Vidiki koordiniranega načrtovanja poti na podlagi prioritet in časovnih oken za optimizacijo logistike avtomatsko vodenih vozičkov

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

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

[2]Univerza v Ljubljani, Fakulteta za strojništvo, Aškerčeva c. 6, 1000 Ljubljana, Slovenija

[3]Epilog, Tehnološki park 22a, 1000 Ljubljana, Slovenija

andrej.zdesar@fe.uni-lj.si, tena.zuzek@fs.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

Aspects of coordinated planning based on priorities and time windows for optimization of the logistics with automated guided vehicles

The work deals with the theoretical and practical aspects of the logistics of autonomous vehicles in an industrial setup, which are considered in the context of an applied research project. The aim of the project is to use intelligent approaches for generation of optimal paths, timetables and task schedules for a fleet of automated guided vehicles. The paper presents approaches for coordinated path planning of vehicles moving between stations connected by a network of intersections and roads. The environment is represented in the form of a graph used for development and evaluation of algorithms in simulation. The basic task that a mobile agent has to perform is to move between any two stations, and before it begins the task, it can be in any location. The implemented solutions are based on safe interval path planning with priorities, where conflicts can be solved by waiting in nodes or on roads. Various extensions are shown that enable lifelong planning with priorities when new tasks are added, so that no intervention in existing plans is required. The solutions are based on merging plans into a task, adding nodes, using charging stations, introducing nearby nodes and other approaches.

Kratek pregled prispevka

Delo obravnava teoretične in praktične vidike logistike avtonomnih vozil v industrijskem prostoru, ki jih obravnavamo v okviru aplikativnega raziskovalnega projekta. Cilj projekta je z inteligentnimi pristopi poiskati optimalne poti, časovne načrte gibanja in razporejanja nalog za floto avtomatsko vodenih vozičkov. V članku so predstavljeni pristopi koordiniranega planiranja poti za vozila, ki se gibljejo med postajami, ki so povezane z omrežjem križišč in cest. Zemljevid okolja je zapisan v obliki grafa in ga uporabljamo za načrtovanje in preizkušanje algoritmov v simulatorju. Osnovna naloga, ki jo mora izvršiti mobilni agent je vožnja med dvema poljubnima postajama, pri čemer se pred izvajanjem naloge agent lahko nahaja na poljubni lokaciji. Izvedene rešitve temeljijo na algoritmu za planiranje poti z varnimi časovnimi intervali in prioritetami, kjer se konflikte lahko rešuje s čakanjem v vozliščih ali na cestah. Prikazane so različne razširitve, ki omogočajo vseživljensko planiranje s prioritetami, ko se dodajo nove naloge, tako da ni potreben poseg v obstoječe plane. Rešitve temeljijo na sestavljanju načrtov v nalogo, dodajanju vozlišč, uporabi polnilnic, uvedbi bližnjih vozlišč in drugih pristopih.

1 Uvod

Algoritmi za načrtovanje poti z več agenti večinoma uporabljajo predstavitev okolja z grafi in temeljijo na algoritmu A* [1, 2]. Da se izognemo kompleksnemu načrtovanju, koordinaciji, vodenju in lokalizaciji več robotskih vozil, so avtomatsko vodeni vozički (AGV, angl. *automated guided vehicle*) v praksi omejeni na gibanje po vnaprej določenih poteh in z znanimi lokacijami postaj vzdolž poti [3]. Optimalno usklajeno načrtovanje z uporabo pristopov, ki temeljijo na A* [2, 4], je možno le za majhno število vozil, saj kompleksnost narašča eksponentno s številom vozil. V praksi je zato potrebno računsko zahtevnost omiliti z uvedbo suboptimalnih pristopov, ki planirajo gibanje vsakega vozila ločeno in nato rešujejo konflikte z uvedbo prometnih pravil, lokalnega večagentnega usklajevanja, pogajanja ali prioritet [2, 4].

V članku so predstavljene nekatere aktivnosti aplikativnega raziskovalnega projekta z naslovom *Razvoj samo-učečega sistema za optimizacijo pravil vožnje avtonomnih transportnih vozil in njihovih časovno-prostorsko usklajenih aktivnosti*. Projekt združuje tri partnerje: Fakulteto za Elektrotehniko, Univerza v Ljubljani, Fakulteto za strojništvo, Univerza v Ljubljani, in podjetje Epilog d.o.o. Projekt sofinancira Javna agencija za raziskovalno dejavnost Republike Slovenije in podjetje Epilog d.o.o. za obdobje 2021 do 2024. Glavne aktivnosti vsebujejo izdelavo simulacijskih okolij za razvoj in vrednotenje rezultatov [5], razvoj algoritmov planiranje MAPF [6], avtomatizirano gradnjo zemljevida cest [7], samo-učeči moduli za dodeljevanje nalog in prenos znanj v prakso. V članku se osredotočamo na teoretične in praktične vidike optimalnega planiranja poti in časovne načrte gibanja, ki so računsko učinkoviti.

2 Definicija problema

Zemljevid okolja v katerem planiramo poti opišemo z grafom cest in križišč. Zemljevid torej predstavimo z naborom povezav ($e_i \in E$) in vozlišč ($v_j \in V$) kot graf prehajanja stanj $\mathscr{G} = \langle E, V \rangle$. Vozlišča v grafu predstavljajo križišča, ceste pa povezave med njimi. Povezave so utežne (cena predstavlja čas prehoda med vozliščema) in usmerjene (enosmerne ali dvosmerne povezave).

Rezultat planiranja je najdena pot $\pi = [a_1, a_2, \cdots, a_n)]$, ki vsebuje akcije a_i. Te opišejo pot s podanim hitrostnim profilom od startnega do ciljnega vozlišča z ustreznimi časovnimi trenutki. Akcijo opiše nabor $a_i = \langle t_i, rID, loc \rangle$, ki definira želen čas prihoda na lokacijo loc znotraj ceste rID. a_i je akcija premika, če se lokacija glede na a_{i-1} spremeni ($\langle rID_i, loc_i \rangle \neq \langle rID_{i-1}, loc_{i-1} \rangle$). Oziroma akcija predstavlja čakanje na lokaciji, če velja $\langle rID_i, loc_i \rangle = \langle rID_{i-1}, loc_{i-1} \rangle$. Osnovni SIPP algoritem omogoča le čakanje v vozliščih, tj. na začetku povezav ($loc = 0$).

2.1 Detekcija konfliktov

Tekom planiranja poti za skupino AGV-jev je potrebno pogosto preverjanje morebitnih konfliktov, iskati alternativne obvoze ali konflikte preprečiti s čakanjem. Trki se lahko zgodijo med vozili, ki vozijo po istih ali različnih povezavah in vozliščih. Detekcija trkov je kompleksna in pogosto zahteva računsko zahtevno numerično reševanje, saj so povezave poljubnih oblik in analitična rešitev kot v [8] ni možna, rešitev z numerično simulacijo [2] pa je računsko potratna.

V nadaljevanju predpostavljamo, da je konflikt možen le znotraj istih povezav ali vozlišč. To znatno poenostavi detekcijo trkov v postopku planiranja. Pri detekciji trka v vozlišču zgolj preverjamo ali lahko v vozlišče vstopimo v njegovem varnem intervalu. Pri določitvi trenutka vstopa upoštevamo dimenzije vozila (njegov očrtan radij). Detekcijo konfliktov znotraj iste povezave nato enostavno ugotavljamo s pomočjo intervalov zasedenosti povezav (OI_r). V kolikor dve vozili uporabljata isto povezavo je trk možen le, če se intervala OI_r prekrivata.

Če predpostavimo, da so konflikti možni le znotraj istih vozlišč in cest, morajo biti različne ceste dovolj narazen, da vozila med vožnjo po njih ne trčijo (zaradi končnih dimenzij vozil). V vseh situacijah omenjena predpostavka ne drži, možni primeri so: trk dveh vozil v vozliščih, ki sta nara-

zen za manj kot je dimenzija vozil ali trk vozil na različnih povezavah, ko sta vozili blizu vozlišča. Rešitev za te in podobne situacije obravnavamo v podpoglavju 4.2.1. Dodatno pa pri detekciji konfliktov vpeljemo še varnostni faktor, ki dejanski očrtan radij vozila poveča za nek faktor (npr. 1,6).

3 Algoritem SIPP

Osnovni algoritem SIPP (angl. *Safe Interval Path Planning*) [1] omogoča optimalno planiranje poti enega vozila v dinamičnih okoljih z znanimi trajektorijami ovir. SIPP izvira iz algoritem A*, ki je nadgrajen z varnimi intervali. Varni intervali so definirani za vozlišča grafa prehajanja stanj in definirajo maksimalni neprekinjen časovni interval v katerem je 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 drug agent. Unija varnih in zasedenih intervalov predstavlja celotno časovno os.

Osnovni algoritem SIPP (Algoritem 1) preiskuje graf in razlikuje stanja definirana z množico $s = \langle v, SI \rangle$, kjer je v vozlišče grafa in SI varni interval (en od več možnih). Algoritem začne v začetnem stanju (Algoritem 1, vrstica 1) $s_{start} = \langle v_{start}, SI_{start} \rangle$ in išče najboljšo pot do ciljnega stanja s_{goal}. V postopku iskanja optimalne (npr. najhitrejše) poti lahko v vozlišče v vstopimo večkrat z različnimi varnimi intervali. Za vsako stanje s vodimo informacijo $\langle g(s), h(s), parent(s), t_{EA}(s) \rangle$, ki vsebuje najmanjšo ceno od začetnega do trenutnega vozlišča ($g(s)$), starša ($parent(s)$), hevristično oceno cene do cilja ($h(s)$) in najkrajši možen čas dostopa do vozlišča t_{EA} znotraj intervala SI. V vsaki iteraciji s seznama OPEN vzamemo stanje s z najnižjo skupno ceno $f(s) = g(s) + h(s)$ (Algoritem 1, vrstica 4) in zanj poiščemo možne naslednike (Algoritem 1, vrstica 5). Če novi naslednik s' še ni bil obiskan oz. je že bil obiskan, a ima višjo ceno, potem ceno posodobimo z vsoto cene starša $g(s)$ in ceno prehoda $c(s, s')$ (Algoritem 1, vrstice 7-13).

Ključno pri implementaciji algoritma SIPP je določitev naslednjih stanj za širitev iskanja (Algoritem 1, vrstica 5), kjer za obstoječe stanje s (stanje je skupek vozlišča in varnega intervala) poiščemo možna naslednja stanja $s' \in S'$. To je, v katera nova vozlišča lahko pridemo in v kateri varni interval lahko vstopimo. Pri tem je potrebno preveriti tudi, da na povezavi do naslednjega vozlišča ni konflikta (npr. trka z drugim vozilom) oz. v primeru konflikta je potrebno ugotoviti, ali ga je možno preprečiti z ustreznim časom čakanja v s.

Algoritem 1 Algoritem SIPP

```
1:  g(s_start) = 0, OPEN = ∅
2:  insert g(s_start) into OPEN with f(s)
3:  while s_goal is not expanded do
4:      remove s with smallest f(s) from OPEN
5:      S' = getSuccessors(s)
6:      for each s' ∈ S' do
7:          if s' not visited before then
8:              f(s') = g(s') = ∞
9:          if g(s') > g(s) + c(s, s') then
10:             g(s') = g(s) + c(s, s')
11:             update t_EA(s')
12:             f(s') = g(s') + h(s')
13:             insert s' into OPEN with f(s')
```

4 Nadgradnje algoritma SIPP

V nadaljevanju je opisanih nekaj razširitev algoritma SIPP za namen večagentnega planiranja (MAPF, angl. *Multi-Agent Path Finding*).

4.1 Algoritem SIPP s prioritetami za MAPF

Z vpeljavo prioritet dosežemo razklopljenost večrobotskega planiranja poti, kar bistveno poenostavi računsko kompleksnost MAPF-algoritmov [2, 4]. Problem lahko rešujemo za vsako vozilo ločeno upoštevajoč zasedenosti ostalih. To je računsko bistveno manj zahtevno kot hkratno reševanje celotne skupine vozil. Vpeljava prioritet pa je hkrati tudi praktično uporabna. Definiramo lahko, da so naloge, ki se že izvajajo (npr. dostave po planiranih poteh), bolj prioritetne od novih nalog ali pa predpišemo želene prioritete. Osnovni koncept planiranja je sledeč:

1. Izvedemo planiranje poti za najbolj prioritetno vozilo in določimo zasedenosti vozlišč ter povezav vzdolž poti.

2. Nadaljujemo s planiramo manj prioritetnega vozila, pri tem upoštevamo zasedenosti vozlišč in cest višje prioritetnih vozil.
3. Korak 2 ponavljamo do najmanj prioritetnega vozila.
4. Začetek vožnje glede na določene plane.

4.2 *Obdelava grafa za splošnejša okolja*

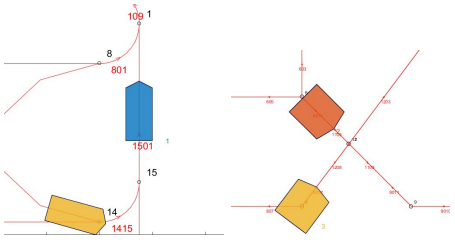
Prikazujemo nekaj dopolnitev grafa, da dosežemo ustrezno in učinkovitejše planiranje poti.

Poleg obstoječih vozlišč, ki predstavljajo križišča cest, lahko dodamo še dodatna vozlišča brez vejitev povezav. Primerna izbira dodatnih vozlišč omogoča efektiven umik s ceste (npr. pri koordinaciji vozil) in nato povratek nazaj. Brez dodatnega vozlišča mora vozilo (glede na osnovni SIPP) peljati do konca povezave po kateri se umika in se nato vračati.

Vozliščem in povezavam dodajamo različne oznake. Cestam lahko dodamo oznako enosmerne, dvosmerne ali večpasovne povezave. Pri slednji preverjanje konfliktov ni potrebno, saj se zanašamo na lokalno inteligenco AGV-jev, da zmorejo lokalno koordinacijo. Nekaj ostalih lastnosti oz. obdelav grafa podajamo v nadaljevanju.

4.2.1 *Bližnja vozlišča*

Osnovni graf predpostavlja, da so trki zgolj znotraj istih povezav ali v vozliču. V določenih situacijah lahko do trka pride tudi, ko so vozila na različnih povezavah ali vozliščih, kot prikazuje slika 1. Če bi skozi vozlišče 15 (slika 1 levo)



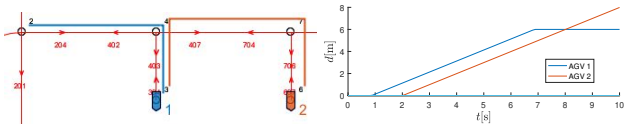
Slika 1: Vpeljava vmesnih vozlišč omogoča učinkovito detekcijo trka pri dveh bližnjih vozliščih (levo) ali pri križanju povezav (desno)

peljalo modro vozilo in bi hkrati v vozlišču 14 čakalo rumeno vozilo (manj prioritetno), da prepreči trk, bi se zaradi dimenzij vozil zgodil trk. To preprečimo, če vozliščema dodamo lastnost bližnjega vozlišča, kar pomeni, da je ob zasedenosti enega zasedeno hkrati tudi drugo vozlišče.

Podobno bi se trk zgodil na sliki 1 desno, ker se vozili vozita po cestah, ki se križata brez vozlišča. Če na mestu križanja na vsaki cesti dodamo svoje novo vozlišče in jima dodelimo lastnost bližnjega vozlišča, potem do trka ne pride. Ob zasedenosti enega novega vozlišča je tako hkrati zasedeno tudi drugo, zato rumeno vozilo čaka v predhodnem vozlišču, da do trka ne pride.

4.2.2 *Označbe slepih ulic*

Pri planiranju je koristno imeti informacijo o slepih ulicah. Zato lahko v fazi priprave grafa vozliščem in povezavam dodatno označimo to lastnost. Ta informacija je koristna v situacijah, ko vozilo začne ali konča vožnjo v slepi ulici. Če vozilo začne pot iz slepe ulice (slika 2) kot manj prioritetno vozilo, bi mu vozila z višjo prioriteto — ki planirajo in definirajo zasedenosti vozlišči in povezav prej — lahko preprečila izhod iz enosmerne ulice. To rešimo s predhodno zasedenostjo vozlišč in povezav, ki so del slepe ulice, za čas vožnje iz slepe ulice. V kolikor vozilo konča plan v slepi ulici (pomembno za vseživljenjski način delovanja, poglavje 5.3), mora predhodno preveri še možnost vrnitve iz slepe ulice, da je plan veljaven.



Slika 2: Planiranje v slepi ulici. Vozilo 2 ima nižjo prioriteto in namerava zapustiti vozlišče 3 ob času 2 s *(pred tem npr. nalaga tovor). Zato mora zasesti slepo ulico za čas do* $t = 4{,}44$ s *(čakanje* 2 s*, vožnja* 2 s *in varni čas zaradi dimenzij vozila* 0,44 s*). Vozilo 1, ki začne ob* $t = 0$ s *mora zato v začetnem vozlišču počakati določen čas (*0,87 s*), da vozilo 2 predhodno zapusti slepo ulico. Časovno izvajanje z napredovanjem prevožene razdalje prikazuje desna slika.*

4.3 Čakanje na povezavah

Osnovni algoritem SIPP koordinacijo izvede s čakanjem v vozliščih (na začetkih povezav). Čakanje v vozlišču lahko ostalim vozilom na drugi cesti za čas čakanja onemogočijo prečkanje križišča. Če premaknemo čakanje na povezavo, lahko v določenih primerih dosežemo boljšo pretočnost križišč. Nadgradnja SIPP algoritma, ki bi omogočal čakanje na poljubnem mestu na povezavi, je možna, a zahteva kompleksen poseg v algoritem, kar ga naredi manj preglednega in računsko zahtevnejšega. Zato vpeljemo dve enostavni in praktično uporabni možnosti za čakanje vzdolž povezav.

Čakanje na povezavi lahko enostavno dosežemo s preureditvijo grafa in dodajanjem vozlišč na ceste (izven križišč), kjer želimo in dovolimo vozilom čakanje (poljubno mesto za čakanje na povezavi namreč ni zaželeno). Dodana vozlišča na povezavah ne povečujejo vejitve grafa in je zato dodatna računska obremenitev zanemarljiva.

Čakanje na cesti lahko enostavno dosežemo tudi tako, da prvotno mesto čakanja v vozlišču na začetku povezave premaknemo na povezavo. Govorimo o manipulaciji plana [9] s katero za vsako povezavo ohranimo čas vstopa in izstopa iz povezave, mesto čakanja pa le prestavimo na povezavo. Pri tem upoštevamo dovoljena območja čakanja na povezavi in predhodna vozila, ki na cesti že čakajo.

5 Izvajanje nalog

Do sedaj smo omenjali le osnovne naloge, kjer vozila iz nekih začetnih leg vozijo v predpisano končno lego. V končni legi lahko dodatno predpišemo zahtevan postanek oz. čakanje (npr. za manipulacijo tovora). Ta postanek je lahko končen, po katerem vozilo nadaljuje vožnjo ali pa neskončen, če ostane parkirano za nedoločen čas. Omenjen postanek zagotovimo v času iskanja poti, kjer za končanje iskanja (Algoritem 1, vrstica 3) dodamo pogoj, da je ciljno vozlišče v varnem intervalu prihoda ($s_{goal} = \langle v_{goal}, SI_{goal} \rangle$) prosto vsaj za želen čas postanka. S tem zagotovimo, da v času postanka v vozlišče ne vstopi nobeno bolj prioritetno vozilo. Z dodano zasedenostjo za ciljno vozlišče (OI_{goal}) pa preprečimo vstop manj prioritetnim vozilom.

V nadaljevanju je prikazana še izvedba bolj kompleksnih nalog, kot so naloge s predpisanim vrstnim redom dostav, sestavljene naloge in vseživljenjsko delovanje. Pri planiranju za izvajanje teh nalog je osnovno vodilo določitev optimalnih oz. blizu-optimalnih planov, ki ne potrebujejo ponovnega planiranja ob dodajanju novih nalog in so računsko učinkoviti.

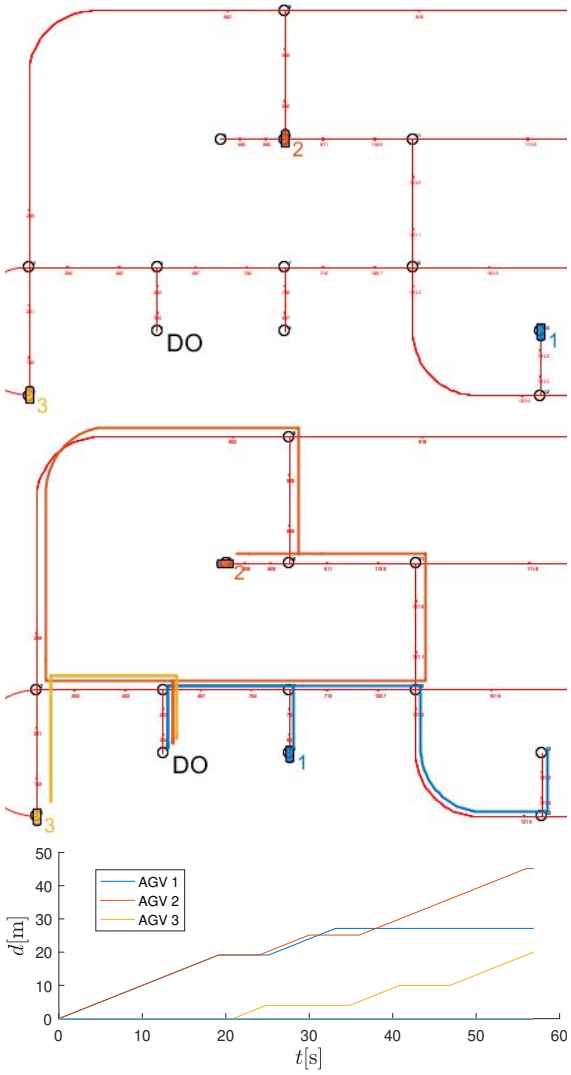
5.1 Predpisan vrstni red prihodov

Vrstni red dostav vozil je lahko predpisan pri sestavljanju nekega izdelka, kjer morajo AGV-ji komponente dostaviti v predpisanem vrstnem redu (primer prikazuje slika 3). Vozila začno iz začetnih pozicij, nato dostavijo v dostavno mesto *DO* in se umaknejo vsak na svoje končno mesto. Vrstni red dostav je predpisan s prioritetami, vozila planirajo pot po prioritetah, najprej AGV1, nato AGV2 in AGV3.

Plan poti AGV1 je določen iz dveh delov π_1 in π_2. π_1 je od začetka do *DO*. Vrstni red prihodov v *DO* zagotovimo z dodatnim pogojem za najdeno pot v Algoritmu 1 (vrstica 3), da je vozliče *DO* v varnem intervalu prihoda prosto do neskončnosti. S tem zagotovimo, da v vozlišče *DO* vstopimo zadnji, torej za vsemi vozili z višjo prioriteto, saj je le tedaj varni interval vozlišča prost do neskončnosti. Hkrati preprečimo, da bi katerokoli manj prioritetno vozilo prišlo prej (in zapustilo *DO* pred našim prihodom). Drugi del plana π_2 se začne s predpisanim postankom (6 s na sliki 3) in nato pot od *DO* do končnega mesta. Končni skupni plan je zlepek $\pi_{AGV1} = [\pi_1, \pi_2]$, nato določimo zasedenosti vozlišč in povezav vzdolž plana in enako nadaljujemo s planiranje za manj prioritetna vozila.

Na podoben način lahko vpeljemo več dostavnih mest in le za tiste, kjer je to potrebno definiramo vrstni red prihodov (vrstni red za vsa izbrana dostavna mesta je enak). Če vrstni red prihodov v *DO* ni predpisan, potem je pogoj za končanje iskanja poti (Algoritem 1, vrstica 3) le, da je končno

vozlišče prosto za nameravan čas postanka v njem. To pomeni, da lahko neko manj prioritetno vozilo obišče isto vozlišče pred nami in ga pravočasno zapusti, brez da bi oviral bolj prioritetna vozila.

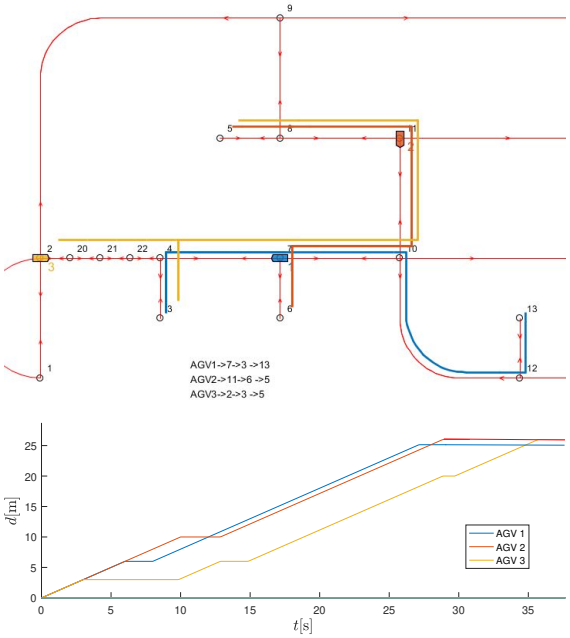


Slika 3: Predpisan vrstni red prihodov na dostavno mesto DO, najprej AGV1, nato AGV2 in zadnji AGV3. Začetne lege vozil (zgoraj), končne lege vozil po opravljeni poti od izhodišča do dostavnega mesta in na končno mesto (sredina). Časovni potek opravljenih razdalj (spodaj), kjer se vidi, da AGV1 ob $t = 20\,\mathrm{S}$ dospe v DO in tam čaka 6S, podobno AGV2 prispe ob $t = 30S$ in AGV3 ob $t = 41S$. Vozili AGV2 in AGV3 čakata še na dodatnih mestih, da preprečita trk z višjeprioritetnim vozilom.

5.2 Sestavljene naloge

Nalogo lahko sestavimo iz več zaporednih opravil. Praktičen primer bi bil sledeč. Vozilo iz svoje trenutne lokacije (TM) najde pot do prevzemnega mesta (PM), kjer naloži tovor in ga pelje do dostavnega mesta (DM), kjer ga odloži. V PM in DM mora počakati določen čas, ki je potreben za manipulacijo tovora. Dodatno lahko zagotovimo tudi predpisan vrstni red za odjeme in/ali dostavne, kot je opisano v poglavju 5.1.

Ker uporabljamo prioritete, je možno take sestavljene naloge ($TM \rightarrow PM \rightarrow DM$) enostavno zložiti iz osnovnih planov $\pi_{AGV_i} = [\pi_1, \pi_2, \pi_3]$, kjer π_1 definira premik iz TM v PM in π_2 premik iz PM v DM. Zaradi kompletnosti planiranja pri tem dodamo še π_3, ki poskrbi za odmik vozila iz DM na neko varno mesto, kjer vozilo lahko čaka dokler ne dobi nove naloge. Odmik iz DM je potreben, da lahko ostali AGV-ji nemoteno do-



Slika 4: Sestavljene naloge za tri vozila. Vozila so narisana na začetni lokaciji TM, njihova pot do PM in DM pa je označena z barvno črto. AGV1 z najvišjo prioriteto planira pot od TM v vozlišču 7 do PM v vozlišču 3 in do DM v vozlišču 13 ($7 \rightarrow 3 \rightarrow 13$), podobno AGV2 planira pot med vozlišči $11 \rightarrow 6 \rightarrow 5$ in AGV3 $2 \rightarrow 3 \rightarrow 5$. Prikazani so še časovni profili napredovanja poti.

stavljajo. Slednje se navezuje na vseživljenjsko planiranje, ki je opisano v poglavjih 5.3 in 5.4.

Primer sestavljenih nalog prikazuje slika 4, kjer je posamezni plan sestavljen le iz dveh osnovnih planov $\pi_{AGV_i} = [\pi_1, \pi_2]$, zato vozila končajo in ostanejo v *DM*.

5.3 Vseživljenjsko delovanje

Do sedaj predstavljene rešitve MAPF in večina rešitev v literaturi so osredotočene le na enkratno planiranje, kjer vsa vozila začno hkrati in imajo le eno dostavo. To namreč predstavi bistvo zahtevnega problema, kar je za večjo nazornost koristno. V praktični izvedbi pa vozila po izvedeni dostavi dobijo nove naloge, ki jih morajo uskladiti z obstoječimi nalogami ostalih AGV-jev. Govorimo o vseživljenjskem oz. neprekinjenem delovanju (angl. *lifelong*).

Vedno je možno ob vsaki spremembi (npr. pojavu nove naloge) ponovno planiranje poti za vsa vozila, a se lahko zgodi, da rešitve za kako vozilo z novo nalogo ni moč najti. Tak primer je lahko višje prioritetno vozilo, ki uspešno konča svojo nalogo in je prosto za prevzem naslednje naloge. Ko mu je dodeljena nova naloga (prej neznana) jo rešuje z najnižjo prioriteto (naloge v izvajanju imajo prednost). Rešitve (plana poti) za novo nalogo algoritem lahko ne najde, saj so lahko povezave, ki vodijo do njega, že zasedene z nasprotno smerjo vožnje drugih vozil. V tem primeru bi bilo potrebno ponovno planirati poti za vsa vozila oz. za vsaj za tiste v konfliktu in jim iterativno spreminjati in iskati ustrezne prioritete. V izogib tega v nadaljevanju predstavimo idejo brez potrebnega ponovnega planiranja, ki je torej računsko nezahtevna, optimalna (pod vpeljanimi predpostavkami — prioritete) in vedno izvedljiva (kompletnost).

Novim nalogam dodelimo najnižjo prioriteto, kar je praktično iz dveh vidikov. Naloge, ki se že izvajajo, imajo prednost pri izvajanju, nove naloge se jim prilagodijo. Dodatno zato ni potrebno ponovno planirati starejših nalog, kar pohitri algoritme. Če novi nalogi želimo dati višjo prioriteto (npr. da se izvaja urgentno), potem je potrebno ponovno planirat vse naloge, ki se že izvajajo in imajo nižjo prioriteto.

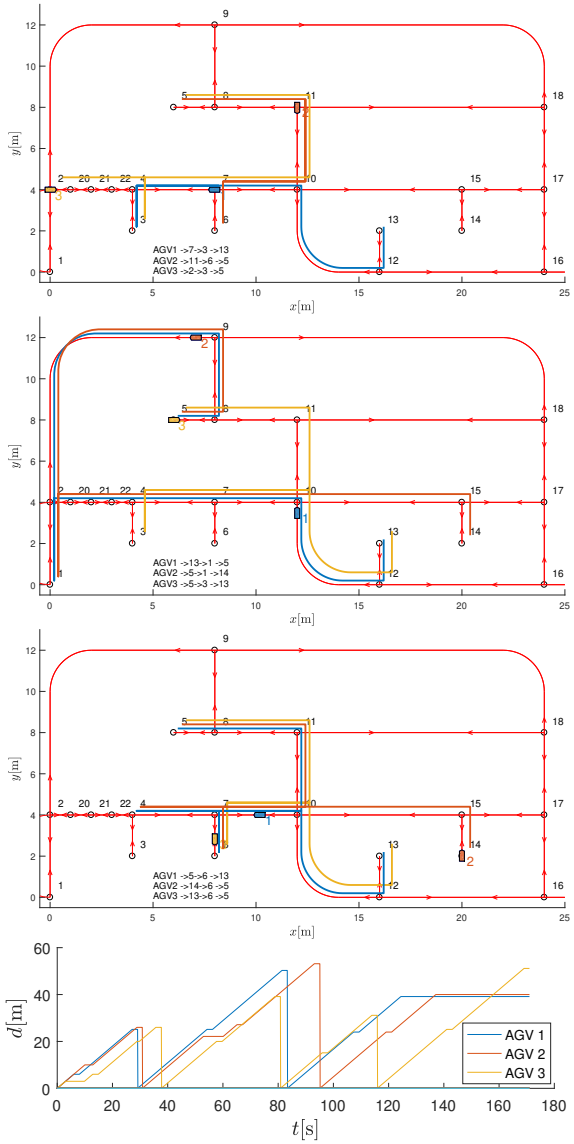
Postaje *PM* in *DM* so pogosto v slepih ulicah kot prikazuje primer na sliki 4. Zato je potrebno pri sestavljenih nalogah, za vsak delni plan π_j ($j \in 1, \cdots, n$, n je število delnih planov) preveriti, če se le-ta konča v slepi ulici. Če je končno vozlišče v slepi ulici (ena od lastnosti vozlišča), potem pri iskanju poti dodamo pogoj (Algoritmu 1, vrstica 3), da je končno vozlišče v_{goal} prosto vsaj do $t_{SI_2} \geq t_{\pi_j} + 2L/vel$, kjer je t_{π_j} končni čas plana π_j, t_{SI_2} je zgornja meja SI_{goal}, L je dolžina slepe ulice in vel potovalna hitrost. Dodatni zahtevani prosti čas $2L/vel$ za končno vozlišče, torej vsebuje čas, ki ga potrebujemo, da izstopimo iz slepe ulice ter dodatno še čas, ki ga neko drugo višjeprioritetno vozilo potrebuje od začetka slepe ulice do ciljnega vozlišča (da ga zasede). S tem najdemo rešitev, ki ima zagotovljen povratek iz slepe ulice in jo lahko enostavno lepimo z naslednjim delnim planom π_{j+1}.

Primer planiranja 10 nalog za 3 vozila prikazuje slika 6. Naloge niso znane vnaprej, vozilo planira naslednjo nalogo, ko konča trenutno. Vsaka naloga vsebuje vožnjo od *TM* preko *PM* do *DM*.

Dodatno pozornost zahteva končno ciljno vozlišče *DM*, kjer moramo, da zagotovimo kompletnost, dodati še zadnji delni plan π_n. To podrobneje opišemo v poglavju 5.4 z vpeljavo polnilnih mest.

5.4 Polnilna mesta

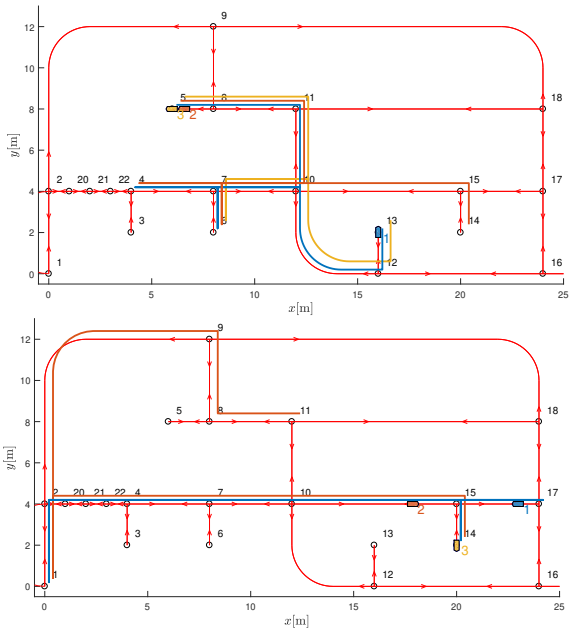
Električni AGV-ji imajo omejeno avtonomijo in potrebujejo polnjene na polnilnih mestih. Strategija izbire primernega časa polnjenja, izbira lokacij polnilnih mest ter njihove gostote je kompleksen proces [10, 11]. V nadaljevanju sta prikazani dve enostavni izvedbi polnilnice. Prikazano je, da lahko polnilna mesta izboljšajo problem planiranja in ne zgolj rešujejo problem polnjena akumulatorjev. S primerno izbiro polnilnih mest je namreč možno problem neprekinjenega planiranja MAP poenostaviti, da je računsko bolj učinkovit (manj ali brez ponovnega planiranja ob pojavu



Slika 5: Izvajanje 10 nalog s tremi AGVji. Na prvi sliki AGV1 vozi od TM vozlišča 7 do PM 3 in konča v DM 13 ($7 \to 3 \to 13$), podobno AGV2 $11 \to 6 \to 5$ in AGV3 $2 \to 3 \to 5$. Ko AGV konča, nadaljuje planiranje nove naloge, druga slika prikazuje naloge 4 do 6 in tretja slika naloge 7 do 9. Četrta slika prikazuje napredovanje nalog.

novih nalog) in zagotavlja popolnost rešitve.

Pri vseživljenjskem planiranju (glej poglavje 5.3) je potrebno za izvedljivost (popolnost algoritma planiranja) vsaki nalogi dodati še zadnji delni plan π_n. Ta pripelje vozilo do lokacije, kjer vedno lahko varno čaka dokler ne dobi naslednje naloge. V času izvajanja π_n je vozilo prosto 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 vozilo lahko vedno umakne na varno mesto in od tam kasneje nadaljuje. Prikladno možnost za izvedbo varnega mesta predstavljajo polnilnice.

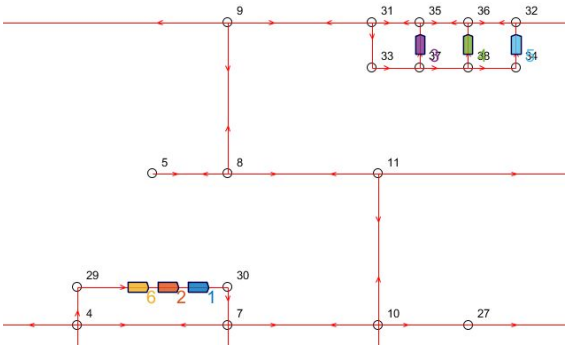


Slika 6: Primera neuspešnega izvajanja sestavljenih nalog. Na gornji sliki AGV3 konča dostavo in čaka v DM 5, ker ne dobi nove naloge. S tem prepreči dostavo AGV2. Na spodnji sliki AGV3 uspešno konča svojo nalogo (z višjo prioriteto kot AGV1 in AGV2) v DM 14, dobi novo nalogo (z najnižjo prioriteto), a zanjo ne uspe najti rešitve, saj mu AGV1 in AGV2 blokirata pot.

Na sliki 6 prikažemo dva primera vseživljenjskega planiranja, kjer vozilo ne uspe najti plana, saj pri lepljenju delnih planov ni dodan umik na varno mesto. Z vpeljavo polnilnice vpeljemo varno mesto, kamor vozila lahko v vsakem trenutku planirajo pot in jo tudi izvajajo, če ni druge alternative za njeno prekinitev. Dva primera izvedbe polnilnice, ki poleg polnjenja zagotavlja varno mesto, za umik prikazuje slika 7.

6 Zaključek

V delu je opisano planiranje poti več avtomatsko vodenih vozil v industriji z uporabo algoritma SIPP in vpeljavo prioritet. Osnovni vodilo pri razvoju rešitev je računska učinkovitost algoritmov in optimalnost delovanja. Nakazane so različne razširitve, ki omogočajo vseživljenjsko planiranje s prioritetami, ko se dodajo nove naloge, tako da



Slika 7: Serijska polnilnica (spodaj levo) z vstopom skozi vozlišče 4 in izstopom skozi vozlišče 7 po sistemu prvi noter prvi ven. V vzporedno polnilnico (zgoraj desno) vozila vstopijo skozi vstopno vozlišče 31, izstopajo pa skozi vozlišča 35, 36, 32 v poljubnem vrstnem redu.

ni potreben poseg v obstoječe plane. Rešitve temeljijo na sestavljanju načrtov v nalogo, dodajanju vozlišč, uporabi polnilnic, uvedbi bližnjih vozlišč in drugih pristopih.

Zahvala

Avtorji se zahvaljujejo za finančno podporo Agencije Republike Slovenije za raziskovalno dejavnost in podjetju Epilog d.o.o. (št. L2-3168, P2-0219 in P2-0220).

Literatura

[1] M. Phillips in M. Likhachev. Sipp: Safe interval path planning for dynamic environments. str. 5628 – 5635. 2011.

[2] A. Andreychuk, K. Yakovlev in sod. Multi-agent pathfinding with continuous time. *Artificial Intelligence*, zv. 305, str. 103662, 2022.

[3] V. Digani, L. Sabattini in sod. Ensemble coordination approach in multi-agv systems applied to industrial warehouses. *IEEE Transactions on Automation Science and Engineering*, zv. 12, str. 922–934, 2015.

[4] G. Sharon, R. Stern in sod. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, zv. 219, str. 40–66, 2015.

[5] A. Zdešar, M. Bošnak in G. Klančar. Cyber-physical platform with miniature robotic vehicles for research and development of autonomous mobile systems. V I. Petrovic, E. Menegatti in I. Marković (Uredniki), *Intelligent Autonomous Systems 17*, str. 897–908. Springer Nature Switzerland, Cham, 2023.

[6] M. Ljubi, G. Klančar in A. Zdešar. Path planning of multiple automatic guided vehicles with tricycle kinematics considering priorities and occupancy time windows. V I. Petrovic, E. Menegatti in I. Marković (Uredniki), *Intelligent Autonomous Systems 17*, str. 883–896. Springer Nature Switzerland, Cham, 2023.

[7] R. Vrabič, T. Žužek in sod. Improving the flow in multi-robot logistic systems through optimization of layout roadmaps. V I. Petrovic, E. Menegatti in I. Marković (Uredniki), *Intelligent Autonomous Systems 17*, str. 923–934. Springer Nature Switzerland, Cham, 2023.

[8] S. Guy in I. Karamouzas. *Guide to anticipatory collision avoidance*, str. 195–208. CRC Press, 2015.

[9] M. Ljubi. *Načrtovanje poti vozil z različnimi prioritetami na osnovi algoritma SIPP*. Magistrsko delo, Fakulteta za elektrotehniko, Univerza v Ljubljani, 2023.

[10] X. Zhan, L. Xu in sod. Study on agvs battery charging strategy for improving utilization. *Procedia CIRP*, zv. 81, str. 558–563, 2019.

[11] C. Luo, Y.-F. Huang in V. Gupta. Placement of ev charging stations–balancing benefits among multiple entities. *IEEE Transactions on Smart Grid*, zv. 8, str. 1–10, 2015.