Raport ştiinţific – proiect PN-III-P1-1.1-TE-2016-0737, nr. 32/2018

Etapa 2 – Soluţii algoritmice şi experimente preliminare

Activitatea 2.1. Algoritmi de planificare pentru specificaţii de mişcare şi efectuare de acţiuni

Alocarea optimă de sarcini pentru îndeplinirea unui obiectiv asumat de o echipă de agenţi mobili poate fi realizată în diferite moduri. În cel mai simplu caz, obiectivul echipei se îndeplineşte prin minimizarea sumei tuturor costurilor agenţilor individuali. Cu toate acestea, în funcţie de modul în care sunt definite costurile agenţilor, acest lucru ar putea să nu fie mereu de dorit. De exemplu, să presupunem cazul în care costul indică timpul necesar executării sarcinilor atribuite agentului respectiv. Cum agenţii pot evolua în paralel, timpul total necesar pentru finalizarea misiunii complete nu este dat de suma costurilor individuale. Acest timp reprezintă costul maxim dintre agenţi [1].

Planificarea mişcării pentru sisteme în aplicaţii reale implică restricţii suplimentare de resurse [2-4]. O echipă de agenţi mobili trebuie să ţină cont de aceste restricţii în avans pentru a putea realiza planificarea mişcării [5]. De exemplu, dacă sunt necesare resurse externe pentru unele dintre sarcini, agenţii trebuie să se coordoneze nu numai în alocarea sarcinilor, ci și alocarea acestor resurse pentru a evita conflictele din planurile de execuție pentru sarcinile individuale.

Problema planificării și controlului sigur pentru echipe de drone prezintă o importanță critică, pe măsură ce domeniul de activitate atribuit acestor sisteme crește. În [6] autorii prezintă o abordare pentru a rezolva această problemă pentru misiuni multi-quadrotor. Având în vedere o misiune exprimată în Semnal Logic Temporal (STL), abordarea generează traiectorii cu robusteţe maximă pentru dronele care satisfac specificația STL în timp continuu. De asemenea,

se demonstrează că restricțiile incluse în optimizare garantează că aceste traiectorii pot fi urmărite aproape perfect de drone folosind sisteme comerciale de control ale poziţiei şi orientării. Soluţia propusă evită simplificarea excesivă a abstractizării întâlnită în multe metode de planificare, păstrând în același timp expresivitatea misiunilor codificate în STL permițând rezolvarea de cerințe complexe spațiale, temporale și reactive. Prin experimente, atât în simulare cât și pe sisteme reale, soluţia propusă arată performanța, scalabilitatea și aplicabilitatea în timp real. Implementarea soluţiei propuse a fost evaluată pentru o echipă de două drone Crazyflie2.0. În definirea problemei de planificare nu au fost incluse în mod distinct acţiuni pe care echipa să le îndeplinească în paralel cu mişcarea.

Proiectarea algoritmilor de planificare cu specificaţii de nivel înalt pentru mişcare şi efectuare de acţiuni reprezintă un subiect de interes în domeniul roboticii. În literatură sunt descrise soluţii pentru mobilitate şi acţiuni doar pentru un agent, în timp ce pentru echipe de agenţi mobili se consideră în special planificarea mişcării. În funcţie de formularea problemei au fost propuse un număr foarte redus de probleme de planificare şi alocare simultană de sarcini pentru echipe de agenţi mobili. În [7,8] autorii propun o soluţie pentru următoarea problemă de tip STAP (Simultaneous Task Allocation and Planning):

Se consideră o descriere de nivel înalt pentru o misiune specifică, o echipă de roboţi mobili cu roţi, costuri pentru acţiuni, resurse iniţiale şi restricţii. Să se găsească o succesiune de acţiuni care să conducă la îndeplinirea misiunii, minimizând costul maxim pe fiecare robot.

Construirea soluţiei porneşte de la extinderea algoritmului Martins [9] a cărui principiu este similar căutării Dijkstra pentru o cale minimă de îndeplinire a unui singur obiectiv. Când se generalizează căutarea căii minime de la un singur obiectiv la probleme multi-obiectiv, pot exista mai multe soluţii pentru a atinge un anumit scop, fiecare putând fi optimă în funcție de ce obiectiv este considerat a fi mai important. O soluţie nu poate fi optimă doar dacă ea este cel mult la fel de bună ca o altă soluţie sub toate criteriile obiectivului. Această noțiune este denumită în mod obișnuit dominanţă Pareto.

Soluţia propusă nu utilizează direct graful de stări, ci consideră pentru fiecare stare un set de etichete ale căror vectori de costuri sunt Pareto optimali. Astfel algoritmul Martins găseşte toate traiectoriile Pareto optimale descrise de aceste etichete. Rezultatele experimentale sunt obţinute din studii de caz efectuate pe o echipă de trei roboţi mobile cu roţi. Spaţiul experimental este modelat pe o schiţă a unui etaj de hotel.

Din punct de vedere al limitărilor, soluţia propusă necesită descompunerea misiunii în submisiuni ce pot fi traduse în formule LTL locale executabile în paralel. Acest fapt nu poate fi îndeplinit în toate formulările LTL. În acelaşi timp, soluţia este valabilă doar pentru formulări LTL „co-safe” [10] (care nu includ sarcini definite de operatorul “always”).

In etapa curenta s-a considerat un task LTL ce exprima cerintele pentru o echipa de roboti identici. Task-ul este exprimat in functie de un set de actiuni ce pot fi efectuate in anumite regiuni de interes, iar scopul este de a gasi in mod automat o strategie pentru echipa de roboti astfel incat acestia sa coopereze pentru indeplinirea misiunii. Pentru aceasta problema, bazele abordarii inovative sunt formulate in articolul [5], ce foloseste abstractii bazate pe retele Petri si formulari in termenii unor probleme de optimizare. Aceste instrumente produc o solutie fezabila computational, spre deosebire de abordarile ce folosesc abstractizari bazate pe grafuri sub forma sistemelor de tranzitie sau automate. Similar abordarii [5] definim un sistem RMPN (Robot Motion Petri Net) ce poate modela intreaga echipa de roboti, cu pastrarea unei topologii fixe, indiferent de numarul de roboti. Problema calcularii unei secvente de executie pentru a atinge o stare (marcaj) finala dorita si a produce o observatie dorita este rezolvata printr-o procedura algoritmica bazata pe doua probleme de optimizare.

Algoritmul iterativ descris in [5] este utilizat pentru a ghida selectia unei rulari acceptate intr-un automat Büchi, rularea optimizarii unei (sau doua) probleme de programare intreaga mixta (MILP), si, in cele din urma, pentru a construi secvente de executie in modelul RMPN ce produc planificari individuale pentru roboti, cu garantia satisfacerii specificatiei date. Desi problemele MILP sunt incluse in clasa de complexitate NP-hard (deci si complexitatea computationala a metodei propuse este NP-hard), prin abordarea propusa se pot rezolva relativ repede unele instante ce nu sunt fezabile pentru metodele bazate pe produse sincrone de modele robotice individuale.

Principala noutate fata de metoda propusa in [5] consta in introducerea in specificarea misiunii pentru echipa de roboti a unui set de actiuni ce trebuie executate de aceasta. Astfel, specificatia LTL este furnizata peste setul de actiuni. Pentru a integra aceasta functionalitate in abstractiile bazate pe Retele Petri si in formularea problemelor de optimizare, propunem definirea modelului RPMN peste un set de regiuni de interes ce au asociate actiuni din setul peste care este definita specificatia LTL. Tranzitiile in reteaua Petri ce modeleaza sistemul corespund atat capabilitatilor de miscare ale robotilor intre regiuni, cat si celor de efectuare a actiunilor asociate.

Presupunem o echipa de R roboti identici ce se deplaseaza intr-un mediu rectangular. Mediul este partitionat intr-un set finit de celule disjuncte folosind o metoda de descompunere precum [12]. O celula a mediului de lucru este notata cu $c_j, j = 1,\ldots,|C|$ unde C reprezinta setul de celule, iar $|C|$ denota cardinalul setului C. In plus, presupunem un set finit de propozitii atomice $\Pi = \{\pi_1, \pi_2, \ldots, \pi_{|\Pi|}\}$. Intr-un scenariu robotic, π_i denota o actiune specifica. O actiune π_i corespunde uneia sau mai multor celule ale mediului de lucru (poate fi efectuata in una sau mai multe celule). Daca cel putin un robot efectueaza actiunea π_i in una din celulele corespunzatoare, spunem ca propozitia π_i este satisfacuta (este Adevarata). Setul Π va fi utilizat pentru furnizarea formulei LTL ce defineste misiunea ce trebuie indeplinita de echipa de roboti.

Pentru a permite formularea unei solutii pentru misiunea specificata ce presupune efectuarea de actiuni introducem urmatoarele ipoteze si notatii aditionale: Fie C^* un subset al C, unde $c^*_j, j \in [1,|C|]$ reprezinta o regiune de interes. Prin regiune de interes intelegem o celula din setul C care indeplineste cel putin una din urmatoarele conditii: este ocupata de cel putin un robot in starea initiala, are asociata o actiune din setul Π.

O actiune π_i din setul Π poate fi efectuata intr-un subset $C_i^* = \{c_j^*, j \in [1, |C^*|]\}$ al C^*. Definim setul P format din elemente $p_k, k = 1, \ldots, |P|$, unde $p_k = (\pi_i, c_j^*)$ a.i. $\pi_i \in \Pi \cup \{\emptyset\}$ si $c_j^* \in C_i^*$. Prin $\emptyset$ intelegem actiunea de vizitare a unei celule. Setul P va include $\sum_{i=1}^{|\Pi|} |C_i^*|$ elemente corespunzatoare regiunilor de interes in care se pot efectua actiuni din Π, si $|C^*|$ elemente corespunzatoare tuturor regiunilor de interes pentru care se asociaza actiunea de vizitare.

Definitie 1. Un sistem RMPN (Robot Motion Petri Net) este un 4-uplu $\mathscr{Q} = \langle \mathscr{N}, m_0, \Pi, h \rangle$, unde:

- $\mathscr{N} = \langle P, T, Post, Pre \rangle$ este o structura de retea Petri cu P setul de pozitii; T reprezinta setul de tranzitii ce modeleaza capabilitatile de miscare ale robotilor intre celule si de efectuare de actiuni; $Post \in \{0,1\}^{|P| \times |T|}$ reprezinta matricea post-incidenta (post-conditii) ce defineste arcele ce unesc o tranzitie cu o pozitie; si $Pre \in \{0,1\}^{|P| \times |T|}$ matricea pre-incidenta (pre-conditii) ce defineste arcele ce unesc o pozitie cu o tranzitie. Spre ex., $Post[p,t] = 1$ daca exista un arc ce conecteaza tranzitia $t \in T$ cu pozitia $p \in P$, altfel $Post[p,t] = 0$ si $Pre[p,t] = 1$ daca exista un arc ce conecteaza pozitia $p \in P$ cu tranzitia $t \in T$, altfel $Pre[p,t] = 0$;

- $\forall t \in T$, $1^T \cdot Pre[\cdot,t] = 1$ si $1^T \cdot Post[\cdot,t] = 1$, unde $1 \in \{1\}^{|P|}$ este un vector cu toate elementele egale cu unu si $Pre[\cdot,t]$ este coloana corespunzatoare tranzitiei t in matricea Pre. Aceasta conditie implica faptul ca toate tranzitiile au o singura pozitie la intrare si o singura pozitie la iesire (in teoria retelelor Petri, modelul este denumit *masina de stare*);

- m_0 denota marcajul initial, unde $m_0[p]$ este numarul de jetoane egal cu numarul de roboti aflati initial in regiunea $p \in P$ (mai precis, asociati cu perechea regiune-actiune vida $p \in P$);

- $\Pi \cup \{\emptyset\}$ reprezinta alfabetul de iesire (set ce contine simbolurile de iesire posibile), unde $\emptyset$ denota observatia vida (in termenii efectuarii de actiuni, $\emptyset$ corespunde vizitarii unei celule fara efectuarea unei alte actiuni);

- $h : P \to 2^\Pi$ realizeaza maparea observatiilor, unde 2^Π este setul tuturor subseturilor lui Π, incluzand setul vid $\emptyset$, iar $h(p_i)$ furnizeaza iesirea pozitiei $p_i \in P$. Daca p_i contine cel putin un jeton, atunci propozitiile din $h(p_i)$ sunt active (satisfacute). ■

Spre deosebire de metoda prezentata in [5], setul P de pozitii ale retelei Petri nu are o corespondenta unu-la-unu cu setul de celule din mediul de lucru. Deoarece intr-o celula pot fi executate mai multe actiuni, aceasta va avea cate o instanta pentru fiecare din aceste actiuni, precum si o instanta asociata actiuni de vizitare ($\emptyset$). Pentru a simplifica structura retelei Petri, celulele care nu au asociate actiuni sau nu reprezinta pozitii initiale ale robotilor nu au pozitii corespondente in P. Tranzitiile T modeleaza miscarea unui robot dintr-o celula in alta apartinand setului C^*, abstractizand (dar nu ignorand) drumul fizic ce trebuie parcurs (acesta poate contine celule ce nu apartin setului C^*). Astfel, o tranzitie va corespunde drumului minim intre doua celule din C^*, calculat pe baza grafului total al mediului de lucru.

Pentru $p \in P$, seturile sale de tranzitii de intrare si de iesire se noteaza ${}^\bullet p = \{t \in T | Post[p,t] > 0\}$ si $p^\bullet = \{t \in T | Pre[p,t] > 0\}$, respectiv. Pentru $t \in T$, setul sau de pozitii

de intrare si de iesire se noteaza ${}^\bullet t = \{p \in P | Pre[p,t] > 0\}$ si $t^\bullet = \{p \in P | Post[p,t] > 0\}$, respectiv. O tranzitie $t_j \in T$ este validata in m daca toate pozitiile de intrare contin cel putin un jeton, adica $\forall p_i \in {}^\bullet t_j, m[p_i] \geq 1$. O tranzitie validata t_j se poate declansa (executa), conducand la o noua stare $\tilde{m} = m + C[\cdot, t_j]$, unde $C = Post - Pre$ reprezinta matricea de evolutie a jetoanelor, iar $C[\cdot, t_j]$ este coloana ce corespunde t_j. Spunem ca $\tilde{m}$ este un marcaj fezabil ce poate fi atins din m prin declansarea t_j si scriem $m[t_j\rangle\tilde{m}$.

Intr-un sistem RMPN, declansarea unei tranzitii t corespunde miscarii unui robot din ${}^\bullet t = \{p_i\}$ in $t^\bullet = \{p_j\}$ si executarii actiunii asociate cu p_j. Se observa ca, prin definitie, fiecare tranzitie are o singura pozitie de intrare si o singura pozitie de iesire, obtinand astfel o masina de stare [13]. Daca $\tilde{m}$ poate fi atinsa din m prin intermediul unei secvente finite de tranzitii $\sigma = t_{i_1} t_{i_2} \dots t_{i_k}$, este satisfacuta urmatoarea ecuatie de stare (fundamentala):

$$\tilde{m} = m + C \cdot \sigma, \tag{1}$$

unde $\sigma \in \mathbb{N}_{\geq 0}^{|T|}$ este vectorul numarului de executari ale t_j in secventa σ. Se observa ca ecuatia (1) reprezinta doar o conditie necesara pentru atingerea unui marcaj. Solutiile (1) ce reprezinta marcaje ce nu pot fi atinse se numesc *marcaje false*.

O retea Petri este viabila daca, indiferent de marcajul ce a fost atins, este posibil ca, în continuare, să fie executată orice tranziţie a reţelei (posibil dupa executarea prealabila a altor tranzitii). Intr-o masina de stare viabila nu exista marcaje false [14], adica solutiile ecuatiei fundamentale (1) reprezinta setul de marcaje ce pot fi atinse.

Fie $v_i \in \{0,1\}^{1 \times |P|}$ vectorul linie caracteristic al observatiei $\pi_i \in \Pi$ a.i. $v_i[p_k] = 1$ daca $\pi_i \in h(p_k)$ si $v_i[p_k] = 0$ altfel. Se observa cu usurinta ca, pentru un marcaj m ce poate fi atins, daca produsul $v_i \cdot m > 0$, atunci observatia π_i este activa in m

Fie $V \in \{0,1\}^{|\Pi| \times |P|}$ matricea formata de vectorii caracteristici ai tuturor observatiilor (prima linie este vectorul caracteristic al π_1, etc.). Produsul $V \cdot m$ este un vector coloana de dimensiune $|\Pi|$ in care elementul i^{th} este diferit de zero daca observatia π_i este activa. Notam cu $||V \cdot m||$ setul de iesiri corespunzatoare elementelor diferite de zero ale $V \cdot m$, adica $||V \cdot m||$ reprezinta setul de observatii active (element al 2^Π) in marcajul m.

Exemplul 1. Consideram mediul de lucru din Fig. 1 constand din 6 celule $\{c_1, \dots, c_6\}$, doi roboti identici localizati initial in c_3 si c_6, precum si trei actiuni $\{\pi_1, \pi_2, \pi_3\}$ astfel incat actiunea π_1 corespunde celulelor c_1 si c_2, actiunea π_2 corespunde celulelor c_2 si c_3, in timp ce actinunea π_3 poate fi executata in celula c_4.

Astfel, subsetul regiunilor de interes este $C^* = \{c_1, c_2, c_3, c_4, c_6\}$, iar setul P consta din 7 elemente definite astfel:

$$\begin{array}{l|l} p_1 : (\pi_1, c_1) & p_5 : (\pi_3, c_4) \\ p_2 : (\pi_1, c_2) & p_6 : (\emptyset, c_3) \\ p_3 : (\pi_2, c_2) & p_7 : (\emptyset, c_6) \\ p_4 : (\pi_2, c_3) & \end{array}$$

Setul de tranzitii este reprezentat de $T = \{t_1, \dots t_{30}\}$, ce contine toate tranzitiile intre oricare doua pozitii distincte din setul P, mai putin cele inspre p_6 si p_7 ce au asociata actiunea

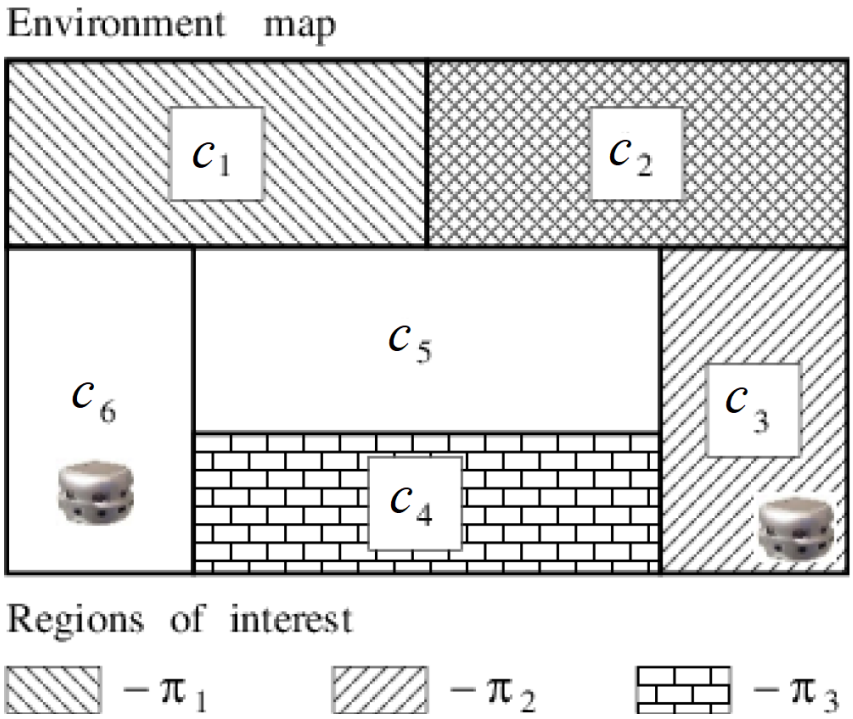


Fig. 1: Harta descompunerii pe regiuni a mediului de lucru in exemplul considerat.

$\emptyset$. Spre exemplu, tranzitia intre p_1 si p_3 are semnificatia deplasarii unui robot intre celulele c_1 si c_2 si efectuarea in c_2 a actiunii π_2. Tranzitia inversa, intre p_3 si p_1 semnifica deplasarea unui robot intre c_2 si c_1 si efectuarea actiunii π_1 in celula c_1. Tranzitia intre p_1 si p_4 echivaleaza cu deplasarea unui robot intre c_1 si c_3, pe drumul cel mai scurt, si efectuarea π_2 in celula c_3.

Marcajul initial al sistemului este $m_0 = [0,0,0,0,0,1,1]^T$, alfabetul de iesire este $\Pi = \{\pi_1, \pi_2, \pi_3\}$ iar functia observatiilor: $h(p_1) = h(p_2) = \{\pi_1\}$, $h(p_3) = h(p_4) = \{\pi_2\}$, $h(p_5) = \{\pi_3\}$, $h(p_6) = h(p_7) = \emptyset$.

Vectorul caracteristic al lui π_1 este $v_1 = [1,1,0,0,0,0,0]$ avand in vedere ca iesirea π_1 poate fi observata in p_1 si p_2, $v_2 = [0,0,1,1,0,0,0]$, iar $v_3 = [0,0,0,0,1,0,0]$. Astfel, $V = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$.

Deoarece $V \cdot m_0 = [0,0,0]^T$, nicio observatie din Π nu este activa in m_0. ■

O *traiectorie* in $\mathscr{Q}$ este o *secventa infinita* $r_{\mathscr{Q}} = m_0[t_{j_1}\rangle\, m_1[t_{j_2}\rangle\, m_2[t_{j_3}\ldots$ ce induce un *cuvant* de iesire, ce reprezinta secventa observata de elemente din 2^{Π}.

Misiunea de miscare si efectuare de actiuni pentru echipa de roboti este furnizata ca formula dintr-o subclasa LTL (Linear Temporal Logic), notata LTL_{-X}. Informal, o formula LTL_{-X} este definita recursiv peste setul de propozitii atomice Π, folosind operatorii Booleeni standard ($\neg$ - negatie, $\vee$ - disjunctie, $\wedge$ - conjunctie, $\Rightarrow$ - implicatie, si $\Leftrightarrow$ - echivalenta) si operatori temporali ($\mathscr{U}$ - until, $\Diamond$ - eventually, si $\Box$ - always). Pentru simplitatea notatiilor, in continuare vom scrie LTL in loc de LTL_{-X}.

Orice formula LTL definita peste setu; Π poate fi transformata intr-un automat Büchi (vezi Def. 2) ce accepta toate si numai siruri de intrare ce satisfac formula [15]. Sunt disponibile instrumente software ce permit astfel de conversii, de ex. [16, 17, 18].

Definitie 2. Automatul Büchi ce corespunde unei formule LTL peste setul Π are structura $B = (S, S_0, \Sigma_B, \rightarrow_B, F)$, unde:

- S este un set finit de stari;
- $S_0 \subseteq S$ este setul de stari initiale;
- $\Sigma_B = 2^{\Pi}$ este setul de intrari;
- $\rightarrow_B \subseteq S \times \Sigma_B \times S$ reprezinta relatia de tranzitie;
- $F \subseteq S$ ieste setul starilor finale. ■

Pentru $s_i, s_j \in S$, notam cu $\rho(s_i, s_j)$ setul tuturor intrarilor lui B ce activeaza tranzitia de la s_i la s_j. Tranzitiile in B pot fi nedeterministe, cu semnificatia ca, dintr-o anumita stare pot exista tranzitii de iesire multiple activate de o aceeasi intrare, adica putem avea $(s, \tau, s') \in \rightarrow_B$ si $(s, \tau, s'') \in \rightarrow_B$, cu $s' \neq s''$. Astfel, o secventa de intrare poate produce mai mult de o secventa de stari.

Un cuvant de intrare de lungime infinita (secventa cu elemente din Σ_B) este *acceptat* de B daca acesta produce cel putin o secventa de stari (*rulare*) ale lui B care viziteaza setul F infinit de des. Concomitent, o rulare a lui B ce viziteaza infinit de des setul F este *acceptata*, si, daca B are cel putin o rulare (infinita) acceptata, atunci, are cel putin o rulare cu o reprezentare prefix-sufix finita [15]. Astfel, rularea acceptata (si cuvantul de intrare corespunzator) poate fi stocata in memoria finita sub forma: (i) un sir de lungime finita denumit *prefix* ce conduce B intr-o stare finala din F, si (ii) un sir de lungime finita denumit *sufix* ce conduce B inapoi in starea finala atinsa de prefix. Rularea lui B este formata din prefix urmat de o infinitate de repetitii ale sufixului, adica *prefix, sufix, sufix,* .... Astfel, ne vom concentra pe gasirea unor rulari ale modelului RMPN $\mathscr{Q}$ cu o structuraprefix-sufix similara care genereaza secvente ce satisfac formulele LTL.

Formularea problemei. *Data fiind o echipa R de roboti identici, un mediu de lucru descompus in setul de celule C, un set de actiuni* Π *si maparea acestora pe celulele din C, precum si o misiune sub forma unei formule LTL peste setul* Π*, gaseste o strategie de miscare si de efectuare a actiunilor de catre echipa de roboti pentru a satisface misiunea data.*

Solutie. Solutia propusa consta din doua etape:

1) Constructia sistemului RMPN plecand de la ipotezele problemei (R, C, Π)
2) Gasirea unei secvente de observatii ce satisface formula LTL si generarea tranzitiilor RMPN ce produc acea secventa

In **prima etapa** se construieste graful celulelor ce formeaza mediul de lucru. Metodele de descompunere in celule partitioneaza mediul in regiuni convexe denumite celule [19]. In urma descompunerii se obtine un graf de conectivitate, in care nodurile reprezinta celulele, iar arcele dintre noduri ilustreaza relatia de adiacenta a acestora.

Definirea setului Π presupune suplimentar si specificarea relatiei dintre fiecare element din Π si celulele grafului de conectivitate. Pe baza acestei relatii si a pozitionarii initiale a robotilor se deduce setul C^* al regiunilor de interes. Pentru acest set se construieste un nou graf de conectivitate in care nodurile sunt reprezentate de regiunile de interes, iar un arc intre doua noduri corespunde drumului cel mai scurt intre cele doua regiuni de interes, determinat din graful de conectivitate initial. Pentru fiecare arc se stocheaza secventa de noduri intermediare corespunzatoare drumului cel mai scurt in graful de adiacenta initial.

Modelul RMPN $\mathscr{Q}$ se construieste definind componentele sale astfel:

1. P: Pentru fiecare regiune de interes, se creeaza o pozitie pentru fiecare actiune ce poate fi efectuata in acea regiune. Pentru regiunile de interes corespunzatoare plasarilor initiale ale robotilor se genereaza pozitii cu actiunea $\emptyset$.

2. T: Intre fiecare doua pozitii din P se defineste o tranzitie, mai putin spre pozitiile ce au asiciata actiunea $\emptyset$. Semnificatia trecerii unui jeton printr-o tranzitie este ca un robot se deplaseaza din celula pozitiei de intrare in celula pozitiei de iesire si efectueaza actiunea asociata pozitiei de iesire. Un aspect important il reprezinta faptul ca cele doua celule asociate pozitiilor de intrare si iesire ale tranzitiei pot fi neadiacente, caz in care tranzitia corespunde unei deplasari pe drumul cel mai scurt intre cele doua celule. De asemenea, celulele pot fi identice, caz in care tranzitia corespunde doar efectuarii actiunii pozitiei de iesire.

3. h: Functia observatiilor va furniza actiunea mapata pe fiecare pozitie, conform datelor de intrare.

Costul unei tranzitii intre doua pozitii este format din costul asociat deplasarii pe drumul cel mai scurt intre celulele corespunzatoare si din costul asociat efectuarii actiunii pozitiei de iesire. Daca actiunea asociata pozitiei de iesire este $\emptyset$, atunci al doilea termen este nul.

Solutia propusa pentru **a doua etapa** se bazeaza pe abordarea descrisa in [5] si consta in 3 pasi principali:

(i) O rulare acceptata r este aleasa din automatul Büchi corespunzator formulei LTL φ;

(ii) Pentru fiecare tranzitie a rularii r, se cauta o secventa de declansari pentru modelul RMPN astfel incat observatiile generate sa produca tranzitia aleasa;

(iii) Strategiile de miscare si efectuare de actiuni ale robotilor sunt obtinute concatenand secventele de declansari din pasul (ii) si impunand momente de sincronizare intre aceste secvente.

Algoritmul 1 descrie constructia iterativa a solutiei conform pasilor de mai sus.

Algorithm 1: Iterative construction of solution

Input: RMPN model $\mathcal{Q}$, Büchi automaton B, R, κ

Output: Solution (firing sequence and synchronizations for each robot)

Run Alg. 2 to find set Γ containing accepted runs;
while $\Gamma \neq \emptyset$ **do**
    Initialize robot sequences as empty and m_0 as initial RMPN marking;
    Pick shortest run $r \in \Gamma$, $r = s_0 s_1 \dots s_p \dots s_{L_r}$;
    for $j = 0, 1, \dots, L_r - 1$ **do**
        Formulate and solve MILP (4);
        if σ *is not spurious* **then**
            Run Alg. 4 to establish if σ is applicable;
            if *solution* σ *is applicable* **then**
                Update robot sequences;
                continue with next j in for loop;
            else
                Formulate and solve MILP (5);
                if *solution* σ *is not spurious* **then**
                    Update robot sequences;
                    continue with next j in for loop;
                else
                    Current transition of r cannot be ensured;
                    break the for loop;
        else
            Current transition of r cannot be ensured;
            break the for loop;
    if *all transitions of* r *were ensured* **then**
        In the obtained robot sequences, keep repeating the last part ensuring suffix $s_{p+1} \dots s_{L_r}$;
        return solution;
    else
        $\Gamma := \Gamma \setminus \{r\}$;

Pasul (i): Automatul Büchi B corespunzator formulei φ este construit folosind instrumente software existente, ex. [17, 18]. Inainte de a cauta rulari acceptate, se pastreaza pe tranzitiile lui B doar intrarile ce pot fi generate de modelul $\mathscr{Q}$. Acest lucru este realizat in primele patru linii ale Alg. 2, in care initial se construieste setul $O \subseteq 2^{\Pi}$ de observatii (iesiri) ce pot fi generate de un robot (linia 1). Avand in vedere ca $\mathscr{Q}$ contine R jetoane, setul de iesiri $H \subseteq 2^{\Pi}$ ce poate fi generat de intreaga echipa se obtine ca produs cartezian al O cu sine de R ori (linia 2). Pentru fiecare tranzitie a lui B, setul de intrari ce activeaza acea tranzitie este actualizat pe baza setului H deoarece modelul $\mathscr{Q}$ poate genera cel mult aceste iesiri (liniile 3-4). Se observa ca unele tranzitii ale B pot disparea dupa rularea Alg. 2.

In continuare, se construieste un set Γ ce contine un numar finit de rulari acceptate ale lui B. Pentru aceasta, se considera un numar mic $\kappa \in \mathbb{N}$ (de obicei $\kappa \leq 4$) si liniile 5-14 ale Alg. 2. Traseele in B cu structura prefix-sufix sunt gasite prin rularea unor cautari folosind algoritmul k-shortest path [20] pe graful de adiacenta corespunzator tranzitiilor lui B. Din fiecare stare initiala s_0 a lui B sunt gasite cel mult κ cele mai scurte trasee catre fiecare stare finala s_f. Aceste trasee sunt prefixe (liniile 7-8). Trebuie remarcat ca algoritmul de cautare nu include cicluri in traseele returnate, astfel este posibil sa se obtina mai putin de κ trasee intre doua noduri date. In liniile 9-12 se construieste un set de trasee ce ar aduce B inapoi in starea finala s_f. Setul intermediar S_{s_f} contine stari ce pot conduce la s_f. Acest set este necesar deoarece cautarea in graf furnizand acelasi nod ca sursa si destinatie ar returna un drum de lungime 1, chiar daca acel nod nu prezinta o bucla cu el insusi. Rotunjirea din linia 10 este necesara pentru a obtine cel putin un sufix pentru fiecare stare din S_{s_f}. Fiecare rulare adaugata in setul Γ (linia 13) include prefixul (ce conduce B intr-o stare finala) si o iteratie a sufixului (ce conduce B inapoi in acea stare finala). Astfel, fiecare element al Γ este finit, iar lungimea infinita necesara semanticii LTL va aparea prin repetarea infinita a sufixului si repetitiile corespunzatoare ale tranzitiilor in modelul $\mathscr{Q}$ - astfel, in linia 14 se stocheaza elementul fiecarei rulari de unde apar repetitiile.

Urmatorii pasi, (ii) si (iii), ai metodei propuse vor fi iterati pentru fiecare rulare diferita, r din setul Γ. Odata ce o rulare poate fi urmarita prin observatiile modelului $\mathscr{Q}$ (pasul (ii) se executa cu succes), iterarea acestor pasi poate fi terminata si se returneaza solutia obtinuta.

Se noteaza rularea curenta a lui B cu $r = s_0 s_1 \dots s_p \dots s_{L_r}$, unde L_r este numarul de stari din prefixul si sufixul lui r, s_p si s_{L_r} reprezinta aceeasi stare finala ce trebuie vizitata de o infinitate de ori, iar sufixul ce se repeta este $s_{p+1} \dots s_{L_r}$. Daca $L_r = p+1$, atunci sufixul are doar o stare, iar repetitiile acestuia se traduc prin oprirea robotilor in celulele atinse.

Pasul (ii): Pentru activarea tranzitiei $s_j \to s_{j+1}$ in rularea r a lui B, $j = 0, \dots, L_r - 1$, sunt necesare urmatoarele doua conditii:

(a) Sistemul $\mathscr{Q}$ trebuie sa atinga un marcaj final m care genereaza orice observatie din setul $\rho(s_j, s_{j+1}) \subseteq 2^{\Pi}$;

(b) Marcajele intermediare trebuie sa genereze observatii incluse in setul $\rho(s_j, s_j) \subseteq 2^{\Pi}$, a.i. sa nu fie cauzata tranzitia din starea s_j catre o alta stare.

Algorithm 2: Update B and construct set Γ of accepted runs

Input: RMPN $\mathscr{Q}$, Büchi B, R, κ

Output: Trimmed Büchi B, set of runs Γ

Compute $O = \bigcup_{p \in P} h(p)$;

$H = \underbrace{O \times O \times \ldots \times O}_{R\text{-times}}$;

for $s_i, \rho(s_i, s_j), s_j \in \rightarrow_B$ **do**

  $\rho(s_i, s_j) = \rho(s_i, s_j) \cap H$;

$\Gamma = \emptyset$;

for $s_0 \in S_0$ *and* $s_f \in F$ **do**

  Find at most κ paths in graph of B from s_0 to s_f (using k-shortest path algorithm);

  Denote the set of above paths by $Pref$;

  Let $S_{s_f} = \{s \in S | \exists (s, \tau, s_f) \in \rightarrow_B\}$ **for** $s \in S_{s_f}$ **do**

    Find at most $ceil\left(\frac{\kappa}{|S_{s_f}|}\right)$ paths from s_f to s;

    Move s_f from the beginning of each path to the end of path;

    Denote the set of above paths by $Suff$;

  Append to each path from $Pref$ each path from $Suff$ and add the resulted run to set Γ;

  For each run from Γ, store the index for beginning its suffix;

Pentru a impune/verifica o observatie intr-un marcaj fezabil dat m, definim pentru fiecare observatie $\pi_i \in \Pi$ o variabila binara x_i astfel incat:

$$x_i = \begin{cases} 1, & \text{daca } v_i \cdot m > 0 \\ 0, & \text{altfel.} \end{cases} \tag{2}$$

Urmatoarele doua constrangeri asigneaza valoarea corecta lui x_i:

$$\begin{cases} N \cdot x_i & \geq v_i \cdot m \\ x_i & \leq v_i \cdot m \end{cases}, \tag{3}$$

unde N este un numar mare. Se observa ca daca $v_i \cdot m > 0$, prima constrangere din (3) impunes $x_i = 1$, in timp ce daca $v_i \cdot m = 0$, a doua constrangere din (3) asigura $x_i = 0$.

In continuare derivam echivalentele formale in termeni de inegalitati liniare pentru pasul (ii-a). In acest sens, se considera un subset generic $\mathscr{S} \subseteq 2^{\Pi}$. Setul $\mathscr{S}$ (set de subseturi ale lui Π) poate fi vazut ca o disjunctie de conjunctii de propozitii din Π - astfel, ca o formula Booleana φ_S in FND (forma normala disjunctiva). Evident, $\varphi_{\mathscr{S}}$ este *True* daca $\neg(\neg\varphi_{\mathscr{S}})$ este *True*. Se observa ca $(\neg\varphi_{\mathscr{S}})$ este formula ce corespunde $2^{\Pi} \setminus \mathscr{S}$, adica, $\varphi_{2^{\Pi} \setminus \mathscr{S}}$, si dorim $\neg\varphi_{2^{\Pi} \setminus \mathscr{S}}$ sa fie *True*. Totusi, avand in vedere ca $\varphi_{2^{\Pi} \setminus \mathscr{S}}$ este o formula Booleana pentru setul $2^{\Pi} \setminus \mathscr{S}$ este si FND iar $\neg\varphi_{2^{\Pi} \setminus \mathscr{S}}$ va fi o CNF (forma normala conjunctiva). Inspirandu-ne din rezultatele din [21], $\neg\varphi_{2^{\Pi} \setminus \mathscr{S}}$ poate fi scrisa ca un set de constrangeri liniare folosind variabilele x_i. Pentru aceasta, este propus Alg. 3. Informal, odata ce este construit complementul lui $\mathscr{S}$ (line 1),

dorim ca orice element din acest complement sa fie fals (sa nu fie observat). Pentru fiecare element $\bar{s} \in 2^{\Pi} \setminus \mathscr{S}$, inegalitatea din linia 2 nu este afectata daca se observa o propozitie din $\bar{s}$ (termenul din partea dreapta este incrementat cu 1 datorita lui x_i, in timp ce termenul din partea stanga este incrementat datorita cardinalitatii lui $\bar{s}$). Totusi, pentru fiecare propozitie din $\bar{s}$ ce nu este observata, termenul drept nu este incrementat, ajutand astfel satisfacerea inegalitatii. In acelasi timp, termenul drept este decrementat (premiat) cu 1 daca este observata o propozitie din afara setului $\bar{s}$ (datorita x_j). Daca inegalitatea din linia 2 este adevarata, atunci conjunctia din $\varphi_{2^{\Pi} \setminus \mathscr{S}}$ corespunzatoare lui $\bar{s}$ este falsa, deci o disjunctie din FNC $\neg\varphi_{2^{\Pi} \setminus \mathscr{S}}$ devine adevarata. Daca $2^{\Pi} \setminus \mathscr{S}$ include elementul $\emptyset$, atunci trebuie sa fie observata cel putin o propozitie din Π pentru a incalca acest element al setului complement, adica inegalitatea din linia 3 ar trebui sa fie adevarata.

In concluzie, pentru a decide daca observatia curenta a sistemului RMPN $||V \cdot m||$ apartine unui set dat $\mathscr{S}$, trebuie ca mai intai sa se asocieze valorile variabilelor binare x_i cu observatiile sistemului RMPN, ca in inegalitatile (3). Apoi, aceste variabile binare trebuie sa satisfaca si inegalitatile returnate de Alg. 3.

Algorithm 3: Constraints for the set $\mathscr{S}$

Input: Set $\mathscr{S} \subseteq 2^{\Pi}$

Output: A set of linear constraints

1 Compute $2^{\Pi} \setminus \mathscr{S}$ (complement of S);

2 Add constraints

$$\sum_{\pi_i \in \bar{s}} x_i - \sum_{\pi_j \in (\Pi \setminus \bar{s})} x_j \leq |\bar{s}| - 1,\ \forall \bar{s} \in \left(2^{\Pi} \setminus \mathscr{S}\right), \bar{s} \neq \emptyset$$

3 If $\emptyset \in \left(2^{\Pi} \setminus \mathscr{S}\right)$, add constraint

$$\sum_{\pi_i \in \Pi} x_i \geq 1$$

Functia de cost: Pe baza constrangerilor liniare pentru RMPN, ce asigura o tranzitie in automatul Büchi, se dezvolta o formulare de programare intreaga mixta. Pentru aceasta, este necesara stabilirea unei functii de cost. Functia de cost propusa include costurile pentru miscarea robotilor intre celule si pentru efectuarea actiunilor corespunzatoare. Avand in vedere ca σ reprezinta vetorul de declansari, minimizarea costului total poate fi impusa prin minimizarea

$$Cost^T \cdot \sigma,$$

unde *Cost* reprezinta costul tranzitiilor definit in etapa 1 a metodei.

Aspectele de mai sus permit formularea conditiei (a) din pasul (ii) sub forma unei probleme de programare intreaga mixta (4).

$$
\begin{aligned}
& \min Cost^T \cdot \sigma \\
\text{s.t.} \quad & m = m_0 + C \cdot \sigma \\
& N \cdot x_i \geq v_i \cdot m, \forall \pi_i \in \Pi \\
& x_i \leq v_i \cdot m, \forall \pi_i \in \Pi \\
& \text{Lin. ineq. in } x_i \text{ given by Alg. 3 for set } \rho(s_j, s_{j+1}) \\
& m_0 - m \leq w \\
& -m_0 + m \leq w \\
& Pre \cdot \sigma + m \leq \gamma \cdot 1 \\
& m \in \mathbb{N}_{\geq 0}^{|P|}, \sigma \in \mathbb{N}_{\geq 0}^{|T|}, \quad x_i \in \{0,1\}, i = 1, \ldots, |\Pi| \\
& w \in \mathbb{R}^{|P|}, \quad \gamma \in \mathbb{R}
\end{aligned}
\tag{4}
$$

Se observa ca in loc sa se introduca constrangeri asupra tuturor variabilelor binare x_i, $i = 1, \ldots, |\Pi|$, se pot include doar constrangerile asupra propozitiilor ce apar in $\rho(s_j, s_{j+1})$ (acest lucru nu este cuprins in problema (4) pentru a mentine simplitatea notatiilor).

Problema (4) este rezolvata colosind rutine specifice de optimizare [22, 23]. Vectorul de declansari obtinut σ este apoi proiectat pe tranzitii individuale ale robotilor si secvente de pozitii vizitate/actiuni efectuate, prin declansarea fiecarei tranzitii validate in marcajul curent (vezi algoritmul din [24]). In cazuri rare, aceasta proiectie poate fi imposibila datorita functie de cost si constrangerilor asociate w si γ, cand se spune ca σ este *falsa*. Daca se obtine o solutie, inseamna ca prin declansarea tranzitiilor din σ, modelul RMPN atinge marcajul final m in care observatia activeaza tranzitia $s_j \rightarrow s_{j+1}$ in B.

Totusi, trebuie verificata daca observatiile intermediare ale modelului RMPN nu au declansat o tranzitie in B ce paraseste rularea curenta, adica intr-o alta stare decat s_j sau s_{j+1}. Pentru aceasta, se foloseste Alg. 4. Evident, daca nu se declanseaza nicio tranzitie in σ, nu este necesara nicio verificare suplimentara (linile 1-2 din Alg. 4) - robotii nu se deplaseaza si nu efectueaza nicio actiune, insemnand ca tranzitia $s_j \rightarrow s_{j+1}$ este deja activata cat timp in m_0. Linile 3-4 reprezinta procedura din [24] pentru a proiecta σ in tranzitii individuale si secvente de pozitii vizitate in RMPN. Apoi, in liniile 5-9 este gasit un set de iesiri intermediare generate de fiecare robot. In acest timp, presupunem ca robotii se deplaseaza individual (fara o sincronizare intermediara) si se sincronizeaza la ultima tranzitie de declansat, astfel incat vor patrunde sincron in pozitiile finale. Din acest motiv nu includem ultima iesire (liniile 8-9), din moment ce toate ultimile iesiri ale echipei vor valida cu siguranta tranzitia $s_j \rightarrow s_{j+1}$ dorita a lui B. Produsul Cartezian din linia 10 furnizeaza setul $Interm_{obs} \subseteq 2^{\Pi}$ ce include observatiile intermediare posibile ale sistemului RMPN ce pot fi generate in timp ce robotii evolueaza conform σ, sincronizarea avan loc doar la ultima tranzitie declansata. Testul din linia 11 este validat daca iesirie intermediare posibile valideaza o bucla $s_j \rightarrow s_j$ in B, caz in care spunem ca *solutia data de σ este aplicabila*. Altfel, (linia 13), starea s_j ar fi putut fi parasita catre o stare diferita de s_{j+1}, inainte ca RMPN sa atinga marcajul final m impus de MILP (4).

Daca σ data de problema (4) nu este aplicabila, se adauga mai multe constrangeri celor din problema (4), pentru conditia (ii-b). Concret, se considera vectorul cumulativ de marcaje m_t definit anterior pentru a impune restrictii asupra observatiilor traiectoriei, fara a include in

Algorithm 4: Check if σ returned by MILP (4) is applicable

Input: σ, RMPN model $\mathscr{Q}$, set $\rho(s_j, s_j)$
Output: Applicability of σ

if $\sigma = 0$ **then**
  σ is applicable;
Project σ to individual robot firing sequences;
Let $seq_i = seq_i[1], seq_i[2], \ldots, seq_i[|seq_i|]$ be the sequence of places visited by i^{th} robot, according to its firing sequence;
for each robot *i,* $i = 1, \ldots, R$ **do**
  if $|seq_i| = 1$ **then**
    $Obs_i = h(seq_i[1])$;
  else
    $Obs_i = \bigcup_{k=1}^{|seq_i|-1} h(seq_i[k])$;
$Interm_{obs} = Obs_1 \times Obs_2 \times \ldots \times Obs_R$;
if $Interm_{obs} \subseteq \rho(s_j, s_j)$ **then**
  σ is applicable;
else
  σ is not applicable;

m_t marcajul final m. Astfel, obtinem problema (5), ce contine de doua ori mai multe variabile binare decat (4), variabilele $x_{i(t)}$ corespunzand valorilor de adevar ale propozitiilor date de m_t.

$$
\begin{aligned}
& \min Cost^T \cdot \sigma \\
\text{s.t.} \quad & m = m_0 + C \cdot \sigma \\
& N \cdot x_i \geq v_i \cdot m, \forall \pi_i \in \Pi \\
& x_i \leq v_i \cdot m, \forall \pi_i \in \Pi \\
& \text{Lin. ineq. in } x_i \text{ given by Alg. 3 for set } \rho(s_j, s_{j+1}) \\
& N \cdot x_{i(t)} \geq v_i \cdot (Pre \cdot \sigma), \forall \pi_i \in \Pi \\
& x_{i(t)} \leq v_i \cdot (Pre \cdot \sigma), \forall \pi_i \in \Pi \\
& \text{Lin. ineq. in } x_{i(t)} \text{ given by Alg. 3 for set } \rho(s_j, s_j) \\
& m_0 - m \leq w \\
& -m_0 + m \leq w \\
& Pre \cdot \sigma + m \leq \gamma \cdot 1 \\
& m \in \mathbb{N}^{|P|}_{\geq 0}, \sigma \in \mathbb{N}^{|T|}_{\geq 0}, \quad x_i, x_{i(t)} \in \{0,1\}, i = 1, \ldots, |\Pi| \\
& w \in \mathbb{R}^{|P|}, \gamma \in \mathbb{R}
\end{aligned}
\tag{5}
$$

Informal, problema (5) include conditiile problemei (4) si conditia (b) din pasul (ii). Astfel, problema (4) poate fi computational mai rapid de rezolvat decat problema (5) (datorita numarului mai mic de variabile si constrangeri), in timp ce problema (5) nu poate returna un vector de declansari neaplicabil.

$$
\begin{aligned}
& & \min Cost^T \cdot \textstyle\sum_{i=1}^{k} \sigma_i & \\
\text{s.t.} & & m_i &= m_{i-1} + C \cdot \sigma_i, i = 1, \ldots, k \\
& & m_{i-1} - Pre \cdot \sigma_i &\geq 0, i = 1, \ldots, k \\
& & N \cdot x_i &\geq v_i \cdot m_k, \forall \pi_i \in \Pi \\
& & x_i &\leq v_i \cdot m_k, \forall \pi_i \in \Pi \\
& & \text{Lin. ineq. in } x_i \text{ given} & \quad \text{by Alg. 3 for set } \rho(s_j, s_{j+1}) \\
& & N \cdot x_{i(t)} &\geq v_i \cdot \left(\textstyle\sum_{i=0}^{k-1} m_i \right), \forall \pi_i \in \Pi \\
& & x_{i(t)} &\leq v_i \cdot \left(\textstyle\sum_{i=0}^{k-1} m_i \right), \forall \pi_i \in \Pi \\
& & \text{Lin. ineq. in } x_{i(t)} \text{ given} & \quad \text{by Alg. 3 for set } \rho(s_j, s_j) \\
& & m_0 - m_k &\leq w \\
& & -m_0 + m_k &\leq w \\
& & m_i &\leq \gamma_i \cdot 1, i = 1, \ldots, k \\
& m_i \in \mathbb{N}_{\geq 0}^{|P|}, \sigma_i \in \mathbb{N}_{\geq 0}^{|T|}, i = 1, \ldots, k & x_i, x_{i(t)} &\in \{0,1\}, i = 1, \ldots, |\Pi| \\
& w \in \mathbb{R}^{|P|}, & \gamma_i &\in \mathbb{R}, i = 1, \ldots, k
\end{aligned}
\tag{6}
$$

Pasul (iii) al solutiei propuse converteste vetorul σ in secvente de declansari pentru fiecare robot, folosind o metoda din [24]. Apoi, se adauga aceste secvente secventelor anterioare ale robotilor si se impune ca robotii ce efectueaza o deplasare/actiune (aceia care efectueaza cel putin o tranzitie) sa se sincronizeze la ultima tranzitie din σ. Fiecare declansare va corespunde parcurgerii drumului cel mai scurt intre cele doua celule corespunzatoare pozitiilor si efectuarea actiunii de iesire.

Activitatea 2.2. Integrarea algoritmilor dezvoltaţi într-un mediu software

Robot Motion Toolbox (RMTool) [11] oferă o colecţie de instrumente dedicate modelării, planificării căilor şi controlului mişcării roboţilor mobili. RMTool este încorporat în mediul MATLAB, care oferă avantajul considerabil de a crea instrumente algebrice, statistice şi grafice puternice care exploatează rutinele de înaltă calitate disponibile în MATLAB. Interacţiunea utilizator mediu de simulare se realizează printr-o interfaţă grafică (Fig. 2) care permite introducerea de valori pentru diferiţi parametri aferenţi algoritmilor de planificare pentru roboţi mobile cu roţi. Interfaţa poate fi împărţită în cinci zone principale:

1. Meniu
2. Zona de reprezentare (spaţiu de lucru, traiectorie, informaţii despre stări interne ale roboţilor mobili)
3. Panou planificare misiuni
4. Panou definire misiuni
5. Panou execuţie simulări

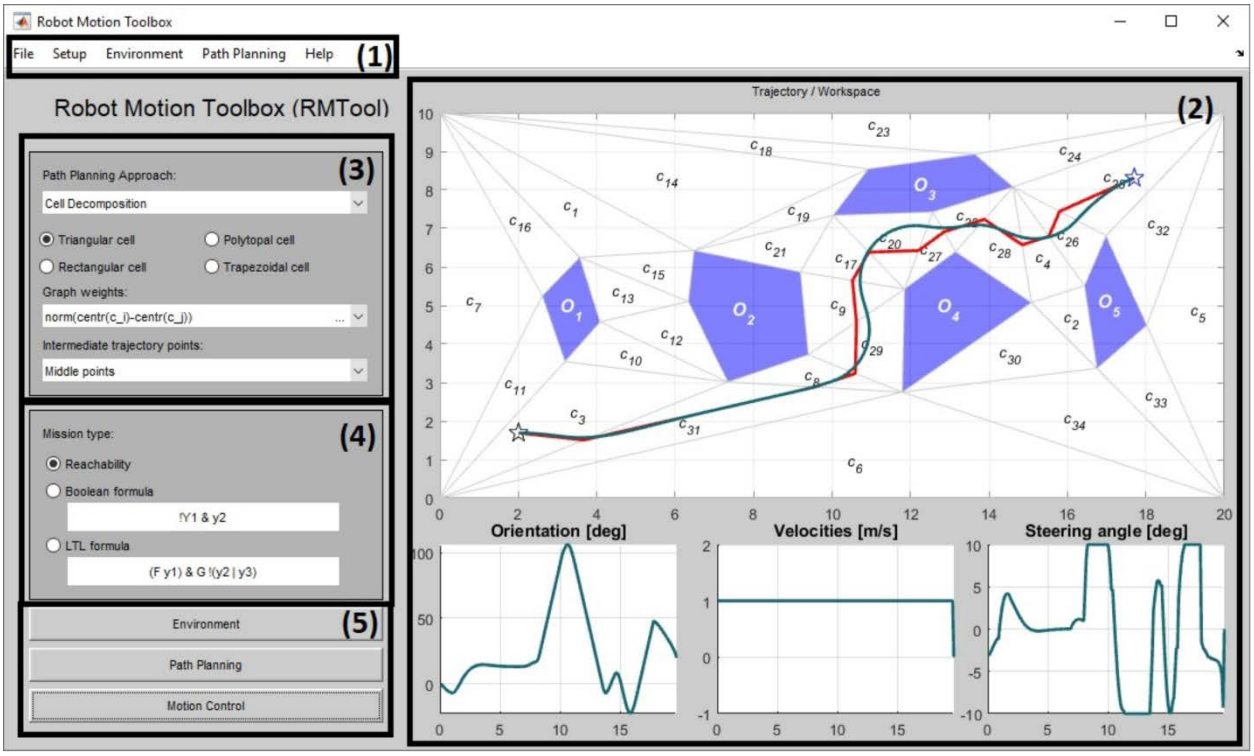


Fig. 2: Interfaţă grafică RMTool

Conform algoritmilor propuşi în această etapă a proiectului [5], RMTool a fost extins prin includerea în zona 4 a descrierii misiunii prin formule boolene şi LTL (Fig. 3). În noua versiune RMTool permite planificarea căilor de mişcare folosind specificaţii expresive sau de nivel înalt, în principal pentru un grup de roboţi mobili identici.

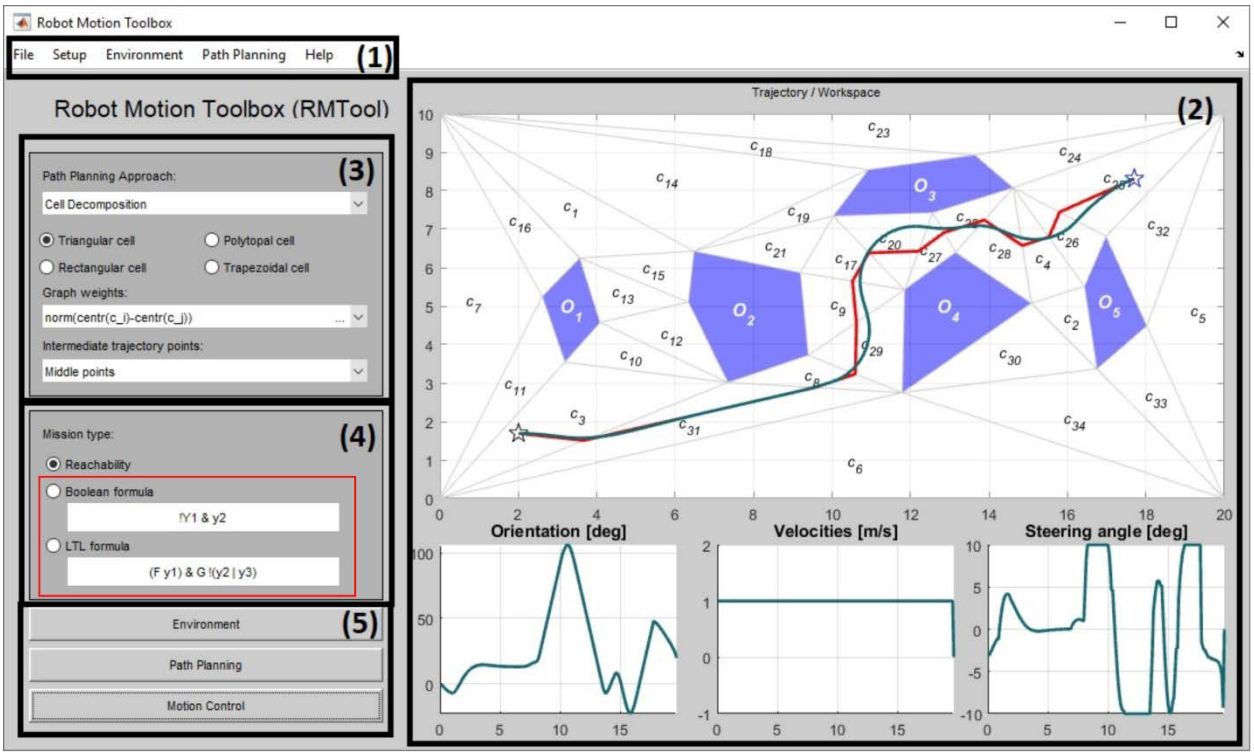


Fig. 3: Extindere RMTool prin includerea descrierii misiunii prin formule boolene şi LTL

Pentru exemplificare considerăm un spaţiu de lucru cu doi roboţi mobili şi şase regiuni de interes notate O1, O2, O3, O4, O5 şi O6. În Fig. 4 este ilustrat rezultatul planificării unei misiuni descrise prin formula LTL:

$$\phi = \mathrm{F}(\mathrm{y}_2)\,\&\,\mathrm{F}(\mathrm{y}_3)\,\&\,\mathrm{G}(!\mathrm{y}_4)\,\&\,(!\mathrm{y}_2 \cup \mathrm{y}_1)\,\&\,\mathrm{F}(\mathrm{y}_5\,\&\,\mathrm{y}_6)$$

Îndeplinirea formulei este echivalentă cu:

- regiunile O2 şi O3 trebuie vizitate eventual
- regiunea O4 trebuie evitată pe parcursul misiunii
- regiunea O2 poate fi vizitată după ce a fost vizitată O1
- regiunile O5 şi O6 trebuie ocupate eventual în acelaşi moment.

Fig. 5 ilustrează rezultatul planificării unei misiuni descrise prin formula booleană

$$\phi = \mathrm{y}_1\,\&\,!\mathrm{Y}_2\,\&\,\mathrm{y}_3\,\&\,\mathrm{Y}_4\,\&\,\mathrm{Y}_5\,\&\,\mathrm{Y}_6$$

Îndeplinirea formulei este echivalentă cu:

- regiunile O1 şi O3 trebuie ocupate în starea finală
- regiunile O4 şi O5 trebuie traversate pe parcursul mişcării
- regiunile O2 şi O6 nu trebuie traversate pe parcursul mişcării

Scopul acestui experiment este de a vedea cum algoritmul de planificare generează schimbări în misiunea roboților în funcție de sarcină și de distanța dintre roboți și regiunile respective de interes pentru a fi vizitate sau evitate. Asta înseamnă că distanța globală parcursă de roboți este cea mai scurtă posibil. Se observă că în acest caz robotul 2 ajunge în trei regiuni (O3, O4, O5) iar robotul 1 vizitează o singură regiune (O1).

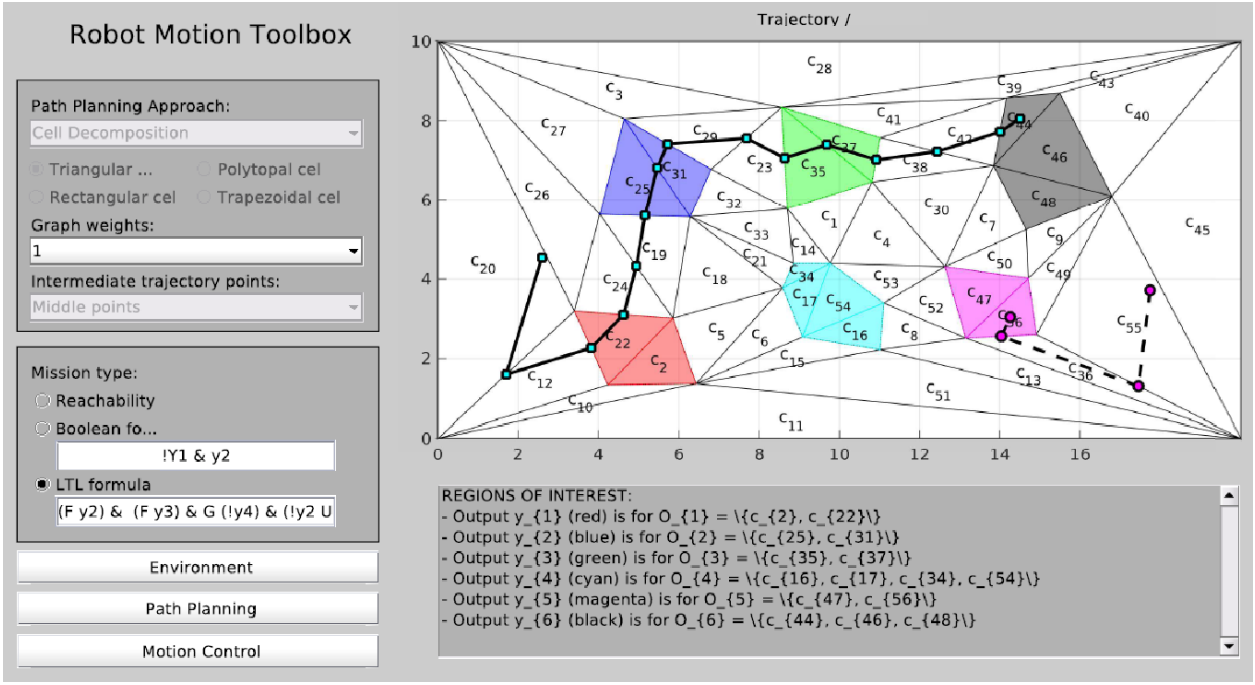


Fig. 4: Rezultat de planificare bazată pe formulă LTL

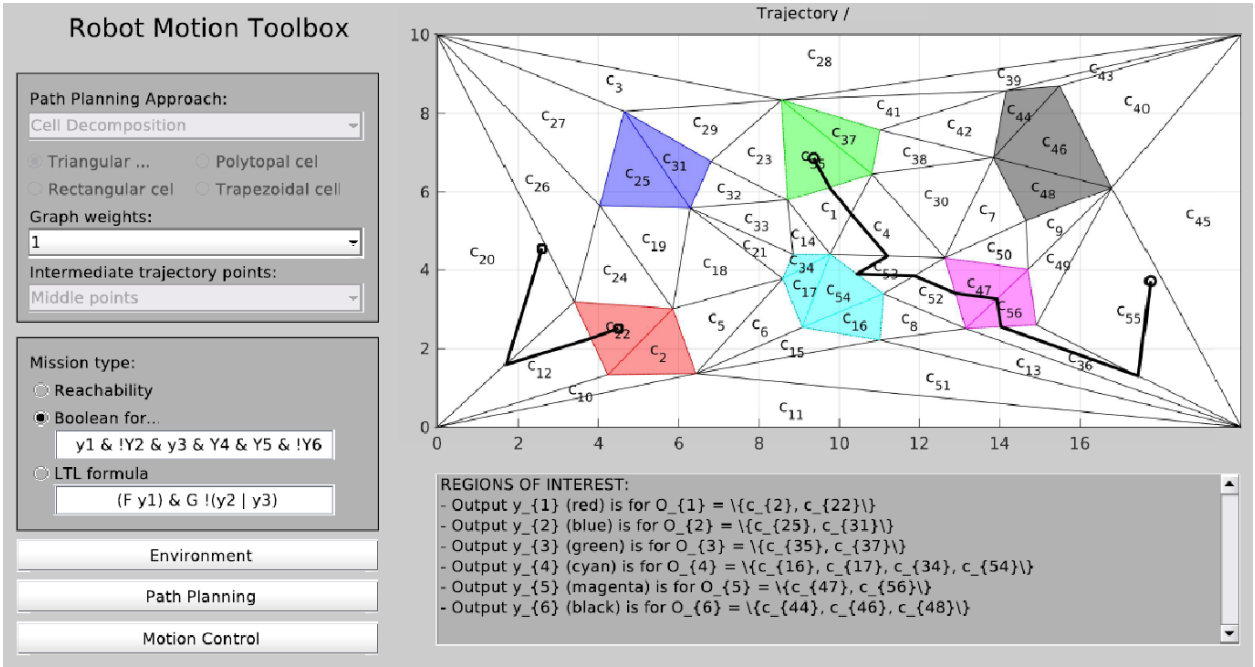


Fig. 5: Rezultat de planificare bazată pe formulă Booleană

Activitatea 2.3. Rezultate preliminare pentru experimente în timp real

Configurații de test

Platforma Crazyflie 2.0 implementează un mecanism configurabil de estimare a poziției. Astfel, agregarea și filtrarea datelor poate fi realizată pe baza unui filtru Kalman extins (EKF) sau pe baza unui filtru complementar [25]. De asemenea, plaja de intrări ale filtrelor este extensibilă - în varianta de bază, estimatorul folosește date de la unitatea inerțială integrată (MPU-9250), iar prin adăugarea unor module de extensie, acesta poate primi la intrare date de poziționare relativă sau absolută.

Platforma Crazyflie 2.0 este echipată cu doi conectori de extensie și poate găzdui simultan unul sau mai multe module suplimentare (eng. decks). Trei astfel de module, Z-Ranger v2 (I), Flow deck v2 (II) și Loco Positioning Deck (III) sunt utilizate pentru a îmbunătăți performanța de estimare a poziției și orientării dronei, deoarece în mod implicit estimatorul folosește doar date de la senzorii integrați [25, 26]. Toate cele trei module testate în acest proiect oferă informații despre poziția (și orientarea dronei), mărimile citite de la senzori fiind utilizate în estimatorul implementat pe dronă [3].

I. Z-Ranger v2 - este echipat cu un senzor laser VL53L1X ce este utilizat pentru a determina distanța până la sol. Distanța maxim de măsurare este de 400 cm, însă performanțele scad în condițiile puternice de iluminare.

II. Flow Deck v2 – este echipat cu un senzor VL53L1X (similar cu Z-Ranger) și un senzor optic de urmărire a mișcării, PMW3901MB. Acest senzor optic este similar în construcție cu un senzor de mouse optic și poate urmări mișcarea începând de la o înălțime de 8 cm. De menționat este faptul că în foaia de catalog a senzorului nu sunt disponibile informații legate de modul în care este procesată și interpretată ieșirea. De asemenea, producătorul dronei CF 2.0 afirmă din cauza lipsei de informații, integrarea senzorului s-a realizat experimental, prin încercări repetate.

III. Loco Positioning Deck - este echipat cu un modul de comunicație DW1000. Prin intermediul unor secvențe de comunicație specifice, drona determină poziția absolută în domeniul de evoluție 3D. Acest modul este asemănător cu un sistem de tip GPS în miniatură [28].

Instalarea și configurarea elementelor hardware

Spațiul de lucru în care evoluează drona este reprezentat de o platformă de 4 x 2,5 metri, ce poate fi înconjurată cu o plasă de siguranță. Aceasta dispune și de un senzor Kinect montat pe plafon, în centrul platformei, senzor ce a fost utilizat în procesul de evaluare. Senzorul este montat la o înălțime de 2,8 m și poate acoperi o suprafață de aproximativ 2,1 x 1,3 m la o înălțime de 1 m [3], [27].

Configurațiile de test care au presupus folosirea modulelor Z-Ranger (I) și Flow Deck (II) nu au ridicat probleme deosebite din punct de vedere hardware, deoarece orientarea sistemului de coordonate în care evoluează drona este relativă la poziția de start din care decolează [3]. Folosirea unor astfel de configurații ridică două probleme când există mai multe drone care evoluează în paralel și care necesită coordonare în acțiuni:

- (1) diferențele de poziție și orientare trebuie cunoscute de toate dronele (i.e. configurare manuală pentru fiecare experiment) și
- (2) erorile de estimare sunt acumulate individual de fiecare dronă (i.e. de la IMU și PMW3901MB).

Astfel, s-a decis folosirea unui sistem de poziționare absolută (Loco Positioning System - LPS (III)), ce presupune instalarea unui set de module radio (ancore) în exteriorul domeniului de evoluție, acestea fiind folosite de drone pentru determinarea poziției. Funcțional, sistemul se bazează pe măsurarea timpului de propagare a unor pachete radio (eng. *Time-of-Flight*) provenite de la ancore aflate la poziții fixe, principiu folosit și cazul altor tehnologii de poziționare trecute și actuale (Loran-C, Omega, RSDN-20, GPS, GLONASS, Beidou etc.).

Modulele LPS implementează standardul IEEE 802.15.4a-2011 (i.e. comunicații radio cu consum redus și rate de transfer mici), ce definește 16 canale UWB (Ultra-Wide Band), cu următoarele proprietăți [28]:

- canalele au o lățime de bandă de cel puțin 500 MHz și prin distribuirea uniformă a energiei în banda de emisie nu ridică probleme de coexistență cu alte tehnologii radio;
- codificările la nivel fizic presupun folosirea unor impulsuri scurte pentru transmisia datelor astfel reducându-se influența interferențelor datorate căilor de propagare multiple (eng. multipath fading);
- timpul de propagare (eng. time-of-flight) al semnalelor poate fi determinat cu o acuratețe mai mare decât în cazul tehnologiilor ce folosesc lățimi de bandă mai mici.

Tehnica de bază folosită pentru determinarea timpului de propagare nu necesită o referință de timp comună pentru sistemele care comunică. TWR (Two-way ranging) presupune generarea unei secvențe de 4 mesaje și atașarea unor mărci de timp la transmisia, respectiv recepția mesajelor așa cum este ilustrat în Fig. 6). Prin măsurarea timpilor de transmisie dus-întors (Tround2, Tround2) și a timpilor de procesare dintre o recepție și o transmisie (Treply1, Treply2), stația Tag care a inițiat secvența de comunicație (i.e. drona) poate extrage timpul de propagare al mesajelor, valoarea lui fiind direct proporțională cu distanța parcursă. Prin comunicarea succesivă cu mai multe ancore și prin cunoașterea pozițiilor absolute ale ancorelor, stația Tag poate să-și estimeze poziția prin triangularizare.

Existența mai multor stații Tag implică o accesare concurentă a mediului de comunicație (fiecare va iniția secvențe de comunicație cu ancorele), astfel apărând necesitatea folosirii unui mecanism de control al accesului la mediu, fie de tip CSMA/CA, fie de tip TDMA. Ambele prezintă dezavantaje, CSMA/CA nu se pretează aplicațiilor de localizare deoarece implică retransmisii, respectiv amânarea unor transmisii cu perioade aleatoare de timp, iar TDMA nu se pretează aplicațiilor ce includ un număr variabil de ancore, respectiv stații Tag (i.e. drone) deoarece prezintă o scalabilitate limitată. Astfel, producătorii LPS pun la dispoziție modul de

funcționare TDoA2 (Time-difference of arrival), descris de Ledergerber [28], mod ce presupune implementarea comunicației TWR între maxim 8 ancore printr-un mecanism TDMA și recepția pachetelor TWR de un număr nelimitat de stații Tag. Pe baza marcării timpilor de recepție a mesajelor provenite de la ancore diferite, o stație Tag poate să determine indirect timpii de propagare, respectiv distanțele față de ancore.

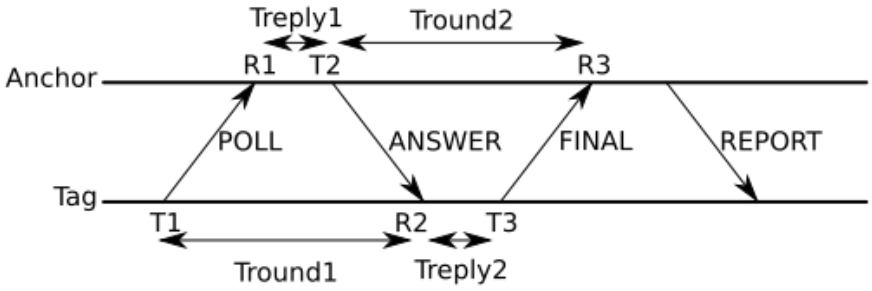


Fig. 6 Secvența de comunicație TWR [28]

Configurația de referință descrisă de producător [29], presupune folosirea a cel puțin șase ancore poziționate la înălțimi diferite (în cel puțin două planuri) și la cel puțin doi metri distanță pe oricare două axe, ca în Fig. 7 (a) și (b). După instalarea inițială a ancorelor, au fost derulate experimente pentru determinarea acurateții de poziționare și s-a constatat existența unei variații în poziționare (referit mai departe ca *jitter*), de 30-50 cm pentru axele X, Y, respectiv 15-20 cm pentru axa Z.

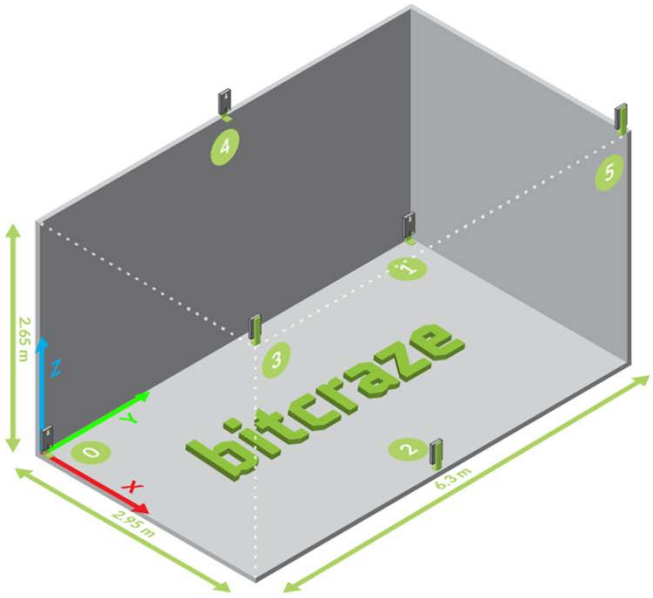


Fig. 7. (a) Configurația de referință

(b) Poziționarea ancorelor la nivelul spațiului de lucru

Pentru a facilita achiziția semnalelor de la ancore, a fost dezvoltată o aplicație de monitorizare a mesajelor vehiculate de ancore, interfață ilustrată în Fig. 8. Această aplicație oferă informații despre media și dispersia distanțelor raportate (A), numărul și secvențierea mesajelor generate (B), respectiv pozițiile raportate de ancore (C). Astfel, au fost observate discrepanțe mari între distanțele determinate în cele două direcții de comunicație pentru anumite perechi de ancore. În Fig. 8, se pot observa diferențe (marcate cu roșu) pentru ancorele A3, A4, A6, respectiv A7, ceea ce sugerează o poziționare defectuoasă a lor deoarece în benzile de frecvență folosite (3-6 GHz) nu poate fi eliminată complet influența obiectelor de dimensiuni mari (pereți, tavan, dulapuri, etc.) nici măcar prin intermediul tehnologie UWB.

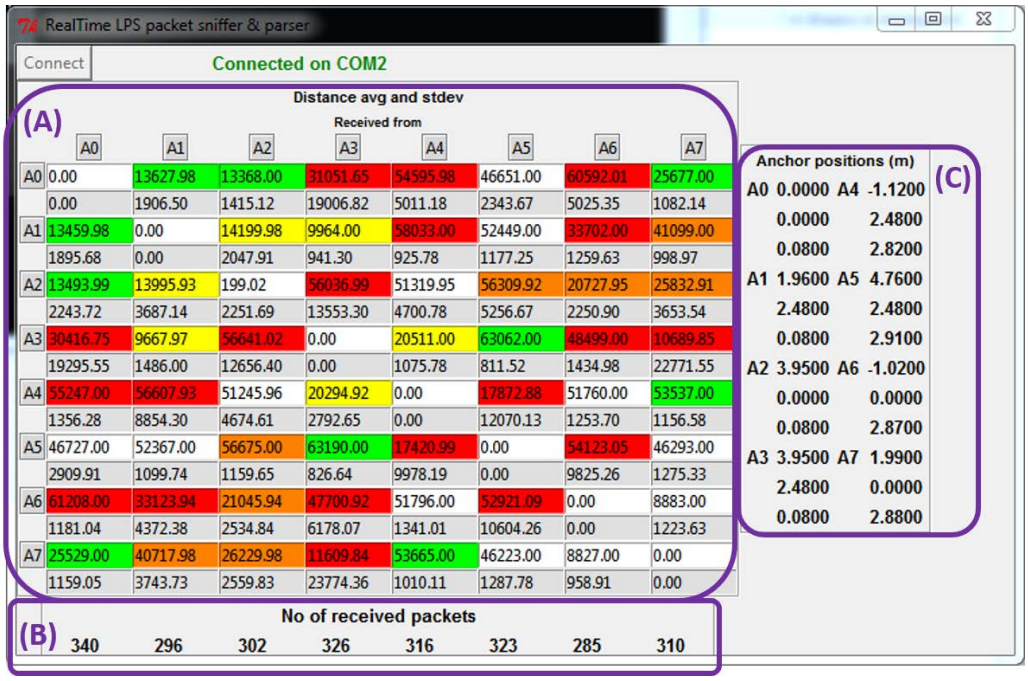
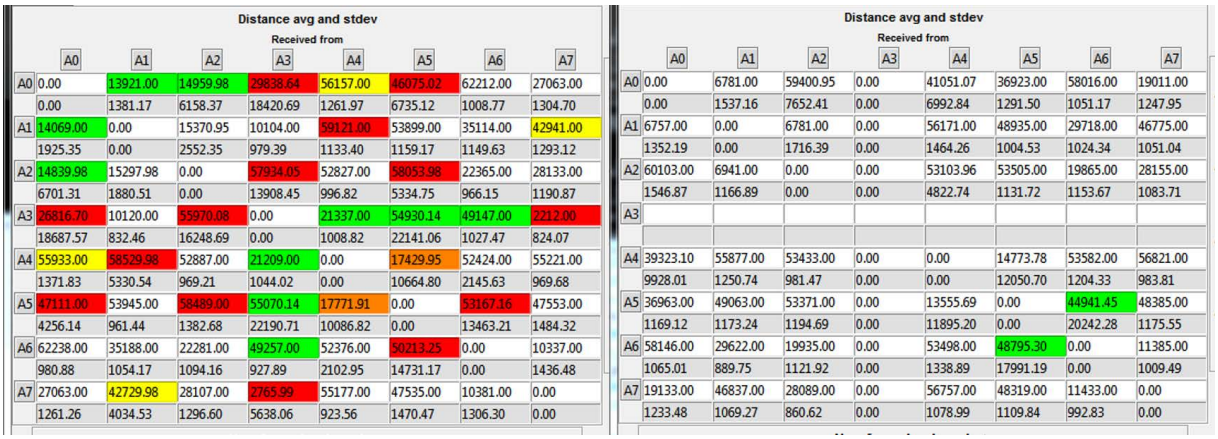


Fig. 8. Aplicația dezvoltată pentru monitorizarea comunicației dintre ancore

Pentru îmbunătățirea acurateții, ancorele au fost montate pe suporturi de plastic cu o înălțime de 20 cm, printate 3D, ce permit fixarea atât în plan orizontal, cât și vertical. Prin monitorizare continuă a mesajelor, conform cu Fig. 9 (a), au fost determinate noi poziții pentru ancore, valorile distanțelor după repoziționare fiind ilustrate în Fig. 9 (b), pentru care *jitter*-ul de poziționare a fost redus la 5 cm, precizia de poziționare declarată de producător fiind de 5 cm [29].

Aplicațiile UWB au fost studiate și de alți autori în diverse scenarii pentru monitorizarea de obiecte, persoane, echipamente, etc. În [30], autorii analizează efectul de ecranare reciprocă și impactul acestuia, precum și de metodele de îmbunătățire a urmăririi multiple a persoanelor în mișcare prin radarele UWB, propunând 3 abordări complementare. Autorii din [31] propun un algoritm de urmărire a pietonilor bazat pe ancore a căror poziție nu se cunoaște, precizia obținută fiind de 0,77 metri în 90% din cazuri.



Distance avg and stdev

Received from

	A0	A1	A2	A3	A4	A5	A6	A7
A0	0.00	13921.00	14959.98	29838.64	56157.00	46075.02	62212.00	27063.00
	0.00	1381.17	6158.37	18420.69	1261.97	6735.12	1008.77	1304.70
A1	14069.00	0.00	15370.95	10104.00	59121.00	53899.00	35114.00	42941.00
	1925.35	0.00	2552.35	979.39	1133.40	1159.17	1149.63	1293.12
A2	14839.98	15297.98	0.00	57934.05	52827.00	58053.98	22365.00	28133.00
	6701.31	1880.51	0.00	13908.45	996.82	5334.75	966.15	1190.87
A3	26816.70	10120.00	55970.08	0.00	21337.00	54930.14	49147.00	2212.00
	18687.57	832.46	16248.69	0.00	1008.82	22141.06	1027.47	824.07
A4	55933.00	58529.98	52887.00	21209.00	0.00	17429.95	52424.00	55221.00
	1371.83	5330.54	969.21	1044.02	0.00	10664.80	2145.63	969.68
A5	47111.00	53945.00	58489.00	55070.14	17771.91	0.00	53167.16	47553.00
	4256.14	961.44	1382.68	22190.71	10086.82	0.00	13463.21	1484.32
A6	62238.00	35188.00	22281.00	49257.00	52376.00	50213.25	0.00	10337.00
	980.88	1054.17	1094.16	927.89	2102.95	14731.17	0.00	1436.48
A7	27063.00	42729.98	28107.00	2765.99	55177.00	47535.00	10381.00	0.00
	1261.26	4034.53	1296.60	5638.06	923.56	1470.47	1306.30	0.00

Distance avg and stdev

Received from

	A0	A1	A2	A3	A4	A5	A6	A7
A0	0.00	6781.00	59400.95	0.00	41051.07	36923.00	58016.00	19011.00
	0.00	1537.16	7652.41	0.00	6992.84	1291.50	1051.17	1247.95
A1	6757.00	0.00	6781.00	0.00	56171.00	48935.00	29718.00	46775.00
	1352.19	0.00	1716.39	0.00	1464.26	1004.53	1024.34	1051.04
A2	60103.00	6941.00	0.00	0.00	53103.96	53505.00	19865.00	28155.00
	1546.87	1166.89	0.00	0.00	4822.74	1131.72	1153.67	1083.71
A3								
A4	39323.10	55877.00	53433.00	0.00	0.00	14773.78	53582.00	56821.00
	9928.01	1250.74	981.47	0.00	0.00	12050.70	1204.33	983.81
A5	36963.00	49063.00	53371.00	0.00	13555.69	0.00	44941.45	48385.00
	1169.12	1173.24	1194.69	0.00	11895.20	0.00	20242.28	1175.55
A6	58146.00	29622.00	19935.00	0.00	53498.00	48795.30	0.00	11385.00
	1065.01	889.75	1121.92	0.00	1338.89	17991.19	0.00	1009.49
A7	19133.00	46837.00	28089.00	0.00	56757.00	48319.00	11433.00	0.00
	1233.48	1069.27	860.62	0.00	1078.99	1109.84	992.83	0.00

Fig. 9. (a) Valorile distanțelor după repoziționarea parțială a ancorelor;

(b) Valorile distanțelor după repoziționarea ancorelor

Descrierea experimentelor și analiza performanțelor

Cercetarea noastră vizează evaluarea performanței zborului dronei în diferite configurații hardware, deoarece acestea sunt direct legate de precizia estimatorului de poziție [3], [32]. În acest studiu sunt efectuate câteva experimente, considerând 3 configurații ale dronei CF 2.0 cu modulele Z-Ranger și Flow Deck și Loco Positioning Deck și analiza performanțelor pentru fiecare configurație. Această necesitate decurge din faptul că elementele hardware nu sunt bine documentate nici dintr-o perspectivă de implementare, nici din una funcțională. Pentru fiecare dintre configurațiile HW și scenariile de test, experimentele conțin două faze principale: (i) rularea testului și achiziția de date și (ii) prelucrarea datelor offline și analiza rezultatelor, așa cum este ilustrat în Fig. 10.

Prima fază include zborul CF2 în diferite scenarii și monitorizarea dinamicii zborului. Configurația experimentală include atât monitorizarea în timp real a datelor de la senzori și procesarea acestora, cât și date obținute de la senzorul Kinect montat la nivelul platformei de lucru. Pentru prima fază, CF2 a fost marcată cu un marker de culoare roșie pentru a-i îmbunătăți vizibilitatea pentru senzorul Kinect. Apoi, comenzile de poziționare (decolare, deplasare și aterizare) sunt trimise secvențial către CF2. În a doua fază, CF2 este detectată de senzorul Kinect, obținându-se astfel poziția dronei pe cele 3 coordonate.

Parametrii de interes monitorizați de pe drona CF2 sunt unghiurile roll (ϕ), pitch (θ) și yaw (ψ), vitezele unghiulare ale dronei *p, q, r*, vitezele liniare ale dronei *u, v, w* corespunzătoare celor 3 axe, valorile PWM ale motoarelor, date preluate de la senzori (accelerație, viteză unghiulară, mișcare XY) [3].

Fig. 10. Rularea testelor, achiziția de date și prelucrarea datelor offline

Au fost utilizate drone CF2, considerând cele 3 module de extensie, iar experimentele au fost rulate de cel puțin 4 ori, fiind raportate cele mai bune rezultate. Pentru fiecare configurație HW, am considerat 4 scenarii de testare. Experimentele s-au desfășurat impunând o înălțime minimă constantă de 1 m față de sol, restricție motivată de precizia senzorului Kinect. În Tabel 1 sunt descrise configurațiile HW utilizate în experimente [3].

Tabel 1: Configurații hardware pentru experimente cu drona CF2

Configurație HW	**Descriere**	**Intrările estimatorului**
Drona CF2	Kit-ul standard, fără module suplimentare. Conține senzor IMU	Accelerație și viteză unghiulară
Drona CF2 și Z-Ranger v2 deck	Z - Ranger utilizează senzorul laser VL53L1X ToF pentru măsurarea distanței față de sol. Interval maxim pentru distanța până la sol: 400cm	Accelerație, viteză unghiulară, distanță pe verticală (axa Z).
Drona CF2 și Flow Deck v2	Flow Deck încorporează senzorul laser VL53L1X și un senzor optic PMW3901MB pentru detecția mișcării pe axele XY. Senzorul optic detectează mișcarea de la cel puțin 8 cm înălțime.	Accelerație, viteză unghiulară, distanță pe verticală (axa Z), deplasare pe axele X,Y.
Drona CF2 și Loco Positioning Deck	Sistemul de ancore este configurat în modul TDoA2, deck-ul determină poziția absolută pe baza pachetelor recepționate.	Accelerație, viteză unghiulară, poziția absolută în sistemul de coordonate LPS.

Pentru fiecare configurație hardware au fost considerate patru scenarii de testare, descrise în Tabelul 2.

Tabel 2 Scenarii de testare

ID	Descriere test
1.	Zbor la punct fix la înălțimea de 1 metru timp de 5 secunde
2.	Deplasare liniară, 2m, pe axa X la înălțimea de 1 metru
3.	Deplasare de-a lungul unui pătrat de 1x1 metru, la înălțimea de 1m
4.	Deplasare în forma unui cerc cu raza de 1 metru (20 de segmente liniare) la înălțimea de 1m

Analiza performanțelor obținute pentru controlul poziției dronei în spațiul 3D

Structura de reglare implicită livrată împreună cu drona CF 2.0 implică existența a 9 regulatoare discrete PID conectate în cascadă, implementarea acestora realizându-se în limbajul de programare C. Bucla exterioară dedicată pentru controlul de atitudine, iar bucla internă cu regulatoare PID pentru controlul vitezelor unghiulare ale dronei.

Experimentele în timp real care implică configurațiile hardware CF2 și CF2 cu Z - Ranger au subliniat faptul că filtrul Kalman extins, implementat pe dronă nu este capabil să estimeze poziția cu o eroare rezonabilă. Încercările de testare din prima configurație, Kit-ul standard, fără module suplimentare, au eșuat din faza de decolare. Pentru configurația care a inclus modulul Z - Ranger, a existat o derivă semnificativă pe axele X-Y. În testele noastre în timp real, folosind CF2 și Flow Deck, am folosit structura de reglare automată în cascadă cu 9 regulatoare PID pornind de la parametrii impliciți ai regulatoarelor. În Tabel 3 sunt ilustrate valorile parametrilor regulatoarelor PID utilizate în experimentele în timp real [3].

Tabel 3: Parametrii PID dronă

Mărimi controlate	Kr	Ti	Td
x	2	0	0
y	2	0	0
z	2	0.5	0
roll	3.5	3	0
pitch	6	3	0
yaw	6	1	0.35
p	25	1	0
q	25	1	0
r	25	15	0

Un comportament stabil al dronei se poate obține dacă, pentru regulatorul PID pentru controlul poziției pe axa Z, factorul proporțional (Kr) ia valori în intervalul [1.8 - 2.3], iar componenta integratoare (Ti) în intervalul [0.1 - 0.5]. Testele au fost efectuate, luând în considerare semnale de referință pentru X (deplasarea înainte) și deplasarea pe axa Z (deplasarea pe înălțime) conform cu scenariile de test din Tabel 2, [3].

În experimentele în timp real, senzorul Kinect nu a putut detecta CF2 atunci când zbura la înălțimi sub 90 de centimetri, drona nefiind detectată din cauza rezolutiei senzorului (512x424 pixeli). Fig. 10 ilustrează traiectoria dronei, mișcarea de-a lungul axei X („forward movement") pentru o distanță de 2m, axei Y („lateral movement") la o înălțime de 1 metru („height"). În Fig. 11 este ilustrată comparativ traiectoria dronei, reprezentarea cu roșu fiind dată de estimatorul Kalman, ce preia date de la senzori, iar cu albastru datele preluate din procesarea imaginilor de la senzorul Kinect. Perioada de eșantionare utilizată în experimentele în timp real este de 50 ms. Din figură, se pot observa oscilații ale dronei cu amplitudinea de 0,1 m în jurul axei Y față de valoarea de referință. Oscilații cu amplitudinea de 0.1 m apar în jurul punctului de referință, de asemenea, în modul „hovering".

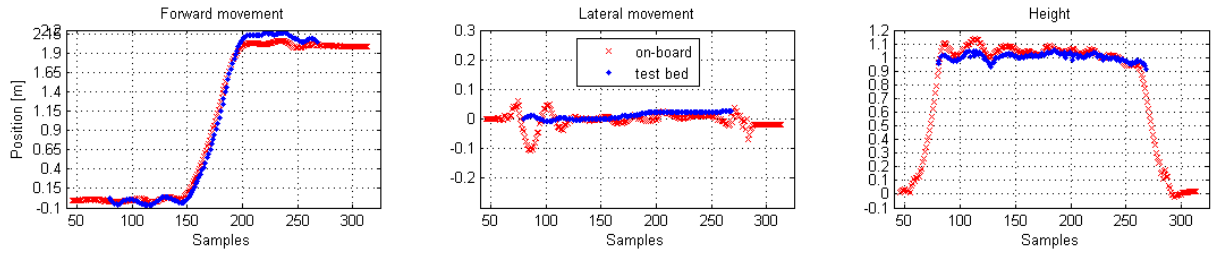


Fig. 11. Deplasarea dronei pe cele 3 axe, considerând scenariul 2

Fig. 12 ilustrează traiectoria parcursă de dronă pentru descrierea unui pătrat. De asemenea, se observă că CF2 urmărește referința impusă, cu mici oscilații pe axa Z [Budaciu et. al, 2019].

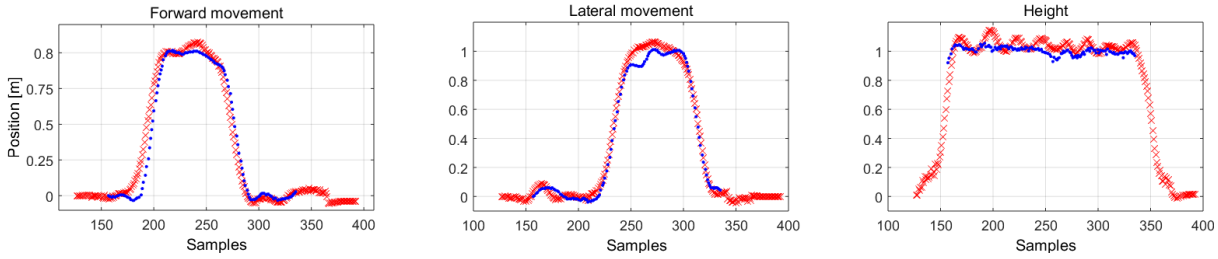


Fig. 12 Deplasarea dronei pe cele 3 axe, considerând scenariul 3

Fig. 13 ilustrează traiectoria circulară parcursă de dronă. De asemenea, se observă că CF2 urmărește referința impusă, un cerc cu rază de 1 metru, cu oscilații mici pe axa Z, timpul de răspuns fiind de aproximativ 2.5 secunde.

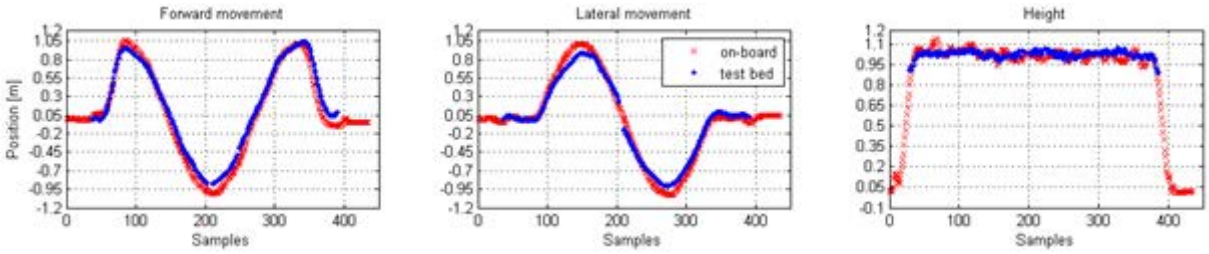
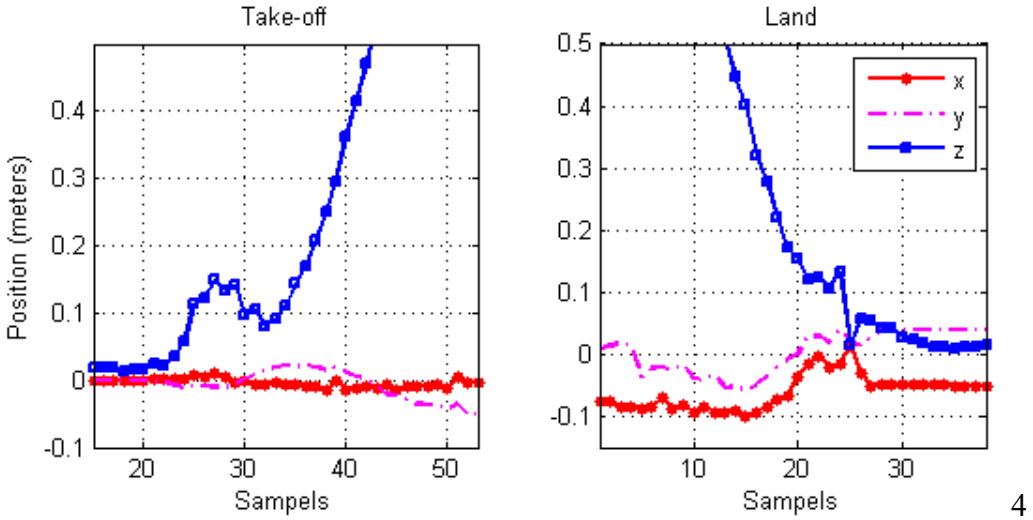


Fig. 13. Deplasarea dronei pe cele 3 axe, considerând scenariul 4



4

Fig. 14 Teste efectuate la decolare (Take off) și aterizare (Land)

De asemenea, s-a urmărit efectuarea unor teste de decolare, ilustrate în Fig. 14 (“Take off”) și aterizare (“Land”) în timpul cărora fiind monitorizate variabilele de ieșire și date de la senzorului optic. Datele înregistrate sugerează faptul că există o legătură între distanța minimă impusă de senzorul PWM3901MB (8 cm) și o ușoară derivă care a fost observată și ilustrată în Fig. 14 [3]. La aterizare, deriva s-a menținut sub pragul menționat. Deși nu există date de la producător cu privire la comportamentul senzorului în afara limitelor sale de lucru sau în ceea ce privește interpretarea datelor de ieșire, s-a observat că rezultate similare au fost obținute indiferent de suprafața solului folosită (mată, lucioasă, culoare uniformă, etc).

În cadrul acestui studiu s-au efectuat și experimente în timp real cu Drona CF2 și Loco Positioning Deck, cu poziționarea ancorelor la nivelul spațiului de lucru conform cu Fig. 7 (b). Sistemul de ancore este configurat în modul TDoA2, deck-ul determinând poziția absolută a dronei pe baza pachetelor recepționate. Rezultatele preliminare obținute cu această configurație HW demonstrează o poziționare a dronei cu o acuratețe comparabilă cu cea obținută cu Flow Deck. Avantajele oferite de configurația cu Loco Positioning Deck pot fi evidențiate în problemele de planificare în care sunt implicate mai multe drone cu sarcini diferite. Pe baza acestor rezultate, etapa următoare va valida algoritmii de planificare descriși anterior, considerând configurația HW cu Loco Positioning Deck. În acest context, algoritmii de reglare implementați implicit pe drona CF 2.0 vor fi reevaluați în vederea obținerii unui zbor cât mai stabil conform cu o referință impusă.

Bibliografie

[1] A. Burlacu, M. Kloetzer, C. Mahulea: Numerical Evaluation of Sample Gathering Solutions for Mobile Robots, *Applied Sciences*, vol.9, pp.791-809, 2019.

[2] M. Lupascu, S. Hustiu, A. Burlacu, M. Kloetzer: Path Planning for Autonomous Drones using 3D Rectangular Cuboid Decomposition, *23rd International Conference on System Theory, Control and Computing (ICSTCC)*, Sinaia, Romania, 2019.

[3] C. Budaciu, N. Botezatu, M. Kloetzer, A. Burlacu: On the Evaluation of the Crazyflie Modular Quadcopter System, *IEEE 24th International Conference on Emerging Technologies and Factory Automation (ETFA)*, Zaragoza, Spain, 2019.

[4] M. Kloetzer, A. Burlacu, G. Enescu, S. Caraiman, C. Mahulea: Optimal Indoor Goods Delivery Using Drones, *IEEE 24th International Conference on Emerging Technologies and Factory Automation (ETFA)*, Zaragoza, Spain, 2019.

[5] M. Kloetzer, C. Mahulea: Path Planning for Robotic Teams based on LTL Specifications and Petri Net Models, *Discrete Event Dynamic Systems - Theory and Applications*, accepted.

[6] Y.V. Pant, H. Abbs, R. Quaye, R. Mangharam: Fly-by-Logic: Control of Multi-Drone Fleets with Temporal Logic Objectives, *9th ACM/IEEE International Conference on Cyber-Physical Systems*, pp.186-197, 2018.

[7] P. Schillinger: Specification Decomposition and Formal Behavior Generation in Multi-Robot Systems, *PhD Thesis,* KTH, Stockholm, Sweden, 2017.

[8] P. Schillinger, M. Burger, D. Dimarogonas: Simultaneous task allocation and planning for temporal logic goals in heterogeneous multi-robot systems, *The International Journal of Robotics Research*, 2018.

[9] E. Q. V. Martins: On a multicriteria shortest path problem. *European Journal of Operational Research*, vol. 16, pp. 236-245, 1984.

[10] O. Kupferman, M. Y. Vardi: Model checking of safety properties, *Formal Methods in System Design*, Vol 19(3), pp. 291-314, 2001.

[11] R. Gonzalez, M. Kloetzer, C. Mahulea: Robot Motion Toolbox - RMTool, available at http://webdiis.unizar.es/RMTool/

[12] S.M. LaValle: *Planning algorithms*, Cambridge, 2006.

[13] T. Murata: Petri nets: Properties, analysis and applications, *Proceedings of the IEEE*, vol 77(4), pp. 541-580, 1989.

[14] M. Silva, E. Teruel, J.M. Colom: Linear algebraic and linear programming techniques for the analysis of P/T net systems, *Lecture on Petri Nets I: Basic Models*, vol. 1491, pp. 309-373, 1998.

[15] P. Wolper, M. Vardi, A. Sistla: Reasoning about infinite computation paths, *24th IEEE Symposium on Foundations of Computer Science*, pp. 185-194. Tucson, AZ, 1983.

[16] G. Holzmann: The Spin model checker, Primer and reference manual, *Addison-Wesley*, Reading, MA, 2004.

[17] P. Gastin, D. Oddoux: Fast LTL to Büchi automata translation, *13th Conference on Computer Aided Verification (CAV)*, LNCS 2102, pp. 53-65, 2001.

[18] A. Duret-Lutz, A. Lewkowicz, A. Fauchille, T. Michaud, E. Renault, L. Xu: Spot 2.0 - A framework for LTL and ω-automata manipulation, *Proc. of ATVA'16*, LNCS 9938, pp. 122-129, 2016.

[19] M.D. Berg, O. Cheong, M. van Kreveld: *Computational Geometry: Algorithms and Applications*, 3rd ed., Springer, 2008.

[20] J.Y. Yen: Finding the k shortest loopless paths in a network, *Management Science*, vol. 17(11), pp. 712-716, 1971.

[21] C. Mahulea, M. Kloetzer: Planning mobile robots with Boolean-based specifications, *53rd IEEE Conference on Decision and Control (CDC)*, Los Angeles, USA, 2014.

[22] A. Makhorin: GNU linear programming kit (2012), http://www.gnu.org/software/glpk/

[23] IBM: Ibm Ilog Cplex optimization studio software (2016), available at http://www.ibm.com/products/ilog-cplex-optimization-studio/.

[24] C. Mahulea, M. Kloetzer: Robot planning based on Boolean specifications using Petri net models, *IEEE Transactions on Automatic Control*, vol. 63(7), pp. 2218-2225, 2018.

[25] Crazyflie 2.0 documentation page, available at https://bit.ly/2vhTDy0

[26] M. Greiff: Modelling and control of the Crazyflie quadrotor for aggressive and autonomous flight by optical flow driven state estimation, *MSc. Thesis*, Lund University, 2017.

[27] S. Hustiu, M. Lupaşcu, S. Popescu, A. Burlacu and M. Kloetzer: Stable hovering architecture for nanoquadcopter applications in indoor environments, *22nd International Conference on System Theory, Control and Computing (ICSTCC)*, 2018.

[28] E. Karapistoli, F.N. Pavlidou, I. Gragopoulos, I. Tsetsinas: An overview of the IEEE 802.15.4a Standard, *IEEE Communications Magazine*, vol. 48(1), pp. 47-53, 2010.

[29] Loco-positioning-system for Crazyflie 2/0, available at https://www.bitcraze.io/getting-started-with-the-loco-positioning-system/

[30] A. Alarifi, A. Al-Salman, M.Alsaleh, A. Alnafessah, S. Al-Hadhrami, M.A. Al-Ammar, H.S. Al-Khalifa: Ultra Wideband Indoor Positioning Technologies: Analysis and Recent Advances, *Sensors*, vol. 16(5), 2016.

[31] C. Gentner and M. Ulmschneider: Simultaneous Localization and Mapping for Pedestrians using Low-Cost Ultra-Wideband System and Gyroscope, *IEEE Int. Conf. on Indoor Positioning and Indoor Navigation*, Saporo, Japan, 2017.

[32] Shi G. and Ming Y.: Survey of Indoor Positioning Systems Based on Ultra-wideband (UWB) Technology, *Wireless Communications, Networking and Applications*, Springer, pp. 1269–1278, 2016.