Nadgradnja algoritma A* za planiranje poti več robotskih vozil po prioritetnem sistemu

Nejc Presečnik, izr. prof. dr. Gregor Klančar
Univerza v Ljubljani, Fakulteta za elektrotehniko
Tržaška c. 25, 1000 Ljubljana, Slovenija
nejc.presecnik@gmail.com, gregor.klancar@fe.uni-lj.si

Upgrade of algorithm A* for priority-based path planning of multiple automated guided vehicles

The article proposes a new path planning approach for mutual operation of group of automated guided vehicles (AGV) in the transport of material in production facilities. The presented algorithm is based on the well-known A* algorithm, which is upgraded for path planning of multiple AGVs in a way that finds a compromise solution without collisions and unnecessary congestions. The approach considers the priorities of transport orders, the map in the form of a weighted oriented graph, and the predicted time windows of the occupancy of the map segments. The algorithm first finds a path for AGVs with higher priorities. At end of every AGV path planning, the algorithm marks predicted time windows of occupancy of roads and nodes according to found path. These occupancies are then considered when planning a path for AGV with lower priority so that it does not obstruct the driving of AGVs with a higher priority and avoids conflict. Two important options that the algorithm considers and suggests are waiting in front of the node for the path to be released and the possibility of retreating to the side road. The approach is evaluated in simulation cases.

Kratek pregled prispevka

Članek predlaga nov pristop planiranja poti za usklajeno delovanje skupine mobilnih robotskih vozil pri transportu materiala v proizvodnih obratih. Predstavljen algoritem temelji na znanem algoritmu A*, ki je nadgrajen za planiranje poti več robotskih vozil, tako da najde kompromisno rešitev brez trkov in nepotrebnih zastojev. Pristop upošteva prioritete transportnih nalogov, zemljevid v obliki uteženega usmerjenega grafa ter predvidena časovna okna zasedenosti segmentov zemljevida. Algoritem najprej poišče pot za vozila z višjimi prioritetami. Ob vsakem planiranju poti vozila, algoritem na koncu zabeleži predvidena časovna okna zasedenosti cest in vozlišč na zemljevidu za najdeno pot. Te zasedenosti se nato upoštevajo pri iskanju poti za vozilo z nižjo prioriteto, tako da ne ovira vožnje vozil z višjo prioriteto in se izogne konfliktom. Dve pomembni možnosti, ki jih algoritem upošteva in predlaga, sta čakanje pred vozliščem, da se pot sprosti in pa možnost umika na stransko cesto v primeru onemogočenega čakanja. Pristop je ovrednoten na simulacijskih primerih.

1 Uvod

Z vse bolj pogosto avtomatizacijo skladišč in proizvodnih obratov je postalo tudi raziskovanje na področju avtonomnih robotskih vozil zelo popularno. Eden bistvenih problemov, ki se tičejo avtonomnih robotskih vozil, je planiranje poti. Medtem ko je planiranje poti za eno vozilo že bolj ali manj rešen problem, pa planiranje poti za več vozil predstavlja zanimiv raziskovalni problem, ki ga raziskujejo različni članki.

V splošnem lahko glavne pristope razdelimo na centralizirane in decentralizirane. Pri centraliziranih pristopih obstaja centralna enota, ki povezuje vsa robotska vozila in jim hkrati določi načrt poti [1, 2, 3]. Njihova glavna prednost je, da omogočajo optimalnost načrtanih poti. Pri bolj obsežnih zemljevidih po navadi postanejo taki pristopi časovno preveč potratni, kar omejuje uporabo v realnih sistemih. Rešitev za kompleksnost, ki jo prinašajo zemljevidi večjih razsežnosti, ponujajo decentralizirani pristopi, ki so v splošnem hitrejši od centraliziranih. V teh pristopih se naloga določanja poti iz višjega nivoja prestavi na sama vozila, tako da avtonomno določajo vsak svojo pot, hkrati pa sprotno rešujejo konflikte in zbirajo informacije o drugih vozilih. Planiranje poti je pogosto izvedeno z algoritmi, ki so namenjeni za eno vozilo, npr. A*, podane pa so različne strategije za reševanje konfliktov, ko ti nastopijo [4, 5, 6].

V našem delu predstavljamo centraliziran pristop, ki poda sub-optimalno rešitev in hkrati ohranja uporabno časovno potratnost. Pristop temelji na upoštevanju prioritetnih nalogov, ki omogočajo prednostno obravnavo pri iskanju poti za vozila z urgentnimi zadolžitvami, pri normalnem obratovanju pa prednost ohranjajo vozila, ki so zadolžitve prejela pred drugimi. Algoritem pri iskanju poti preverja časovna okna zasedenosti cest in vozlišč, ki so določena ob vsaki načrtani poti. V primeru konflikta predlaga čakanje na poti do tja ali umik na stransko cesto.

2 Metodologija

Predlagan algoritem je nadgradnja algoritma A* in omogoča planiranje poti za več robotskih vozil, ki se hkrati vozijo na istem zemljevidu. Algoritem se izvaja ločeno za vsako vozilo posebej po prioritetnem sistemu. Vsakemu vozilu določimo stopnjo prioritete. Večja kot je stopnja prioritete, bolj pomembno je, da to vozilo doseže cilj v najkrajšem možnem času. Preden izvedemo algoritem na kateremkoli vozilu, je zemljevid popolnoma prost. Vsa vozlišča in povezave so brez predvidenih časovnih oken zasedenosti. Algoritem najprej izvedemo na problemu vozila z najvišjo prioriteto. Ker je zemljevid še prost, bo algoritem zanj našel optimalno pot. Hkrati bo določil časovna okna zasedenosti za ceste in vozlišča v časovnih trenutkih, ki jih narekuje rezultat planiranja poti. Ko zaženemo algoritem na robotu z nižjo prioriteto, bo z upoštevanjem zasedenosti časovnih oken cest in vozlišč zanj našel čim hitrejšo možno pot, ki ne bo ovirala vozil z višjo prioriteto.

Osnovni potek iskanja poti za posamezno vozilo je enak kot pri algoritmu A* [7]. Algoritem najprej doda začetno vozlišče na odprti seznam. Nato v zanki izvaja iterativni postopek, dokler ni odprti seznam prazen oz. dokler iz odprtega seznama na zaprti seznam ne prestavi ciljnega vozlišča, kar pomeni, da je pot najdena. V iterativnem postopku najprej iz odprtega na zaprti seznam prestavi vozlišče z najmanjšo skupno ceno. Nato za vsako sosednje vozlišče določi cene ter preveri zasedenost vozlišča in pripadajoče ceste. Če je pot prosta, algoritem nadaljuje z enakim potekom kot A*. Glavna razlika med algoritmoma nastopi, kadar algoritem pri odpiranju sosednjega vozlišča in preverjanju zasedenosti naleti na konflikt. V tem primeru s poizkušanjem išče prosta mesta za čakanje in izogib konfliktu z vračanjem po poti, po kateri je prišel. Pseudo-algoritem z opisanim postopkom izogibanja konfliktov je podan v algoritmu 1.

Algoritem 1: Planiranje poti za več vozil.
Inicializacija: Dodelitev prioritet transportnim nalogom za N vozil. Predstavitev prostora z grafom prehajanja stanj in okni zasedenosti.

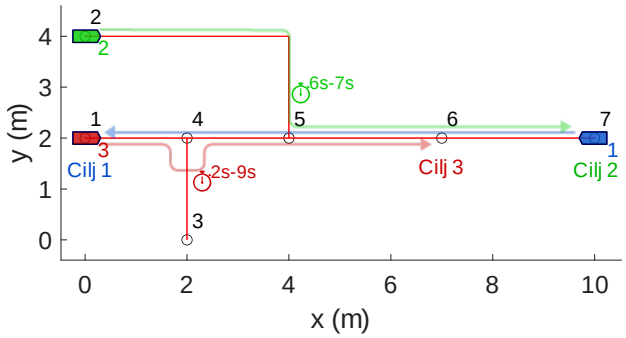
For: vozilo s prioriteto i = 1 do N
 -V zanki izvajaj iterativni postopek iskanja poti…
 …po algoritmu A*
 -V primeru konflikta izvajaj:
  LoopWhile: (mestoKonflikta != začetniPoložajVozila)
   If: *prosto*(cesta pred mestoKonflikta)
    -Predlagaj čakanje na cesti pred mestoKonflikta
    -*Break LoopWhile*
   End If
   If: *prosto*(stranska cesta pred mestoKonflikta)
    -Predlagaj umik in čakanje na stranski cesti…
    …pred mestoKonflikta
    -*Break LoopWhile*
   End If
  -Prestavi mestoKonflikta na predhodno cesto po…
  …poti nazaj
  End LoopWhile

-Določi okna zasedenosti za vozlišča in ceste na poti

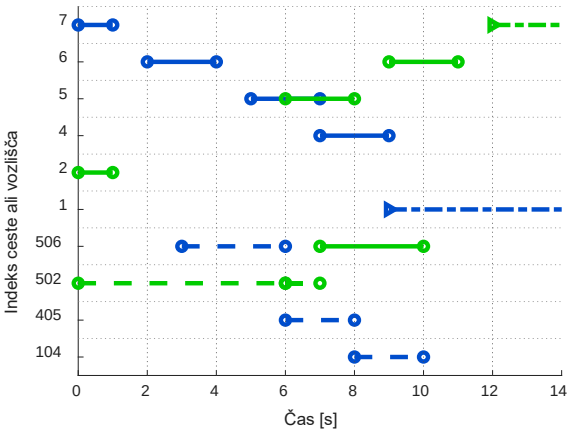
End For

2.1 Opis algoritma na primeru

Algoritem bo predstavljen na primeru, ki je prikazan na sliki 1, kjer lahko vidimo začetne položaje treh robotskih vozil. Modro vozilo 1 z najvišjo prioriteto pot prične na vozlišču 7 in konča na vozlišču 1. Zeleno vozilo 2, ki je naslednje po prioriteti, prične pot na vozlišču 2 in konča na vozlišču 7. Zadnje, rdeče vozilo 3, katerega postopek iskanja poti bomo po korakih opisali, pot prične na vozlišču 1 in konča na vozlišču 6. Koordinate so predstavljene v metrih, hitrosti vseh vozil pa so 1m/s. Poti za vozila 1 in 2 sta že načrtani. Posledično so za te poti določena tudi časovna okna zasedenosti, ki za vsako cesto določajo čas prihoda, odhoda in čakanja vozila na njej. Prav tako za vsako vozlišče določajo trenutek prihoda vozila nanj. Časovna okna zasedenosti za načrtane poti vozil 1 in 2 lahko vidimo na sliki 2. Vsaka črta predstavlja časovni interval oz. časovno okno, ko vozilo zasede cesto ali vozlišče. Levo od črte je zapisana oznaka vozlišča ali ceste, ki je zasedena. Vse ceste so usmerjene in vodijo od nekega vozlišča k drugemu. Njihove oznake so sestavljene iz oznak vozlišč, ki jih povezujejo. Npr. cesta 502 vodi od vozlišča 5 proti vozlišču 2. Pri cestah polna črta predstavlja zasedenost z vožnjo v smeri ceste, črtkana črta pa zasedenost z vožnjo v nasprotni smeri ceste.



Slika 1: Prikazan zemljevid ter začetni položaji, cilji in končne poti robotskih vozil iz primera, na katerem je opisano delovanje algoritma.



Slika 2: Časovna okna zasedenosti cest in vozlišč za modro in zeleno vozilo.

Levo krajišče črte predstavlja čas prihoda na začetek ceste, desno krajišče pa čas prihoda na konec ceste. Čakanje na cesti je določeno z dodatno označeno točko na črti, kjer čas od srednje točke do desnega krajišča predstavlja čas čakanja. Pri vozliščih čas prihoda na vozlišče predstavlja sredina črte, ki je razširjena z namenom, da vozila na vozlišče prihajajo z razmakom varnostnega časa. Izjema je poltrak oblike črta-pika, ki označuje čas od prihoda vozila na ciljno vozlišče naprej. Ta časovna okna algoritem upošteva pri iskanju poti rdečega vozila 3.

Postopek dodajanja vozlišč na seznama je prikazan v tabeli 1, kjer je *k* korak iterativnega postopka, *a* je oznaka vozlišča, *b* oznaka vozlišča predhodnika, c_{DS} cena-do-sem sestavljena iz treh delov in sicer cena-do-sem-brez-čakanja ($c_{DSBČ}$), cena-čakanja-na-predhodni-cesti ($c_{ČPC}$) in cena-čakanja-na-trenutni-cesti ($c_{ČTC}$), vsoto teh treh delov imenujmo skupna-cena-do-sem (c_{DSSk}), c_{DC} je ocena cene do cilja, c_{Sk} je skupna cena, ki predstavlja vsoto $c_{DSBČ}$, $c_{ČPC}$, $c_{ČTC}$ in c_{DC}, *cesta je* oznaka ceste, ki vodi do tega vozlišča, *d* je trenutni indeks poti, *e* predhodni indeks poti, *f* pa parameter umika. V stolpcu *g* je zabeležen korak, v katerem se to vozlišče prestavi na zaprti seznam. Tabela prikazuje samo dodana vozlišča in ne prikazuje celotnega delovanja algoritma. Med potekom algoritem računa cene in preverja zasedenosti tudi za vozlišča, ki jih kasneje ne doda na seznama. To je še posebej opazno pri koraku 3, ko algoritem naleti na konflikt in poizkuša z dodajanjem čakanja na večih mestih po poti od vozlišča nazaj, na odprti seznam pa na koncu doda samo eno vozlišče (glej korak 3, tabela 1).

2.1.1 Inicializacija

Najprej algoritem na odprti seznam doda začetno vozlišče 1, ki nima določenega predhodnega vozlišča in predhodne ceste. V tabeli 1 je to korak 0 (glej tabelo 1 (v nadaljevanju t1), *k*=0). Nato prične iterativni postopek iskanja poti.

2.1.2 Korak 1

V prvem koraku iz odprtega na zaprti seznam prestavi vozlišče 1 (glej k=1, zaprti seznam (v nadaljevanju ZS). Nato preveri zasedenost poti do edinega možnega naslednika – vozlišča 4 in ker je prosto, ga doda na odprti seznam. Njegova $c_{DSBČ}$ znaša 2s, $c_{ČTC}$ in $c_{ČPC}$ 0 s, c_{DC} pa 5s, indeksa poti in parameter umik so 0 (glej *k*=1, odprti seznam (v nadaljevanju OS)).

2.1.3 Korak 2

V drugem koraku algoritem pravkar dodano vozlišče 4 prestavi na zaprti seznam (glej t1, *k*=2, ZS). Nato doda prosti sosednji vozlišči 3 in 5 na odprti seznam (glej t1, *k*=2, OS). Za obe vozlišči je $c_{DSBČ}$ enaka 4, $c_{ČTC}$ in $c_{ČPC}$ 0s, indeksa poti in parameter umik so 0. Vrednost c_{DC} se med vozliščema razlikuje in sicer pri vozlišču 3 znaša 7s, pri vozlišču 5 pa 3s. Sosednjega vozlišča 1 ne dodaja, saj je le to z istim indeksom poti, a z manjšo c_{Sk} že na zaprtem seznamu. Prva dva koraka sta glede dodajanja vozlišč na seznama v osnovi enaka kot pri algoritmu A*, v naslednjem koraku pa

Tabela 1: Prikaz postopka iskanja poti vozila 3. V tabeli sta prikazana zaprti seznam in odprti seznam ter dodajanje vozlišč po korakih. Oznake v tabeli pomenijo: k – korak postopka, a – oznaka vozlišča, b – oznaka predhodnega vozlišča, c_{DS} – cena-do-sem sestavljena iz cene-do-sem-brez-čakanja, cene-čakanja-na-predhodni-cesti in cene-čakanja-na-trenutni-cesti, c_{DC} – cena-do-cilja, c_{Sk} – skupna cena, cesta – oznaka ceste, d – trenutni indeks poti, e – predhodni indeks poti, f – parameter umika, g – korak v katerem se vozlišče prestavi iz odprtega na zaprti seznam.

	ODPRTI SEZNAM										ZAPRTI SEZNAM								
k	*a*	*b*	c_{DS}	c_{DC}	c_{Sk}	*cesta*	*d*	*e*	*f*	*g*	*a*	*b*	c_{DS}	c_{DC}	c_{Sk}	*cesta*	*d*	*e*	*f*
0	1	0	[0,0,0]	7	7	0	0	0	0	→1.									
1	4	1	[2,0,0]	5	7	104	0	0	0	→2.	1	0	[0,0,0]	7	7	0	0	0	0
2	3	4	[4,0,0]	7	11	403	0	0	0	→4.	4	1	[2,0,0]	5	7	104	0	0	0
	5	4	[4,0,0]	3	7	405	0	0	0	→3.									
3	4	4	[2,0,7]	5	14	403	1	0	1	→5.	5	4	[4,0,0]	3	7	405	0	0	0
4											3	4	[4,0,0]	7	11	403	0	0	0
5	3	4	[11,0,0]	7	18	403	1	1	0		4	4	[2,0,7]	5	14	403	1	0	1
	5	4	[11,0,0]	3	14	405	1	1	0	→6									
6	2	5	[17,0,0]	5	22	502	1	1	0		5	4	[11,0,0]	3	14	405	1	1	0
	6	5	[14,0,0]	0	14	506	1	1	0	→7.									
7											6	5	[14,0,0]	0	14	506	1	1	0

pride do glavne razlike med algoritmoma, zato ga opišimo bolj podrobno.

2.1.4 Korak 3

V koraku 3 algoritem iz odprtega na zaprti seznam prestavi vozlišče 5 s predhodnikom 4, saj ima najmanjšo c_{Sk} (glej t1, k=3, ZS). Nadaljuje z dodajanjem vozlišč na odprti seznam.

Najprej kot trenutno vozlišče izbere vozlišče 4, ki je sicer prosto, a na zaprtem seznamu že obstaja enako vozlišče z enakim indeksom poti a z manjšo c_{Sk} , zato ga ne doda na odprti seznam.

Kot naslednje trenutno vozlišče v koraku 3 izbere vozlišče 6. Cesta 506 postane trenutna cesta, vozlišče 5 predhodno vozlišče in cesta 405 predhodna cesta. Algoritem izračuna $c_{DSBČ}$ trenutnega vozlišča 6 kot vsoto c_{DSSk} predhodnega vozlišča 5, ki znaša 4s in cene povezave med vozlišči 5 in 6, ki znaša 3s (glej sliko 2), skupaj torej 7s. Algoritem nato preveri zasedenost ceste 506, ki vodi do vozlišča 6, v času od predvidenega prihoda na cesto do predvidenega prihoda na konec ceste (4s-7s), ter samo vozlišče 6 v predvidenem trenutku prispetja nanj (7s). Iz slike 2 lahko razberemo, da je po cesti 506 v obratni smeri predvidena vožnja v časovnem intervalu (3s-6s). Cesta 506 je torej zasedena do trenutka 6s, kar predstavlja konflikt, kateremu se želimo izogniti.

S tem se začne rekurziven postopek v katerem algoritem s poizkušanjem išče primerno mesto za čakanje na sprostitev poti in izogib konfliktu. Najprej poizkusi s čakanjem na cesti, ki se na poti do zasedene ceste nahaja pred njo, torej je njena predhodna cesta (trenutno cesta 405). Če tudi ta cesta ni prosta, poizkusi s čakanjem na kateri od stranskih cest. Stranske cesta so vse ceste povezane s predhodnim vozliščem, razen predhodne in trenutne ceste (trenutno cesta 502). Če nobena od stranskih cest ni prosta za čakanje, se algoritem rekurzivno pomakne po poti nazaj. S tem predhodno vozlišče in cesta postaneta trenutno vozlišče in cesta, ki imata drugo predhodno vozlišče in cesto. Algoritem nadaljuje z novo iteracijo, kjer poizkusi s čakanjem na novi predhodni cesti, če ta ni prosta na novih stranskih cestah itd., dokler ne najde prostega mesta ali ne pride do začetka poti.

Nadaljujmo z opisom postopka algoritma na primeru. Še vedno smo pri koraku 3. Ker je cesta 506 zasedena, moramo prihod vozila na cesto zakasniti, kar storimo z določitvijo $c_{ČPC}$ vozlišča 6 na 2s. S tem podaljšamo čas čakanja na predhodni cesti in zakasnimo prihod na zasedeno cesto 506 na čas 6s, ko se ta sprosti. Ko dodajamo čakanje na predhodni cesti, moramo preveriti, če je ta v dodatnem času še prosta. Ko preverimo zasedenost ceste 405 v časovnem intervalu (2s-6s) (prihod na začetek ceste in zakasnjen prihod na konec ceste) opazimo, da je cesta prosta, a zasedeno je vozlišče 5 do časa 8s, kar še podaljša čakanje na cesti 405 za 2s, v tem dodatnem času pa cesta ni več prosta, torej podaljšanje čakanja ni mogoče. Naprej algoritem preveri, če bi lahko vozilo čakalo na sprostitev ceste 506 na stranski cesti predhodnega vozlišča 5. Vozlišče poleg trenutne ceste in predhodne ceste povezuje še eno cesto, cesto 502. To je stranska cesta predhodnega vozlišča 5. Preverimo zasedenost stranske ceste v času od prispetja na predhodno vozlišče 5 za čas čakanja 2s, torej v intervalu (4s-6s). Tudi cesta 502 ni prosta, umik na stransko cesto torej ni mogoč. Algoritem v nadaljevanju preizkusi vsa možna mesta čakanja po poti, ki vodi do trenutnega vozlišča, nazaj, z namenom, da najde mesto, kjer lahko brez oviranja drugih vozil čaka na sprostitev poti do željenega vozlišča. To naredi s pomočjo omenjenega rekurzivnega postopka pomika čakanja po poti nazaj. Ko najde prosto mesto, ponovno doda pripadajoče vozlišče na odprti seznam z novimi cenami čakanja, v tem primeru bo to vozlišče 4 (glej t1, k3, OS), a nismo še tam. Algoritem je sedaj izčrpal možnosti čakanja pred ali ob predhodnemu vozlišču 5, zato »se pomakne« po poti za eno vozlišče nazaj in še tam preizkusi mesta čakanja. Prej predhodno vozlišče 5 sedaj postane trenutno vozlišče, ki ima za predhodnika vozlišče 4. Ponovno preverimo zasedenost ceste 405 ob podaljšanem čakanju, torej v intervalu od (2s-6s). Kot lahko razberemo iz slike 2 je cesta v tem času prosta,

zasedeno pa je vozlišče 5 do 8s, kar podaljša čakanje na cesti 405 za 2s, ko pa cesta ni več prosta. Cesta je zasedena do časa 8s. Algoritem predlaga čakanje na predhodni cesti 104, da se cesta 405 sprosti in sicer dodatnih 6s. To se izvede s povečanjem $c_{ČPC}$ vozlišča 5 za 6s. Nato preveri zasedenost ceste 104 v podaljšanem času, torej v intervalu (0s-8s). Ponovno je cesta prosta, a zasedeno vozlišče 4 do časa 9s, kar podaljša čakanje na cesti 104 za 1s, ko pa ta ni več prosta. Čakanje na predhodni cesti vozlišča 5 torej ni mogoče. Algoritem preveri še možnost umika na stransko cesto vozlišča 4. Stranska cesta vozlišča 4 je v tem primeru cesta 403. Preverimo zasedenost na njej od prispetja na vozlišče 4 do sprostitve ceste 405, tj. na intervalu (2s-8s). Cesta je v tem času prosta, zasedeno pa je vozlišče 4 na katerega se vrnemo, in sicer do časa 9s. Ponovno preverimo zasedenost ceste na intervalu od (2s-9s) in tudi v tem intervalu je cesta prosta. Dodamo lahko torej umik na stransko cesto 403 iz vozlišča 4 za 7s. Na odprti seznam dodamo vozlišče 4 (glej t1, k=3, OS) z naslednjimi podatki; predhodno vozlišče 4, $c_{DSBČ}$ 2s, kar je enako času prvega prihoda na to vozlišče, $c_{ČPC}$ 0s, $c_{ČTC}$ 7s, c_{DC} 5s, c_{Sk} 14s, oznaka ceste 403, trenutni indeks poti 1, predhodni indeks poti 0 in parameter umika 1. S pravkar dodanim vozliščem algoritem še ni odprl poti do vozlišča 6, je pa ponudil možnost čakanja, da se zasedena pot do vozlišča 6 sprosti in v nadaljevanju najde pot do vozlišča brez konfliktov. Ker smo dodali novo vozlišče 4, z zakasnjenim prihodom nanj, ga mora algoritem v nadaljevanju dojemati ločeno, kar dosežemo z drugim indeksom poti. V nadaljevanju bomo pri dodajanju vozlišč, do katerih pridemo iz vozlišča 4 z zakasnjenim prihodom, ta primerjali le z vozlišči, ki imajo isti indeks poti.

Ostali smo pri raziskanem vozlišču 5 in še vedno smo pri koraku 3. Na odprti seznam želimo dodati še vozlišče 2. Algoritem preveri cesto 502, ki vodi do vozlišča, v intervalu (4s – 10s). Cesta je zasedena do časa 7s, zato predlaga čakanja na cesti 405 3s. Pri preverjanju zasedenosti ceste 405 v intervalu (2s – 7s) ugotovi, da je zasedena do časa 8s, torej čakanje na tej cesti ni mogoče. Preveri še možnost čakanja na stranski cesti vozlišča 5, to je v tem primeru cesta 506, a tudi ta ni prosta, zato se algoritem premakne po poti nazaj. Prej predhodno vozlišče 5 postane trenutno vozlišče s predhodnikom 4. Ponovno preveri zasedenost ceste 405 ob podaljšanem čakanju (4s – 7s) in ugotovi zasedenost ceste do časa 8s ter predlaga čakanje na predhodni cesti 104 do časa 8s, to je dodatnih 6s. Cesta 104 v tem času ni več prosta, zato preveri še možnost čakanja na stranski cesti 403, ki pa je v intervalu (2s – 9s), ko se sprosti poleg ceste 405 tudi vozlišče 4. Opazimo lahko, da nas je postopek pripeljal do enake možnosti umika kot pri dodajanju vozlišča 6. Ker bi na odprti seznam dodali vozlišče z enakimi $c_{DSBČ}$, $c_{ČPC}$ in $c_{ČTC}$ in enako oznako vozlišča in njegovega predhodnika, tega vozlišča ne dodajamo ponovno.

2.1.5 Korak 4

Smo pri koraku 4 in iz odprtega seznama na zaprti seznam, zaradi najmanjše $c_{Sk,}$ premaknemo vozlišče 3. Na odprti seznam lahko dodamo edino sosednje vozlišče 4, ki pa se z enakim indeksom poti 1 že nahaja na zaprtem seznamu, zato to vozlišče preskočimo in na odprti seznam v tem koraku ne dodajamo vozlišč.

2.1.6 Korak 5

V petem koraku na zaprti seznam prestavimo vozlišče 4 z indeksom poti 1 in predhodnim vozliščem 4, ki predstavlja umik na stransko cesto. Na odprti seznam najprej želimo dodati vozlišče 1, a pri preverjanju zasedenosti v intervalu (9s – 11s) ugotovimo, da čas konca zasedenosti vozlišča 1 ni znan, torej vozlišča ne moremo dodati. Pot do ostalih dveh vozlišč 3 in 5 je prosta, zato jih lahko dodamo na odprti seznam. Njune $c_{DSBČ}$ določimo kot vsota vseh delov c_{DS} predhodnega vozlišča 4, ki znaša 9s in cene povezave med predhodnim in trenutnim vozliščem, ki je pri obeh sosednjih vozliščih enaka 2s.

2.1.7 Korak 6

V naslednjem koraku 6 na zaprti seznam prestavimo vozlišče 5 s predhodnim indeksom poti 1. Na odprti seznam dodamo sosednji

vozlišči 2 in 6, sosednje vozlišče 4 pa se z enakim indeksom poti na zaprtem seznamu že nahaja.

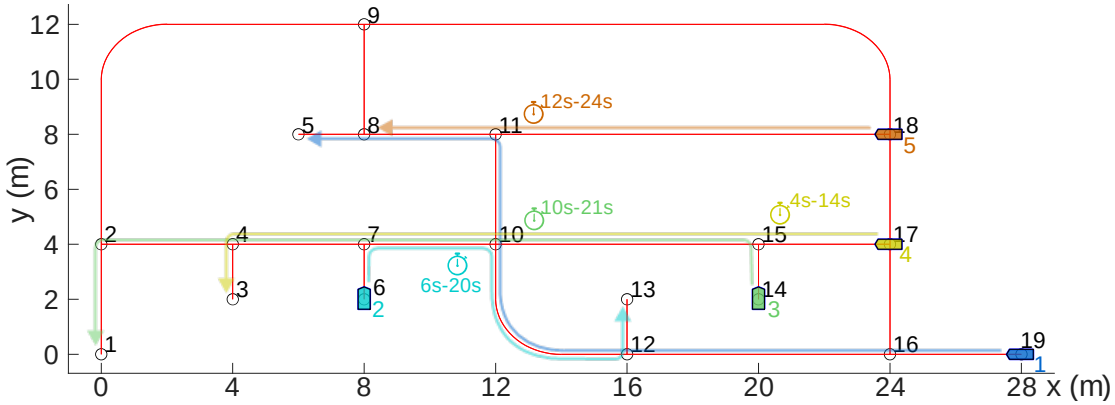
2.1.8 *Korak 7 in sestavljanje najdene poti*

V koraku 7 iz odprtega na zaprti seznam prestavimo vozlišče 6, ki je tudi ciljno vozlišče.

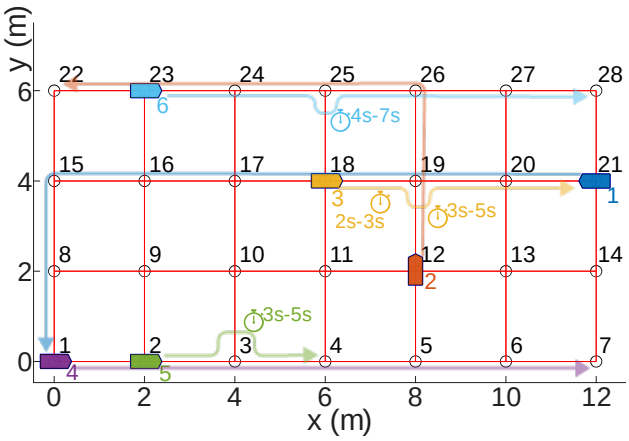
S tem je pot do cilja najdena, potrebno jo je še sestaviti. Najdeno pot sestavimo v obratni smeri od cilja do začetka s pomočjo podatka o predhodnem vozlišču in o predhodnem indeksu poti. Poleg cest je potrebno beležiti tudi podatek o umiku na stransko cesto in podatek o času čakanja na cesti. Če je za neko vozlišče določena vrednost $c_{ČTC}$, ta čas čakanja pripišemo cesti, ki pripada vozlišču. Vrednost $c_{ČPC}$ pa pripada času čakanja na predhodni cesti, torej jo upoštevamo pri naslednjem vozlišču. Ciljno vozlišče 6 ima oznako pripadajoče ceste 506, ki je zadnja cesta, ki vodi od začetka do cilja. Vrednost $c_{ČTC}$, $c_{ČPC}$ in parametra umika je 0. Predhodnik vozlišča 6 je vozlišče 5 in predhodni indeks poti je 1, na zaprtem seznamu iščemo torej vozlišče 5 z indeksom poti 1. Tudi to vozlišče ima vrednosti $c_{ČTC}$, $c_{ČPC}$ in parametra umika 0. Pripadajoča cesta tega vozlišča je cesta 405, ki jo dodamo na seznam cest, ki sestavljajo najdeno pot od začetka do cilja, pred cesto 506. Predhodnik vozlišča 5 je vozlišče 4 z indeksom poti 1. Pripadajoča cesta je 403 z oznako umika na stransko cesto in z vrednostjo $c_{ČTC}$ 7s. Dodamo jo na seznam cest, pri tem pa zabeležimo, da gre za umik na stransko cesto s časom čakanja 7s. Predhodni indeks poti vozlišča 4 je 0, njegov predhodnik pa vozlišče 4. Na zaprtem seznamu iščemo torej vozlišče 4 z indeksom poti 0. Temu vozlišču pripada cesta 104, ki jo dodamo na seznam cest. Vrednosti $c_{ČTC}$, $c_{ČPC}$ in parametra umika so 0. Predhodnik vozlišča 4 je vozlišče 1. To vozlišče nima predhodnika, to pomeni, da smo prišli do začetka in da seznam cest sestavlja celotno pot od začetka do cilja. Seznam cest, ki vodijo od začetka do cilja sestavljajo naslednje ceste: 104, 403 (umik=1, čas čakanja=7s), 405 in 506.

3 Rezultati

Na slikah 3 in 4 sta prikazana rezultata planiranja poti na dveh zemljevidih. Prikazani so začetni položaji vozil z njihovimi oznakami ter izris načrtanih poti. Vsa vozila vozijo s hitrostjo 1 m/s. Vozilo z oznako 1 ima najvišjo prioriteto in vozilo z najvišjo oznako ima najnižjo prioriteto. Čakanje je ponazorjeno z znakom ure s pripisom časa začetka in časa konca čakanja. Zemljevid na sliki 3 predstavlja skladišče z mesti dolaganja tovora na vozila (slepe ceste) ter z bolj obremenjenim osrednjim delom iz vozlišč 10 in 11. Zemljevid na sliki 4 predstavlja preprost a splošno uporaben mrežast zemljevid, ki ga lahko, z poljubno odstranitvijo posameznih cest ali vozlišč apliciramo na mnoga skladišča ali tovarne.



Slika 3: Prikaz rezultata planiranja na primeru 1. Izrisani so začetni položaji vozil in načrtane poti. Simbol ure s pripisom časa začetka in časa konca čakanja označuje mesto čakanja vozila.



Slika 4: Prikaz rezultata planiranja na primeru 2. Izrisani so začetni položaji vozil in načrtane poti. Simbol ure s pripisom časa začetka in časa konca čakanja označuje mesto čakanja vozila.

3.1 Analiza časovne potratnosti algoritma

Pri iskanju poti na zemljevidu s prostimi cestami in vozlišči, predlagani algoritem najde pot po popolnoma enakem postopku kot algoritem A*. Kadar pa pri iskanju poti naleti na zasedeno cesto ali vozlišče, poskuša z rekurzivnim postopkom najti prosto mesto za čakanje in izogib vozil, ki zasedajo ceste. Po dodanem čakanju ponovno dodaja že raziskana vozlišča, saj jih zaradi zakasnjenega prihoda dojema ločeno. Ti postopki povečajo časovno potratnost algoritma, kar smo testirali na mrežastem zemljevidu iz 100 vozlišč in 180 povezav, podobnemu kot na sliki 4. Rezultati testiranja so prikazani v tabeli 2. Za različno število vozil (3, 4, 5 in 10) smo v vsakem poizkusu naključno generirali možen scenarij začetnih in ciljnih položajev, v katerem je algoritem načrtal toliko poti, kolikor je vozil. Za vsako število vozil smo izvedli toliko poizkusov (glej tabela 2, stolpec 2), da je bilo število vseh načrtanih poti enako 6000. V tretjem stolpcu tabele je prikazan čas iskanja za vseh 6000 poti, v četrtem stolpcu pa povprečni čas iskanja ene poti. Opazimo lahko, da je pri večjem številu vozil skupni čas iskanja poti večji kot pri manjšem številu vozil, kljub temu, da se skupno število iskanj ne spremeni. To pojasni dejstvo, da pri večjem številu vozil, vozila z nižjo prioriteto pri iskanju večkrat naletijo na zasedene dele zemljevida, kar poveča potratnost algoritma.

Tabela 2: Analiza časovne potratnosti algoritma

Št. vozil	Št. poizkusov	Št. načrt. poti	Čas [s]	Povprečni čas na najdeno pot [ms]
3	2000	6000	264,7	44,1
4	1500	6000	287,5	47,9
5	1200	6000	290,3	48,4
10	600	6000	328,1	54,7

4 Zaključek

V članku smo predlagali nov algoritem za planiranje poti več robotskih vozil, ki temelji na algoritmu A*. Algoritem upošteva prioritete transportnih nalogov, kar omogoča bolj učinkovito izvajanje prioritetnih zadolžitev, vozila z nižjo prioriteto pa se prilagodijo prvim. Strategija določanja časovnih oken zasedenosti omogoča zaznavo potencialnih konfliktov, ki se jim algoritem izogne z izbiro druge poti, s čakanjem pred konfliktom ali z umikom na stransko pot.

Kompleksnost algoritma bi lahko prinesla preveliko kombinatorično zahtevnost, zato smo analizirali učinkovitost oz. časovno potratnost algoritma.

5 Literatura

[1] R. Olmi, C. Secchi and C. Fantuzzi (2008). Coordination of multiple AGVs in an industrial application. In IEEE International Conference on Robotics and Automation, pp. 1916-1921, 2008.

[2] E. Gawrilow, E. Köhler, R. Möhring and B. Stenzel. (2008). Dynamic Routing of Automated Guided Vehicles in Real-time. 10.1007/978-3-540-77203-3_12.

[3] S. Maza and P. Castagna. Conflict-free AGV routing in bi-directional network. In ETFA 2001, 8th International Conference on Emerging Technologies and Factory Automation, vol.2, pp. 761-764, 2001.

[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, no. 3, pp. 922-934, 2015.

[5] C. Lee, B. Lin, K.K.H. Ng, Yaqiong Lv and W.C. Tai. Smart robotic mobile fulfillment system with dynamic conflict-free strategies considering cyber-physical integration. Advanced Engineering Informatics, vol.42, pp. 100998, 2019.

[6] Z. Zhang, Q. Guo, J. Chen and P. Yuan. Collision-Free Route Planning for Multiple AGVs in an Automated Warehouse Based on Collision Classification. In IEEE Access, vol. 6, pp. 26022-26035, 2018.

[7] P. E. Hart, N. J. Nilsson and B. Raphael. A formal basis for the heuristic determination of minimum cost paths, IEEE Transactions on Systems Science and Cybernetics, vol. 4, no. 2, str. 100–107, 1968.