Foundation for Autonomous Agents and Multiagent Systems.

[5] Roni Stern, Nathan Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, T. K. Satish Kumar, Eli Boyarski, and Roman Bartak. Multi-agent pathfinding: Definitions, variants, and benchmarks, 2019.

[6] Prasanna Velagapudi, Katia Sycara, and Paul Scerri. Decentralized prioritized planning in large multirobot teams. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4603–4609. IEEE, 2010.

[7] H. Wang and M. Rubenstein. Walk, stop, count, and swap: Decentralized multi-agent path finding with theoretical guarantees. *IEEE Robotics and Automation Letters*, 5(2):1119–1126, 2020.