

Učinkovito večagentno načrtovanje poti za logistični sistem

Andrej Zdešar¹, Matevž Bošnjak¹, Viktor Zaletelj², Rok Vrabič³ in Gregor Klančar¹

¹Univerza v Ljubljani, Fakulteta za elektrotehniko, Tržaška c. 25, 1000 Ljubljana, Slovenija

²Epilog, Tehnološki park 22a, 1000 Ljubljana, Slovenija

³Univerza v Ljubljani, Fakulteta za strojništvo, Aškerčeva c. 6, 1000 Ljubljana, Slovenija

andrej.zdesar@fe.uni-lj.si, matevz.bosnak@fe.uni-lj.si, viktor.zaletelj@epilog.net,
rok.vrabic@fs.uni-lj.si, gregor.klancar@fe.uni-lj.si

Efficient Multi-Agent Path Planning for Logistics System

The paper presents an approach that enables efficient multi-agent path planning. The environment is described by a graph that represents a network of roads and crossroads for vehicles of different shapes and velocities. The multi-agent path finding approach is based on a safe interval path planning algorithm that includes implicit collision checking in a computationally efficient way. Priorities and safe locations that have been introduced into the algorithm enable lifelong operation. The algorithm is complete and it produces near-optimal plans, which are continuous in time and space, and it is computationally efficient. With additional mechanisms such as charging stations, the proposed approach is practically applicable in a wide variety of realistic industrial scenarios for material handling and delivery tasks within a logistics system. In this paper, the computational efficiency of the algorithm is analysed with respect to the number of vehicles and the size of the map. The results show that the presented approach is computationally efficient even in the case of a large number of vehicles and large maps.

Kratek pregled prispevka

V članku je predstavljen pristop, ki omogoča učinkovito načrtovanje poti za več agentov. Okolje je opisano z grafom, ki predstavlja omrežje cest in križišč za vozila različnih oblik in hitrosti. Pristop planiranja temelji na algoritmu za načrtovanje poti z varnimi intervali, ki na računsko učinkovit način omogoča implicitno preverjanje trkov. V algoritem so bile vpeljane prioritete in varna mesta, kar omogoča vseživljenjsko delovanje. Algoritem je kompleten in daje skoraj optimalne poti, ki so časovno in prostorsko zvezne, ter je računsko učinkovit. Z dodatnimi mehanizmi, kot so polnilna mesta, je predlagani pristop praktično uporaben v raznovrstnih realnih industrijskih scenarijih pri nalogah razvažanja in dostavljanja materiala znotraj logističnega sistema. V članku je analizirana računsko učinkovitost algoritma glede na število vozil in velikost zemljevida. Rezultati kažejo, da je predstavljen pristop računsko učinkovit tudi pri velikem številu vozil in v primeru velikih zemljevidov.

1 Uvod

Sodobni industrijski proizvodni obrati in skladišča vse pogosteje uporabljajo avtomatsko vodene vozičke za razvoz materiala med delovnimi postajami. Za ustrezno in nemoteno delovanje logističnega sistema, ki upravlja s skupino vozil (agentov), so potrebne napredne rešitve večagentnega planiranja poti, ki zagotavljajo (čim bolj) optimalne poti brez trkov in zastojev.

Računska zahtevnost optimalnih algoritmov za večagentno načrtovanje poti, ki temeljijo na različicah algoritma A^* [1, 2], narašča eksponentno s številom agentov, zato so le-ti primerni le za majhne skupine agentov. Računska zahtevnost običajno omilimo z uvedbo suboptimalnih pristopov, ki planirajo pot vsakega agenta ločeno, in nato rešujejo morebitne konflikte z uvedbo prometnih pravil, lokalnega večagentnega usklajevanja, pogajanja ali prioritizacije [1–3]. Dodaten izziv predstavlja t. i. vseživljenjsko planiranje, kjer bodoče naloge niso znane vnaprej, in moramo zagotoviti neprekinjeno delovanje, tako da ne pride do neresljivih situacij tudi v prihodnosti [4, 5].

V članku je predstavljen računsko učinkovit algoritem za večagentno načrtovanje planov, ki temelji na algoritmu planiranja poti z varnimi intervali [6]. Algoritem je prirejen za okolje opisano z grafom, ki predstavlja omrežje križišč in cest. Planiranja omogoča iskanje časovno in prostorsko zveznih planov, ki so blizu optimalnim rešitvam, Z vpeljavo prioritizacije in varnih mest smo dosegli, da je planiranje kompletno tudi v vseživljenjskem načinu delovanja. Vpeljava varna mesta lahko v praksi funkcionirajo kot polnilnice za polnjenje baterij, servisne postaje, mesta za umik idr.

2 Vseživljenjsko večagentno planiranje poti

V nadaljevanju je na kratko predstavljeno prioritizirano planiranje poti z varnimi intervali za okolje, ki je opisano z grafom. Algoritem planiranja je nato nadgrajen še z vmesnimi cilji in z varnimi mesti, da dosežemo vseživljenjsko delovanje. Predlagani pristop je računsko učinkovit in je praktično uporaben za različne naloge planiranja, kot so zaporedne dostave, prevzemi in dostave v pred-

pisanih vrstnih redih [7] idr. Omogočena je tudi izvedba planiranja v polnilnice [8], ko agentom zmanjka energije za nadaljnje naloge.

2.1 Opis okolja in problema

Okolje opišemo z grafom $G = \langle E, V \rangle$, kjer so V vozlišča (križišča povezav) in E povezave (ceste med vozlišči). Povezave so usmerjene in utežene s trajanjem vožnje. Za vsako usmerjeno povezavo $e \in E$ predpišemo neposredno dosegljive povezave iz končnega vozlišča $v_k \in V$ povezave e oz. povezave, ki vodijo do usmerjene povezave e čez njeno začetno vozlišče $v_z \in V$. V splošnem tako lahko niso možni prehodi med vsemi povezavami, ki se stikajo v določenem vozlišču. Agenti se lahko gibljejo le v predpisanih smereh povezav.

Predpostavljamo še, da se trki med agenti lahko zgodijo le v okolici vozlišč, sicer pa ne more priti do trka med vozili na različnih povezavah, razen če gre za par usmerjenih povezav, ki predstavljata dvosmerno cesto. V primeru sekajočih-se povezav lahko graf nekoliko preoblikujemo z vpeljavo dodatnih vozlišč na sečiščih povezav. Graf opremimo tudi z informacijo o t. i. bližnjih vozliščih, ki so tako skupaj, da lahko pride do trkov med agenti, ki se nahajajo v okolici kateregakoli izmed bližnjih vozlišč.

Večagentno planiranje išče nabor nekonfliktnih planov $\{\pi_1, \dots, \pi_k, \dots\}$ v G za skupino agentov. Vsak plan π_k se začne v vozlišču $v_{start,k} \in V$, kjer se nahaja agent k , in gre čez zahtevane vmesne cilje ($v_{cilj,k}(1), v_{cilj,k}(2), \dots, v_{cilj,k}(N-1)$), če so le-ti podani, v končno ciljno vozlišče $v_{cilj,k}(N)$, $v_{cilj,k} \in V$.

Posamezen plan $\pi = (a_1, a_2, \dots, a_n)$ vsebuje zaporedje akcij a_i . Te opišejo pot s podanim hitrostnim profilom od startnega do ciljnega vozlišča z zelenimi lokacijami v ustreznih časovnih trenutkih. Akcijo opišemo torej z $a_i = \langle t_i, e_i, l_i \rangle$, kjer t_i podaja zelen čas prihoda na lokacijo, ki je podana s položajem l_i znotraj povezave $e_i \in E$. Akcija a_i opisuje premik, če se lokacija glede na a_{i-1} spremeni, tj. $e_i \neq e_{i-1} \vee l_i \neq l_{i-1}$. Sicer akcija a_i predstavlja čakanje za čas $t_i - t_{i-1}$ na mestu $l_i = l_{i-1}$ znotraj povezave $e_i = e_{i-1}$.

2.2 Prioritetno planiranje z varnimi intervali

Za določitev planov za več agentov smo izhajali iz algoritma za *planiranje poti z varnimi intervali* (PPVI) [6]. 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 vozlišče zaseda nek agent. Unija varnih in zasedenih intervalov predstavlja celotno časovno os. PPVI pri raziskovanju grafa razlikuje stanja $s = \langle v, VI \rangle$, ki so definirana kot skupek vozlišča $v \in V$ in enega od njegovih varnih intervalov VI . V posamezno vozlišče lahko vstopimo večkrat (v različnih varnih intervalih).

V prvem koraku algoritma postavimo začetno stanje $s_{start} = \langle v_{start}, VI_{start} \rangle$ z začetnim vozliščem v_{start} na odprti seznam. Za vsako stanje s vodimo evidenco $\langle g, h, s_s, t_p \rangle$, ki vsebuje najmanjšo ceno g od začetnega stanja s_{start} do trenutnega stanja s , stanje starša s_s , heuristično oceno cene h od trenutnega stanja s do cilja v_{cilj} in najkrajši možen čas prihoda t_p v stanje s znotraj varnega intervala VI . V vsaki iteraciji z odprtega seznama vzamemo stanje $s_i = \langle v_i, VI_i \rangle$ z najnižjo skupno ceno $f_i = g_i + h_i$ in zanj poiščemo možne naslednike, ki jih dodamo na odprti seznam.

Ključno pri implementaciji algoritma PPVI je določitev naslednjih stanj za širitev iskanja. Za trenutno stanje s_i določimo v katera nova vozlišča lahko pridemo in v kateri varni interval lahko vstopimo. Pri tem preverimo tudi, da na povezavi do naslednjega vozlišča ni konflikta (npr. trka z drugim vozilom) oz. v primeru konflikta pa preverimo, če ga je možno preprečiti z ustreznim čakanjem v vozlišču v_i . Upoštevamo torej le tista sosednja vozlišča, ki jih agent (po povezavah z upoštevanimi hitrostnimi omejitvami in brez trkov z drugimi agenti) lahko doseže v varnih intervalih.

Če z odprtega seznama vzamemo stanje s ciljnim vozliščem $v_{cilj} = v_i$, končamo preiskovanje grafa, saj obstaja pot do cilja. V tem primeru najboljši plan π najdemo s pregledom prednikov, ki vodijo od ciljnega stanja s_{cilj} do začetnega stanja s_{start} . Če izčrpamo odprti seznam, preden naj-

demo ciljno vozlišče, iskana pot ne obstaja.

Dodatno vpeljemo še prioritete, ki omogočijo razklopljenost večagentnega planiranja, kar bistveno zmanjša računsko kompleksnost [1, 2]. Prioritete obenem ponujajo tudi možnost, da agenti opravljajo bolj ali manj pomembne naloge z različnimi prioritetami. Problem planiranja tako lahko rešujemo za vsakega agenta ločeno, ob upoštevanju gibanja ostalih višjeprioritetnih agentov. Torej, ko za agenta najdemo primeren plan, nastavimo ustrezne intervale zasedenosti vseh vozlišč, ki so povezana s stanji plana. Pri naslednjem nižjeprioritetnem planiranju nato upoštevamo posodobljene intervale zasedenosti vozlišč.

Z rekurzivno uporabo algoritma planiranja lahko tako dosežemo tudi iskanje poti od starta v_{start} do več zaporednih ciljev $(v_{cilj}(1), v_{cilj}(2), \dots, v_{cilj}(N))$. Ko dosežemo prvi cilj $v_{cilj}(1)$ in dobimo načrt poti $\pi_{cilj}(1)$, nadaljujemo z iskanjem plana $\pi_{cilj}(2)$ od trenutnega cilja $v_{cilj}(1)$ do naslednjega cilja $v_{cilj}(2)$ itd. Če najdemo pot do končnega cilja $v_{cilj}(N)$, združimo posamezne plane v celoten plan $\pi' = (\pi_{cilj}(1), \pi_{cilj}(2), \dots, \pi_{cilj}(N))$.

Ključno za računsko učinkovito izvedbo algoritma je tudi, da se vsi trki preverjajo implicitno le s preverjanjem časovnih intervalov. Glede na oblike, dimenzije in hitrosti agentov lahko vnaprej določimo minimalni varnostni čas (s katerim razširimo intervale zasedenosti), ki zagotovi, da se trki v okolici vozlišč nikoli ne zgodijo.

2.3 Varna mesta za kompletno planiranje

Varno mesto (VM) definiramo kot področje v prostoru (in grafu), kamor se agent po končani nalogi lahko varno umakne, če je potrebno, in neovirajoče čaka na naslednjo nalogo. V varno mesto lahko vstopajo le agenti, ki planirajo pot s ciljem v njem, drugače ga poti ne smejo prečkati. Varno mesto tako določimo z grafom $G_{VM} = \langle V_{VM}, E_{VM} \rangle$, ki ni del osnovnega grafa ($G \cap G_{VM} = \emptyset$). Znotraj G_{VM} določimo eno varno lokacijo, bodisi na eni izmed cest $e_{VM} \in E_{VM}$ bodisi v vozlišču $v_{VM} \in V_{VM}$. Med planiranjem poti do varnega mesta dovolimo iskanje tudi čez vsto-

pne povezave $e_{vstop} \in E_{VM}$, ki vodijo v graf G_{VM} . Če začnemo iskanje znotraj G_{VM} pa dovolimo iskanje tudi čez izstopne povezave $e_{izstop} \in E_{VM}$, ki vodijo do osnovnega grafa G . Definiramo lahko tudi več varnih mest.

Pri planiranju posameznega plana vedno poiščemo še plan poti π_{VM} od zadnjega cilja $v_{cilj}(k)$ do enega izmed varnih mest. Končni rezultat planiranja je tako plan $\pi = (\pi', \pi_{VM})$. Ta pristop omogoča, da znotraj G ne pride do blokad agentov (npr. v slepih ulicah). S tem postane planiranje kompletno, torej iskanje vedno najde pot, če le-ta obstaja. Smatramo, da je agent v času izvajanja plana π_{VM} prost, če še ni znotraj varnega mesta, zato se lahko izvajanje tega plana kadarkoli prekine, če je možno najti plan do novih zelenih ciljev. V kolikor to ni možno, se agent še vedno lahko umakne v VM in od tam nadaljuje kasneje.

Vsakemu varnemu mestu predpišemo tudi kapaciteto agentov, ki jih lahko sprejme. Za izvedljivost mora biti skupna kapaciteta vseh varnih mest večja ali enaka številu vseh agentov N_A . Iskanje plana π_{VM} lahko omejimo le v izbrano varno mesto ali pa v kateregakoli izmed nabora varnih mest. S predstavljenim konceptom lahko izvedemo serijsko in vzporedno izvedbo varnih mest. Če onemogočimo preverjanje trkov na določenih cestah znotraj G_{VM} , je možna izvedba tudi t. i. virtualnih varnih mest, kjer nadzor nad upravljanjem z agenti lahko prepustimo drugemu (avtonomnemu) nadzornemu sistemu.

Z vpeljavo varnih mest je omogočeno tudi vseživljenjsko delovanje, ko se naloge pojavljajo sproti in niso vnaprej znane. Osnovni koraki, ki skrbijo za dodeljevanje nalog, so:

1. Če so na voljo nove naloge in je prost vsaj en agent (je obiskal vse cilje in se pelje proti varnemu mestu ali že čaka v njem in je na voljo), dodelimo agentu najnižjo prioriteto in poiščemo plan poti za najstarejšo novo nalogo.
2. Če najdemo nov plan, prekinemo morebitno izvajanje trenutnega plana, sprostimo zasedenosti za neizveden preostanek trenutnega plana, nastavimo intervale zasedenosti za

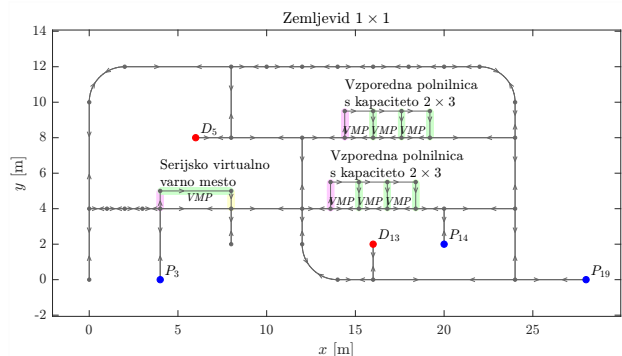
novi plan ter ga začnemo izvajati. Nadaljujemo s korakom 1.

3. Če novega plan ne najdemo, nadaljujemo trenutno pot proti varnemu mestu ali v njem nadaljujemo čakanje. Po določenem času nadaljujemo s korakom 1.

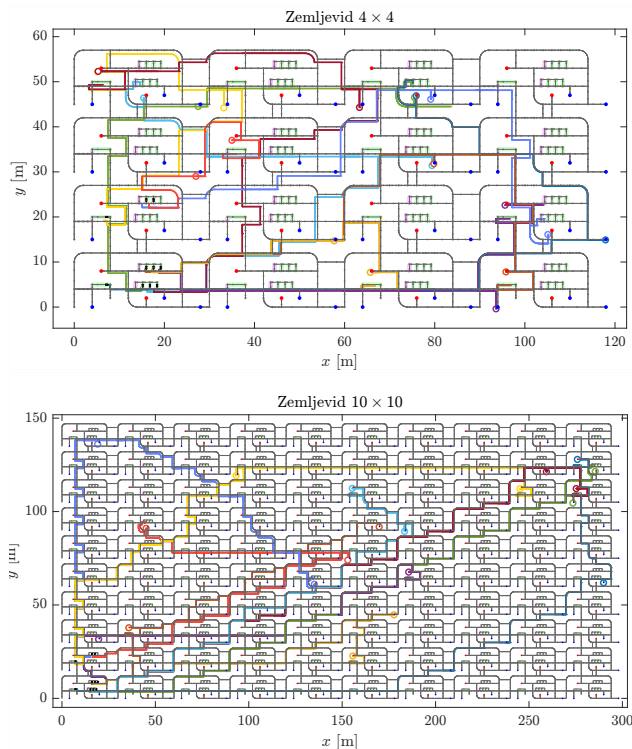
Varna mesta so praktično uporabna tudi kot polnilnice ali servisna mesta. Varnim mestom lahko torej priredimo tudi eno ali več storitev, ki jih ponujajo. V primeru, ko se upošteva tudi stanje energije agentov, lahko strategijo dodeljevanja nalog tako nekoliko spremenimo, da omogočimo polnjenje v polnilnicah. Agentu dodelimo nalogo le, če ima dovolj energije. Sicer poiščemo plan do najbližje proste polnilnice *VMP* (polnilnico rezerviramo). Če ni na voljo nobene proste polnilnice, pošljemo agenta začasno v običajno varno mesto *VM* dokler se ne sprostí vsaj ena polnilnica. Zahtevamo, da se plana do polnilnice ne sme prekiniti, dokler agent ne napolni svoje baterije. To je le preprosta strategija izbire polnilnic, ki je uporabljena v eksperimentih — strategije dodeljevanja polnilnic in nalog so izven obsega tega dela.

3 Analiza učinkovitosti delovanja

Simulacijsko analizo učinkovitosti trenutne izvedbe algoritma za večagentno planiranje smo opravili na različno velikih zemljevidih: osnovnem zemljevidu 1×1 (slika 1) in zemljevidih 4×4 ter 10×10 (slika 2), ki sta zgrajena iz kopij osnovnega zemljevida. Uporabili smo različno šte-



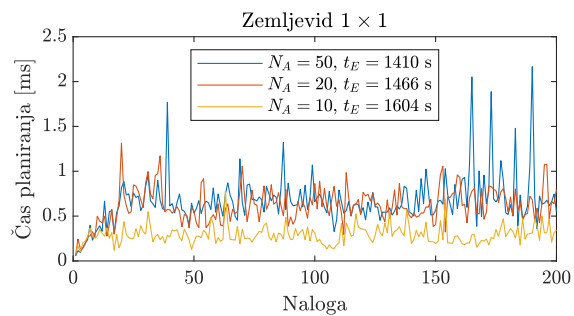
Slika 1: Zemljevid 1×1 s 53 vozlišči — označena so varna mesta *VM* (polnilnice *VMP*) in prevzemna *P* ter dostavna *D* mesta



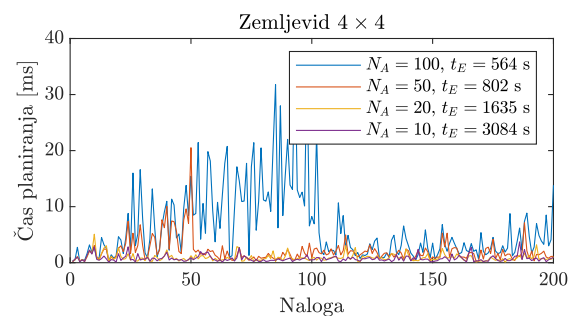
Slika 2: Zemljevid 4×4 z 836 vozlišči (zgoraj) in zemljevid 10×10 s 5210 vozlišči (spodaj) z nekaj začetnimi plani

vilo agentov, ki opravljajo naloge dostav. Vsaka naloga zahteva plan vožnje π_P iz trenutne lokacije agenta na naključno izbrano prevzemno mesto P , nato π_D na naključno dostavno mesto D in končno π_{VM} v varno mesto VM ali VMP . Celo ten plan vsake naloge je torej $\pi = (\pi_P, \pi_D, \pi_{VM})$. Med delovanjem agenti po potrebi opravljajo tudi polnjenje svojih baterij. Ko jim energija pade pod kritični nivo, prekinejo izvajanje nalog in iščejo plan π_{VMP} do najbližje proste polnilnice. Simulacijsko analizo smo izvedli na računalniku z Intel Core i7-6700HQ in 16 GiB delovnega spomina.

Na sliki 3 iz podatkov o časih t_E za izvedbo vseh nalog hitro ugotovimo, da dvakrat več agentov ne opravi nalog dvakrat hitreje. Pri $N_A = 50$ agentih postane promet zgoščen, zato je potrebno veliko dodatnega časa za koordinacijo izogibanja trkov, tako da je čas izvedbe nalog t_E za 50 agentov glede na 10 agentov le 12 % krajši. Odvisnost časa t_E za izvedbo vseh nalog glede na število agentov je podrobneje analizirana za srednji zemljevid 4×4 na sliki 6. Čas t_E se pri 20 agentih



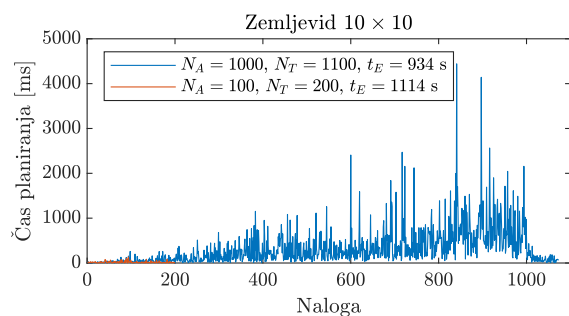
Slika 3: Časi planiranja na zemljevidu 1×1



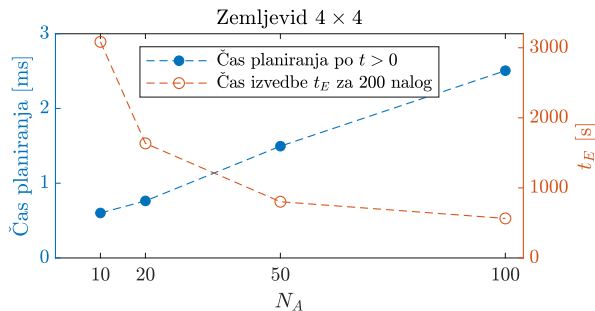
Slika 4: Časi planiranja na zemljevidu 4×4

skrajša na približno 50 % časa glede na 10 agentov, pri večanju števila agentov pa je prihranek časa zaradi naporov koordinacije ustrezno manjši. Tudi na velikem zemljevidu 10×10 opazimo, da 1000 agentov v primerjavi s 100 agenti, opravi le 6-krat več nalog v enakem časovnem obdobju.

Računsko zahtevnost, ki jo ocenimo s časom planiranja posamezne naloge, za $N_A \in \{10, 20, 50\}$ agentov in $N_T = 200$ nalog na zemljevidu 1×1 prikazuje slika 3. Planiranje večine posameznih nalog na osnovnem zemljevidu traja manj kot 1 ms. Na sliki 4 so podani časi planiranja na srednjem zemljevidu 4×4 , kjer naloge opravlja $N_A \in \{10, 20, 50, 100\}$ agentov. Slika 5 prikazuje



Slika 5: Časi planiranja na zemljevidu 10×10



Slika 6: Čas planiranja in čas izvedbe nalog t_E glede na število agentov N_A na zemljevidu 4×4

še rezultate na velikem zemljevidu 10×10 , kjer imamo največ $N_A = 1000$ agentov. Pri analiza časov planiranj opazimo približno linearno naraščanje časa planiranja za prvih N_A nalog, nato pa sledi občuten padec časa planiranja posamezne naloge. To začetno naraščanje časa planiranja se zgodi zaradi inicializacije, ko so ob času $t = 0$ vsi agenti prosti in se vsem hkrati dodelijo naloge. Vsak plan vnese nove intervale zasedenosti, ki lahko, ne pa vedno, razdrobijo varne intervale, kar povzroči dodatne vejitve stanj pri nižjeprioritetnih planiranjih. Kasneje se planiranje izvede le, ko določen agent opravi nalogo in postane prost ter je na voljo nova naloga. Takrat algoritem lahko pobriše znaten del intervalov zasedenosti v preteklosti, ki niso relevantni za prihodnost. To sprosti spomin in zmanjša število stanj, ki jih je potrebno pregledati tekom planiranja. Na sliki 6 je prikazana odvisnost časa planiranja od števila agentov za naloge po začetni inicializaciji. Opazimo, da kompleksnost narašča linearno s številom uporabljenih agentov. Tudi analiza časov planiranj na velikem zemljevidu 10×10 , ki sta 10 ms za $N_A = 100$ in 75 ms za $N_A = 1000$, to potrjuje. Glede na optimalne pristope v literaturi [1], kjer računska zahtevnost narašča eksponentno, je to za praktično uporabo obetaven rezultat.

4 Zaključek

V članku je bila predstavljena izvedba algoritma za večagentno planiranje poti. Predstavljeni algoritem je kompleten in omogoča vseživljenjsko delovanje ter je praktično uporaben v različnih scenarijih. Opisani so bili mehanizmi, ki omogočajo učinkovito delovanje algoritma. Računska

učinkovitost delovanja je bila preizkušena na več različno velikih zemljevidih in pri različnem številu agentov. Rezultati kažejo, da je že trenutna izvedba algoritma računsko zelo učinkovita in deluje hitro tudi za velike zemljevide z veliko agenti. V prihodnosti načrtujemo optimizacijo kode, kar pričakujemo, da bo prineslo še dodatne pohitritve.

Zahvala

Avtorji se zahvaljujemo za finančno podporo Javni agenciji za znanstvenoraziskovalno in inovacijsko dejavnost Republike Slovenije in podjetju Epilog d.o.o. (št. L2-60153 in P2-0219).

Literatura

- [1] G. Sharon, R. Stern in sod. Conflict-Based Search for Optimal Multi-Agent Pathfinding. *Artificial Intelligence*, zv. 219, str. 40–66, 2015.
- [2] A. Andreychuk, K. Yakovlev in sod. Multi-Agent Pathfinding with Continuous Time. *Artificial Intelligence*, zv. 305, št. 103662, str. 1–32, 2022.
- [3] T. Žužek, R. Vrabič in sod. Simulation-Based Approach for Automatic Roadmap Design in Multi-AGV Systems. *IEEE Transactions on Automation Science and Engineering*, zv. 21, št. 4, str. 6190–6201, 2024.
- [4] V. Digani, L. Sabbatini in sod. Ensemble Coordination Approach in Multi-AGV Systems Applied to Industrial Warehouses. *IEEE Transactions on Automation Science and Engineering*, zv. 12, št. 3, str. 922–934, 2015.
- [5] A. Zdešar, T. Žužek in sod. Vidiki koordiniranega načrtovanja poti na podlagi prioritet in časovnih oken za optimizacijo logistike avtomatsko vodenih vozil. V *Zbornik trinajste konference Avtomatizacija v industriji in gospodarstvu*, str. 1–9. Društvo avtomatikov Slovenije, 2023.
- [6] M. Phillips in M. Likhachev. SIPP: Safe Interval Path Planning for Dynamic Environments. V *IEEE International Conference on Robotics and Automation*, str. 5628–5635. 2011.
- [7] M. Bošnjak, A. Zdešar in sod. Managing Just-In-Sequence Micro-Logistics with Multi-Agent-Path-Finding. V *IEEE International Conference on Automation Science and Engineering (CASE)*, str. 404–409. 2024.
- [8] A. Zdešar, M. Bošnjak in sod. Vpeljava varnih mest za vseživljenjsko večrobotsko planiranje poti v interni logistiki. str. 163–166. Slovenska sekcija IEEE; Fakulteta za elektrotehniko, 2024.