Vpeljava varnih mest za vseživljenjsko večrobotsko planiranje poti v interni logistiki

Andrej Zdešar[1], Matevž Bošnak[1], Rok Vrabič[2], Viktor Zaletelj[3], Gregor Klančar[1]

[1]*Univerza v Ljubljani, Fakulteta za elektrotehniko, Tržaška cesta 25, 1000 Ljubljana, Slovenija*
[2]*Univerza v Ljubljani, Fakulteta za strojništvo, Aškerčeva cesta 6, 1000 Ljubljana, Slovenija*
[3]*Epilog, Tehnološki park 22a, 1000 Ljubljana, Slovenija*
E-pošta: gregor.klancar@fe.uni-lj.si

Introduction of safe locations for lifelong multi-robot path planning in internal logistics

A lifelong multi-agent pathfinding (MAPF) algorithm for an intralogistic system is proposed. To ensure the completeness of planning, we introduce safe locations. We increase the practical applicability of the MAPF algorithm by providing different types of safe locations (virtual, serial, parallel, charging and others). We show that by appropriately choosing the types, number and locations of safe locations, as well as their selection strategies, we can significantly influence the performance of the intralogistic system.

1 Uvod

Napredne rešitve večrobotskega planiranja poti (MAPF, angl. *Multi-Agent PathFinding*) omogočajo ustrezno delovanje internih logističnih sistemov, skladišč ali proizvodnje, kjer skupina transportnih vozil (agentov) prevaža in dostavlja material med delovnimi postajami.

Zaradi računske kompleksnosti so optimalni algoritmi, ki temeljijo na različicah algoritma A* [1, 2], možni le za majhno število agentov, saj kompleksnost narašča eksponentno s številom agentov. V praksi je zato potrebno računsko zahtevnost omiliti z uvedbo suboptimalnih pristopov, ki planirajo gibanje vsakega agenta ločeno in nato rešujejo konflikte z uvedbo prometnih pravil, lokalnega večagentnega usklajevanja, pogajanja ali prioritet [1, 2, 3]. Dodatno zahtevnost vnaša vseživljenjski način delovanja, saj je potrebno zagotoviti neprekinjeno delovanje sistema ter zagotavljati kvaliteto rešitev [4, 5]. Dodatni izziv predstavljajo bodoče naloge, katerih vsebina in čas nastopa nista vnaprej znana.

V članku predstavljamo izvedbo planiranja z nadgradnjo algoritma SIPP (angl. *Safe Interval Path Planning*) [6] s prioritetami in predlaganim konceptom varnih mest. Vpeljava varnih mest izboljša delovanje sistema in zagotovi kompletnost planiranja v vseživljenjskem načinu delovanja. Varna mesta so tudi praktično uporabna kot dodatni prostori za odlaganje tovora pri kompleksnejših nalogah, parkirni prostori za neovirajoče čakanje, polnilnice za polnjenje baterij, servisna mesta in podobno.

2 Definicija problema

Okolje opišemo z grafom $\mathcal{G} = \langle E, V \rangle$, kjer so $v_j \in V$ vozlišča (križišča povezav) ter $e_i \in E$ povezave. Povezave so usmerjene in utežene s trajanjem vožnje.

Algoritem MAPF najde nabor nekonfliktnih planov $[\pi_{AGV_1}, \ldots, \pi_{AGV_k}, \ldots]$ v $\mathcal{G}$ za skupino agentov AGV_k. Vsak plan π_{AGV_k} se začne v trenutnem vozlišču $start_k \in V$ agenta k in gre preko zahtevanih vmesnih ciljev $[cilj_k(1), \ldots, cilj_k(N-1)] \in V$, če so podani, v končno ciljno vozlišče $cilj_k(N) \in V$. V nadaljevanju se osredotočamo na naloge tipa prevzemi in dostavi, kjer agent AGV_k določi načrt poti iz svoje trenutne lokacije (TM_k) do prevzemnega mesta (PM_k), kjer naloži tovor, ga odpelje do dostavnega mesta (DM_k) in se umakne v varno mesto (VM_k). Za to nalogo sestavljen plan $\pi_{AGV_k} = [\pi_1, \pi_2, \pi_3]$ vsebuje načrt poti π_1 za premik iz TM_k v PM_k, π_2 za premik iz PM_k v DM_k in π_3 za premik iz DM_k v VM_k.

Posamezen plan $\pi = (a_1, a_2, \ldots, a_n)$ vsebuje akcije a_i. Te opišejo pot s podanim hitrostnim profilom od startnega do ciljnega vozlišča z ustreznimi časovnimi trenutki. Akcijo opiše nabor $a_i = \langle t_i, e_i, loc_i \rangle$, ki definira želen čas prihoda na lokacijo loc_i znotraj povezave $e_i \in E$. Akcija a_i opisuje premik, če se lokacija glede na a_{i-1} spremeni. Akcija predstavlja čakanje na mestu, če se lokacija ne spremeni. Čakanje je možno le v vozliščih (kot v [6, 2]), tj. na koncu povezav.

3 MAPF-algoritem

Za določitev planov smo izhajali iz algoritma SIPP [6], ki smo ga nadgradili za večagentno iskanje s prioritetami, vmesnimi postajami in varnimi mesti. Algoritem SIPP je razširitev algoritma A* z varnimi intervali, ki najde optimalne plane za enega agenta v okoljih z znanimi trajektorijami dinamičnih ovir.

Varni interval definira maksimalni neprekinjen časovni interval v katerem je neko vozlišče dostopno. Posamezno vozlišče ima lahko enega ali več varnih intervalov, ki so ločeni z intervali zasedenosti. Interval zasedenosti definira časovni interval, ko to vozlišče zaseda nek agent. Unija varnih in zasedenih intervalov predstavlja celotno časovno os. SIPP pri raziskovanju grafa razlikuje stanja, ki so definirana kot skupek vozlišča in enega od njegovih varnih intervalov. V

posamezno vozlišče lahko torej vstopimo večkrat ob različnih varnih intervalih. Med delovanjem SIPP poišče možna naslednja stanja za neko obstoječe stanje. To so torej tista vozlišča, ki jih agent (preko povezave z upoštevanimi hitrostnimi omejitvami in brez trkov z drugimi agenti) lahko doseže v varnih intervalih.

Pri večrobotskem planiranju (MAPF) v SIPP vpeljemo prioritete (algoritem PSIPP). Vpeljava prioritet omogoči razklopljenost algoritma MAPF, kar bistveno zmanjša računsko kompleksnost [1, 2]. Problem lahko rešujemo za vsakega agenta ločeno, ob upoštevanju gibanja ostalih. Hkrati pa imamo možnost, da agenti opravljajo bolj ali manj pomembne naloge z različnimi prioritetami.

3.1 Vseživljenjsko delovanje

Večina znanih algoritmov se, zaradi nazornosti in poenostavitve, osredotoča na enkratno planiranje, kjer vsi agenti začno hkrati in imajo le eno dostavo. V praksi agenti po izvedeni dostavi dobijo nove naloge, ki jih morajo uskladiti z obstoječimi nalogami v izvajanju. Pri tem nove naloge večinoma niso poznane vnaprej. Govorimo o vseživljenjskem oz. neprekinjenem delovanju.

Ker v trenutku planiranja gibanja posameznega agenta prihodnje naloge in plani izvajanja teh nalog za ostale agente niso poznani, lahko pride pri izvajanju do konfliktov. Tak primer je lahko agent, ki uspešno konča svojo nalogo in je prost za prevzem naslednje naloge. Ko mu je dodeljena nova naloga (prej neznana), jo rešuje z najnižjo prioriteto (predhodne naloge v izvajanju imajo prednost). Rešitve (načrta poti) za novo nalogo algoritem lahko ne najde, saj so lahko povezave, ki vodijo do njega, že zasedene z nasprotno smerjo vožnje drugih agentov. V tem primeru bi bilo potrebno ponovno planirati poti za vse agente oz. vsaj za tiste v konfliktu in jim iterativno spreminjati in iskati ustrezne prioritete. To bi postalo računsko prezahtevno zaradi kombinatornosti problema [1]. V izogib tega v nadaljevanju predstavljamo idejo varnega mesta brez potrebnega ponovnega planiranja, ki je torej računsko nezahtevna, optimalna (pod vpeljanimi predpostavkami — prioritete) in vedno izvedljiva.

4 Vpeljava varnih mest

Varno mesto (VM) je področje v prostoru (in grafu), kamor se agent po končani nalogi lahko varno umakne, če je potrebno, in čaka na naslednjo nalogo. V VM lahko vstopajo le agenti, ki planirajo pot s ciljem v VM-ju, drugače poti ne smejo prečkati VM-ja. Osnovni koraki planiranja so:

1. Če so na voljo nove naloge in je prost vsaj en agent (je opravil dostavo in se pelje proti ali čaka v VM-ju), dodelimo agentu najnižjo prioriteto in poiščemo načrt poti za najstarejšo novo nalogo.
2. Če najdemo nov plan, prekinemo izvajanje trenutnega plana, sprostimo zasedenosti za neizveden preostanek trenutnega plana, nastavimo zasedenosti vozlišč in povezav za novi plan ter ga začnemo izvajati. Pri tem upoštevamo zasedenosti vozlišč in povezav višje prioritetnih agentov. Nadaljujemo s korakom 1.
3. Če novega plan ne najdemo, nadaljujemo trenutno pot proti VM-ju ali čakamo v VM-ju. Po določenem času nadaljujemo s korakom 1.

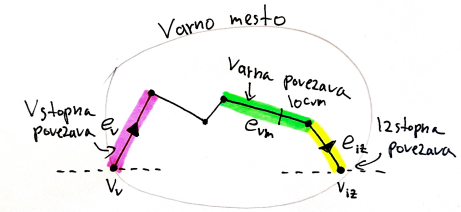
4.1 Lastnosti PSIPP-algoritma z varnim mestom

Predstavljen algoritem je kompleten, blizu optimalen, računsko učinkovit in omogoča vseživljenjsko delovanje (PSIPPL). Pri vseživljenjskem planiranju je potrebno za izvedljivost (kompletnost) algoritma planiranja vsaki nalogi dodati še zadnji delni plan π_{N+1}. Ta pripelje agenta do VM-ja, kjer vedno lahko varno čaka dokler ne dobi naslednje naloge. V času izvajanja π_{N+1} je agent prost in se zato lahko izvajanje tega zadnjega plana kadarkoli prekine, če je možno najti nov plan do želene lokacije. V kolikor to ni možno, pa se agent lahko vedno umakne v VM in od tam nadaljuje kasneje.

Algoritem PSIPPL je optimalen, če prioritete upoštevamo (manj prioritetni agenti ne smejo ovirati višje prioritetnih), drugače pa je skoraj optimalen. Je računsko zelo učinkovit, saj njegova kompleksnost narašča linearno s številom agentov. Kompleksnost uveljavljenih optimalnih MAPF-algoritmov, kot sta CCBS in A*, ki ne podpirata vseživljenjskega načina, narašča eksponentno s številom agentov [2, 1].

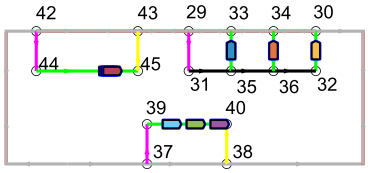
Kot motivacijo za razvoj algoritma PSIPPL podajmo primerjavo z uveljavljenim algoritmom CCBS [2]. Običajno je čas za dostavo vseh agentov pri CCBS le nekoliko krajši kot pri PSIPPL, medtem ko je število iteracij algoritma CCBS večje za več (3-4) velikostnih razredov. V primeru vseživljenjskega delovanja in nalog tipa prevzemi-dostavi, najde PSIPPL boljše rešitve, saj CCBS ne omogoča vseživljenjskega delovanja in ga je potrebno zaganjati paketno. Zaradi kompleksnosti problema in časovne omejitve za izračun, CCBS najde rešitev le v določenih primerih, PSIPPL pa vedno.

4.2 Izvedba varnega mesta



Slika 1: Splošna izvedba varnega mesta

Splošno varno mesto VM (slika 1) predstavlja varna povezava e_{vm} s predpisano lokacijo loc_{vm}, kjer lahko agent neovirajoče čaka. V VM lahko vstopimo le preko usmerjene vstopne povezave e_v, ki se odcepi od preostalega grafa v vozlišču v_v. VM pa lahko zapustimo le preko usmerjene izstopne povezave e_{iz}, ki se priklopi na preostali graf v vozlišču v_{iz}. Poleg povezav e_v in e_{iz} pripadajo VM-ju tudi vse povezave med koncem e_v in začetkom e_{iz}, med katerimi se mora nahajati tudi e_{vm}. Vstopnih povezav e_v je lahko več, vendar mora biti z vseh dosegljiva povezava e_{vm}. Prav tako je lahko več tudi izstopnih povezav, ki morajo biti vse dosegljive s povezave



Slika 2: Primeri izvedb varnih mest

e_{vm}. Vstop v VM lahko onemogočimo tako, da odstranimo povezavo e_v oz. prepovemo njeno uporabo. Če odstranimo povezavo e_{iz}, onemogočimo izstop iz VM-ja. Če je vstopnih ali izstopnih povezav več, lahko preprečimo vstop oz. izstop le po določenih povezavah.

VM lahko sprejme največ $C \geq 0$ agentov, kar predstavlja njegovo kapaciteto. VM-ju predpišemo tudi enega ali več tipov (npr. polnilnica, servisna postaja, počivališče itd.), ki podajajo kaj VM omogoča (npr. polnilnica omogoča polnjenje baterije), in tudi dodatne lastnosti, ki lahko omejijo kakšni agenti lahko vanj vstopijo. Prav tako kot za katerokoli povezavo v grafu, lahko tudi za povezave v VM-ju predpišemo, ali se na njih preverjajo trki ali ne, kar omogoča izvedbo virtualnih VM-jev.

Nekaj primerov izvedb VM-jev prikazuje slika 2. Primer virtualnega VM-ja, ki ima neskončno kapaciteto $C = \infty$ in nima preverjanja trkov na e_{vm}, je izveden z vstopno povezavo $e_v = (42, 44)$, varno povezavo $e_{vm} = (44, 45)$ z $loc_{vm} = 0{,}8$ in izstopno povezavo $e_{iz} = (45, 43)$. Serijski VM po sistemu *prvi noter – prvi ven* s kapaciteto $C = 3$ je izveden z $e_v = (37, 39)$, $e_{vm} = (39, 40)$ in $e_{iz} = (40, 38)$. Vzporedni VM s poljubnim vrstnim redom je izveden s tremi splošnimi VM-ji s kapaciteto $C = 1$, kjer so varne povezave na vzporednih povezavah $e_{vm} = (35, 33)$, $e_{vm} = (36, 34)$ in $e_{vm} = (32, 30)$. Vstopna povezava $e_v = (29, 31)$ je skupna, izstopne povezave pa so skozi vozlišča $v_{iz} \in \{33, 34, 30\}$. V tem primeru so izstopne povezave kar varne povezave ($e_{iz} = e_{vm}$). Možne so tudi kombinacije virtualnih, serijskih in vzporednih VM-jev. Predstavljena splošna definicija VM-ja torej omogoča skoraj poljubne izvedbe VM-jev.

4.3 Varno mesto s polnilnico

Varno mesto s polnilnico (VMP) je poseben tip VM-ja, ki omogoča polnjenje baterij. Vsak agent ima baterijo s kapaciteto C_B. Ko je agent v VMP-ju, se nivo njegove baterije $c(t)$ polni s funkcijo polnjenja

$$c(t + \Delta t) = \min\{c(t) + \Delta\Phi^+(c(t), \Delta t), C_B\} \quad (1)$$

kjer $\Delta\Phi^+(c(t), \Delta t) \geq 0$ določa profil polnjenja. Ko agent ni v VMP-ju, se nivo baterije $c(t)$ prazni s funkcijo praznjenja

$$c(t+\Delta t) = \max\{c(t) - \Delta\Phi^-(c(t), \Delta t, \Delta d(t)), 0\} \quad (2)$$

kjer $\Delta\Phi^-(c(t), \Delta t, \Delta d(t)) \geq 0$ določa profil praznjenja. Primer enostavnega profila polnjenja, ki smo ga uporabili v tem delu pri simulacijah, je odvisen le od časa polnjenja $\Delta\Phi^+ \propto \Delta t$, profil praznjenja pa le od prepotovane razdalje $\Delta\Phi^- \propto \Delta d(t)$.

Pri polnjenju predpišemo minimalni zgornji nivo baterije C_M, ki ga mora agent doseči, preden se polnjenje lahko zaključi in agent nadaljuje z izvajanjem nalog. Ko nivo baterije pade pod določeno minimalno spodnjo mejo C_m, mora agent oditi v VMP preden začne izvajati naslednjo nalogo. Če nivo baterije pade pod kritično spodnjo mejo C_k, agent obstoji — potreben je poseg operaterja.

Na podoben način kot varna mesta s polnilnico lahko definiramo tudi servisna varna mesta, kamor pošljemo agenta v primeru manjših okvar ali v rednih intervalih za izvedbo vzdrževalnih del. Definiramo pa lahko tudi druge tipe varnih mest.

4.4 Strategija izbiranja varnih mest

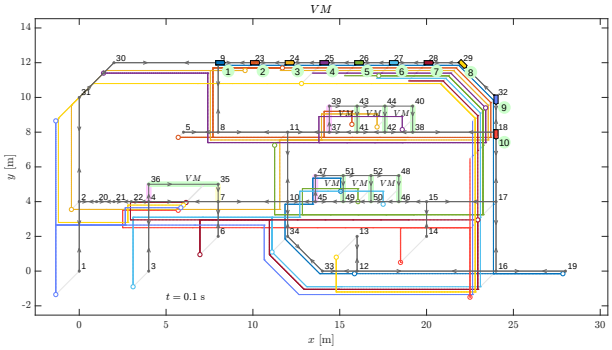
V kolikor imamo na voljo več VM-jev, moramo (pred planiranjem poti) agentu dodeliti varno mesto. Možno je izvesti mnogo strategij, ki so odvisne od oddaljenosti vstopnega vozlišča v_v in/ali izstopnega vozlišča v_{iz} določenega VM-ja od agenta, prevzemnih in/ali dostavnih mest, tipa in zasedenosti VM-ja, stanja baterije itd. V nadaljevanju je predstavljena strategija dodeljevanja VM-jev, ki je uporabljena v tem delu.

Če ima agent nivo baterije nad spodnjo mejo ($c(t) \geq C_m$), poiščemo prosti VM, ki je najbližje dostavnemu mestu njegove naloge in po možnosti ni VMP. Ko agent prispe do dostavnega mesta, se mu že lahko dodeli nova naloga, še preden prispe v VM, čeprav je bilo izvedeno planiranje poti do VM-ja.

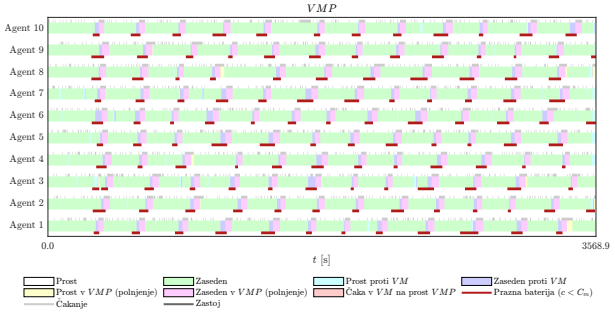
Če pa je nivo baterije pod spodnjo mejo ($c(t) < C_m$), mora agent, preden lahko nadaljuje z izvajanjem nalog, v VMP. Poiščemo najbližji prost VMP. Če le-ta ne obstaja, poiščemo najbližji prost VM, kjer agent počaka, da se sprosti VMP, kamor nato načrta pot. Če se že nahajamo v VMP-ju ali v VM-ju in ni nobenega prostega VMP-ja, ostanemo na trenutni lokaciji. Šele ko agent doseže VMP in se nivo njegove baterije napolni nad minimalni zgornji nivo ($c(t) > C_M$), lahko agent nadaljuje z opravljanjem nalog.

5 Simulacijski eksperimenti

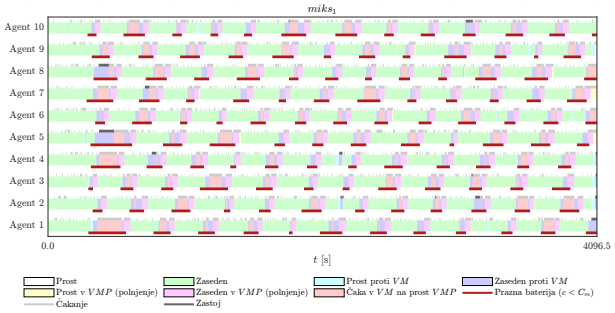
Prikazujemo primer vseživljenjskega delovanja z nalogami tipa prevzemi in dostavi, kjer 10 agentov izvaja 520 nalog (slika 3). Algoritem je splošen in deluje tudi za še večja okolja s še več agenti.



Slika 3: Začetni načrti 10 agentov, ki opravljajo 520 nalog tipa prevzemi in dostavi v vseživljenjskem načinu (konfiguracija varnih mest iz primera VM)



Slika 4: Časovni potek stanj vseh agentov iz primera VMP



Slika 5: Časovni potek stanj vseh agentov iz primera $miks_1$

5.1 Virtualna in realna varna mesta

Preverili smo delovanje ob predpostavki, da agenti ne rabijo polnjenja baterij. Podobne rezultate dobimo, če uporabimo en virtualni VM na $e_{vm} = (36, 35)$ (tabela 1, vrstica 1) ali 10 realnih (praktično uporabnih) VM-jev, tj. serijski VM s $C = 6$ na $e_{vm} = (36, 35)$ in 6 paralelnih VM-jev s $C = 1$ (tabela 1, vrstica 2). Čas za izvedbo vseh nalog t_{fin} je nekoliko krajši pri realnih VM-jih, saj so bolj razpršeni po prostoru in so poti tako krajše zaradi manj izogibanj. Podobno so malo krajši skupni čas vožnje $\sum t$, skupna razdalja $\sum d$, skupen čas voženj v VM $\sum t_{VM}$, skupen čas za koordinacijo $\sum t_{CO}$ ter število iteracij N_{it} algoritma PSIPPL.

5.2 Varna mesta in polnilnice

Rezultati delovanja, če imajo vsa varna mesta možnost polnjenja (6 v serijskem na $e_{vm} = (36, 35)$ in 6 v vzporednih), so zbrani v 3. vrstici tabele 1. Opazimo, da se t_{fin}, $\sum t$ in $\sum d$ malo podaljšajo, saj se agenti po določenem času delovanja dodatno vozijo v VMP-je za potrebe polnjenja. Slednje se opazi tudi na porabljenem času $\sum t_{VM}$ za vožnjo v VMP-je. Vpliv polnjenja je razviden tudi iz časovnega poteka stanj vseh agentov na sliki 4.

V primerih $miks_1$ in $miks_2$ (4. in 5. vrstica tabele 1) imamo kombinacijo osnovnih VM-jev in polnilnih VMP-jev. V primeru $miks_1$ imamo le dve polnilni mesti (VMP-ja na $e_{vm} = (44, 42)$ in $e_{vm} = (52, 50)$), v primeru $miks_2$ pa imamo še dodatna dva VMP-ja na $e_{vm} = (40, 38)$ in $e_{vm} = (48, 46)$. V primeru $miks_2$ glede na $miks_1$ je t_{fin} približno $10\,\%$ krajši, v skupnem času vožnje $\sum t$ in opravljeni razdalji $\sum d$ pa ni večjih razlik. V $miks_1$ je manj VMP-jev, zato morajo agenti čakati v ostalih VM-jih, ko rabijo polnjenje, saj ni dovolj prostih VMP-jev (glej sliko 5). Zato se čas za izvedbo nalog podaljša. Občutno se podaljša tudi čas $\sum t_{VM}$ porabljen za vožnjo v VM-je. Čas potreben za koordinacijo $\sum t_{CO}$ pa je pri $miks_2$ večji, saj se agenti vozijo v več različnih VMP-jev. S tem zasedejo vozlišča in morajo zato agenti z nižjo prioriteto voziti drugje ali čakati.

Tabela 1: Rezultati delovanja z različnimi varnimi mesti: le virtualna ($virt$), le navadna serijska in paralelna (VM), vsa polnilna (VMP), kombinacije navadnih in polnilnih ($miks$)

Primer	t_{fin} [s]	$\sum t$ [s]	$\sum d$ [m]	$\sum t_{VM}$ [s]	$\sum t_{CO}$ [s]	N_{it} [1]
$virt$	3 245	31 923	25 704	652	4 158	2 949
VM	3 075	30 273	24 864	454	3 285	2 532
VMP	3 626	30 928	26 144	7 292	2 595	2 662
$miks_1$	4 096	31 664	26 888	13 243	2 540	2 659
$miks_2$	3 794	32 111	27 238	9 463	2 759	2 649
$miks_3$	3 802	31 714	26 751	9 234	2 826	2 637

Primer $miks_3$ ima glede na $miks_2$ vse VMP-je s $C = 2$ (serijski tip, polni se le prvi agent na e_{vm}), kar v tem primeru nima večjega vpliva na delovanje.

6 Zaključek

Predstavljen je algoritem vseživljenjskega planiranja poti v internem logističnem sistemu. Za zagotovitev kompletnosti planiranja smo vpeljali varna mesta ter jim z uvedbo različnih tipov (virtualni, serijski, vzporedni, polnilni) povečali praktično uporabnost. Z ustrezno izbiro tipov VM-jev ter VMP-jev, njihovega števila in lokacij ter strategij izbiranja VM-jev lahko znatno in dodatno (poleg izbire algoritma MAPF) vplivamo na optimalnost delovanja sistema. Strategije postavljanja in izbiranja VM-jev so še predmet nadaljnjih raziskav s strojnim učenjem.

Zahvala

Avtorji se zahvaljujemo za finančno podporo Agenciji Republike Slovenije za raziskovalno dejavnost in podjetju Epilog d.o.o. (št. L2-3168 in P2-0219).

Literatura

[1] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 219, pp. 40–66, 2015.

[2] A. Andreychuk, K. Yakovlev, P. Surynek, D. Atzmon, and R. Stern, "Multi-agent pathfinding with continuous time," *Artificial Intelligence*, vol. 305, p. 103662, 2022.

[3] T. Žužek, R. Vrabič, A. Zdešar, G. Škulj, I. Banfi, M. Bošnak, V. Zaletelj, and G. Klančar, "Simulation-based approach for automatic roadmap design in multi-agv systems," *IEEE Transactions on Automation Science and Engineering*, pp. 1–12, 2023.

[4] V. Digani, L. Sabattini, C. Secchi, and C. Fantuzzi, "Ensemble coordination approach in multi-agv systems applied to industrial warehouses," *IEEE Transactions on Automation Science and Engineering*, vol. 12, pp. 922–934, 7 2015.

[5] A. Zdešar, T. Žužek, M. Bošnak, V. Zaletelj, R. Vrabič, and G. Klančar, "Vidiki koordiniranega načrtovanja poti na podlagi prioritet in časovnih oken za optimizacijo logistike avtomatsko vodenih vozičkov," in *zbornik trinajste konference Avtomatizacija v industriji in gospodarstvu*, pp. 1–9, AIG'23: 13.-14. april 2023, Maribor, 2023.

[6] M. Phillips and M. Likhachev, "Sipp: Safe interval path planning for dynamic environments," in *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 5628 – 5635, 06 2011.