POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

KIERUNEK: Automatyka i Robotyka (AIR)
SPECJALNOŚĆ: Robotyka (ARR)

PRACA DYPLOMOWA MAGISTERSKA

Sterowanie rojem robotów w środowisku symulacyjnym

Robot swarm control in simulated environment

AUTOR:
Piotr Bogdoł

PROWADZĄCY PRACĘ:
dr inż. Wojciech Domski, K29W04D02

WROCŁAW 2021

Spis treści

Rozdział 1

Wstęp

Od dawna człowiek tworzył coraz to nowsze narzędzia ułatwiające mu pracę. Wraz z rozwojem techniki narzędzia stawały się coraz bardziej skomplikowane, ale równocześnie w znacznie większym stopniu ułatwiały pracę. Rewolucja przemysłowa pozwoliła zamienić narzędzia na całe maszyny, natomiast rewolucja związana z powstaniem komputera dała możliwość na stworzenie maszyny działającej w pewnym stopniu niezależnie od człowieka. Dalszy rozwój badań doprowadził do powstania robotów, urządzeń mogących wykonywać ciężkie prace w sposób automatyczny, bez konieczności angażowania do nich ludzi. Dalsze dążenia do wykorzystania robotów w coraz to nowszych i bardziej skomplikowanych zadaniach doprowadziło do konieczności rozwoju technologii pozwalających na współpracę wielu robotów w celu realizacji zadań, które dla pojedynczego robota mogłyby być zbyt wymagające lub czasochłonne.

Obecnie, obszar badań nad systemami składającymi się z wielu mobilnych robotów znacznie się powiększył od czasu najwcześniejszych prac na ten temat, publikowanych w latach osiemdziesiątych ubiegłego wieku [8], [4]. Na ogólnym poziomie, zagadnienie systemu składającego się z wielu mobilnych robotów można podzielić na dwie kategorie: system roju robotów oraz system robotów współpracujących. W systemach robotów współpracujących, roboty posiadają wiedzę o obecności innych robotów w środowisku i wspólnie ze sobą działają w oparciu o stan, operacje lub zdolności innych robotów z zespołu do osiągnięcia wspólnego celu. Zespoły robotów współpracujących mogą składać się z różnych rodzajów robotów, przez co każdy z robotów może realizować odmienne zadania. W systemach rojów robotów, pojedyncze roboty wykonują własne zadania przy minimalnej wiedzy na temat innych członków roju. Takie systemy są typowe przy założeniu, że występuje w nich duża liczba jednorodnych robotów mobilnych. Każdy taki robot wykorzystuje lokalne prawa sterowania do generowania globalnie zgodnych zachowań całego zespołu, przy niewielkiej i jawnej komunikacji między robotami [18].

Wiele badań dotyczących sposobu kontrolowania roju robotów czerpie inspiracje z obserwacji świata przyrody. Zachowania stadne wielu gatunków zwierząt, między innymi mrówek, pszczół i ptaków pozwalają im na osiągnięcie imponujących wyników ich działania. Przykładem może być zdolność mrówek do zbiorowego przenoszenia dużej zdobyczy lub budowanie dużych mrowisk, posiadających rozległe i skomplikowane sieci tuneli. Te niezwykłe możliwości skłoniły badaczy do próby odtworzenia zachowań zwierząt z wykorzystaniem systemów składających się z wielu robotów [18].

Rozwój badań nad zachowaniem i sterowaniem zespołami robotów pozwolił na możliwość ich praktycznego wykorzystania w rozmaitych zastosowaniach. Przykładowymi zadaniami mogą być zarządzanie kontenerami w porcie [2], eksploracja pozaplanetarna [19], poszukiwania i ratownictwo [13], wydobycie minerałów, transport, konserwacja przemysłowa

i gospodarcza. Wykorzystanie wielu współpracujących ze sobą robotów w wymienionych dziedzinach może przynieść wiele korzyści wynikających choćby z możliwości równoległej pracy robotów. Możliwe jest wyróżnienie kilku głównych kategorii zadań, które mogą być realizowane przez zespoły robotów [18].

Pierwszy typ zadań polega na żerowaniu to znaczy na zbieraniu przez roboty ładunków rozproszonych po całej przestrzeni roboczej imitujących pożywienie i przetransportowaniu ich do jednego lub wielu obszarów imitujących bazę. Taki typ zadania często wykorzystywany jest do testowania algorytmów sterowania rojów robotów o dużej liczebności, w których nie występują mocne zależności między poszczególnymi zadaniami robotów oraz nie jest wymagana ich ścisła synchronizacja. Zadanie żerowania jest tradycyjnym zadaniem ze względu na to, że posiada bliską analogię do systemów biologicznych. Z zadaniem żerowania stowarzyszone jest również zadanie przeszukiwania polegające na odwiedzeniu przez roboty wszystkich miejsc danego obszaru w celu ich zbadania, znalezienia pewnych obiektów rozmieszczonych w środowisku lub wykonania pewnych akcji. Obydwa typy zadań posiadają odzwierciedlenie w rzeczywistym świecie, na przykład przy usuwaniu toksycznych odpadów rozrzuconych na znacznym obszarze [20].

Kolejna grupa zadań nawiązuje do zachowań stadnych zwierząt i obejmuje zachowanie odpowiedniej formacji przez roboty, na przykład podczas ucieczki. Problem wymaga poruszania się przez roboty wzdłuż wspólnej ścieżki ruchu (nie odłączania się od stada). Przemieszczanie się robotów w formacji powoduje potrzebę utrzymywania odpowiednich względnych pozycji podczas manewru. Formując zadanie często zakłada się, że roboty mają jedynie minimalną liczbę czujników, efektorów, zdolność komunikacji między sobą oraz moc obliczeniową. Kluczowe w przypadku zadania utrzymania formacji jest określenie lokalnych praw kontroli poszczególnych robotów, które generują pożądany zachowanie zbiorowe. Tego typu zadania znajdują zastosowanie w różnych dziedzinach, na przykład w zastosowaniach militarnych, czy podczas przeszukiwania jakiegoś obszaru. Odpowiednie utrzymanie formacji pozwala na wykorzystanie czujników o ograniczonym zasięgu, zamontowanych na poszczególnych robotach, w celu przebadania dużego obszaru [3].

Inną kategorią zadań, wymagających synchronizacji działania poszczególnych robotów, są zadania przesuwania ciężkiego ładunku oraz skoordynowana manipulacja. Ten typ zadania jest szczególnie popularny podczas demonstrowania możliwości współpracy wielu robotów, ponieważ wymaga ścisłej koordynacji i kooperacji. Przesuwanie ciężkich ładunków polega na przemieszczaniu przez kilka robotów jednego dużego ładunku, który z założenia nie może być przemieszczony przez pojedynczego robota. Założenia zadania mogą nie tylko wyznaczać miejsce docelowe, ale również ścieżkę po jakiej ładunek ma zostać przesunięty. Zadanie skoordynowanej manipulacji zakłada podnoszenie i przenoszenie obiektu do wyznaczonego miejsca. Ta grupa zadań nadaje się do badania strategii zakładających silną kooperacje wielu robotów. Można również znaleźć kilka rzeczywistych zastosowań takiego zadania, na przykład transport dużych przedmiotów w środowisku przemysłowym [5].

Ostatnia wyróżniona grupa zadań związana jest z wykonywaniem przez roboty własnych, niezależnych celów we wspólnej przestrzeni roboczej, na przykład przewożenie ładunków przez pojedyncze roboty. Tego typu problem zwykle pojawia się kiedy przestrzeń, w której pracują roboty zawiera „wąskie gardła", na przykład roboty poruszają się wspólną jezdnią. W zadaniu otwarta przestrzeń może być traktowana jako zasób, którym roboty muszą się dzielić w możliwe najwydajniejszy sposób, unikając kolizji i zakleszczeń [18].

Krytycznym problemem pojawiającym się przy zastosowaniu mobilnych zespołów robotów w zadaniach związanych z transportem ładunków jest koordynowanie ruchów wielu robotów współpracujących w tej samej przestrzeni roboczej. Niezależnie od misji robo-

tów, muszą one efektywnie współdzielić przestrzeń roboczą aby zapobiegać zakłóceniu zadań pozostałych robotów. Możliwe są różne rozwiązania problemu koordynacji ruchu w zależności od celów jakie zostały postawione przed zespołem robotów. W niektórych przypadkach ścieżki robotów są wyraźnie planowane i koordynowane z wyprzedzeniem, w innych przypadkach większy nacisk kładzie się na mechanizmy unikania kolizji przy mniejszym nacisku na planowanie ścieżek. W jeszcze innych przypadkach dokonuje się planowania wstępnego, natomiast uwaga koncentrowana jest na koordynowaniu ruchów robotów w czasie rzeczywistym przy użyciu podejścia wykorzystującego różnego typu reguły opisujące żądane zachowanie robotów [14].

Problem planowania ścieżek dla wielu robotów polega na określeniu ścieżek pozwalających wszystkim robota na przemieszczenie się z położenia początkowego do zadanego położenia końcowego w bezpieczny sposób. Oznacza to, że ścieżki muszą być wyznaczone w taki sposób aby unikając kolizji z otoczeniem i innymi robotami. Wiele algorytmów zarządzających zespołami robotów kładzie nacisk na rozwiązanie tego rodzaju problemu. Ma to na celu poprawienie efektywności pracy wielu robotów we wspólnym środowisku. Jednym ze sposobów kategoryzacji jest podział na podejście scentralizowane i zdecentralizowane. Podejście scentralizowane polega na wykorzystaniu globalnych informacji i stworzeniu planu ruchu uwzględniając wszystkie aspekty zadania. Powoduje to najczęściej uzyskanie rozwiązania optymalnego, jednakże wymaga rozwiązania zadania o dużej złożoności. Podejście zdecentralizowane kładzie nacisk na wydajność obliczeniową poprzez podział zadania i rozwiązywanie niektórych aspektów problemu niezależnie od innych. Zazwyczaj powoduje to zmniejszenie jakości rozwiązania, jednak pozwala na rozwiązywanie zadań o większym stopniu złożoności [14].

Przykładowym rozwiązaniem problemu planowania ścieżek robotów opartych na podejściu scentralizowanym jest metoda bazująca na mapie dróg [15]. Metoda polega na wielofazowym planowaniu ścieżek robotów z wykorzystaniem grafu oraz drzewa rozpinającego. Pozwala ona na uzyskanie bezkolizyjnych ścieżek w środowisku pracy robota. Na początku tworzony jest graf, którego wierzchołkami są możliwe do osiągnięcia przez robota pozycje wynikające z mapy dróg. Krawędzie reprezentują możliwość przejazdu między poszczególnymi pozycjami określonymi we wierzchołkach. Graf zawiera jednocześnie pozycje początkowe i końcowe robotów. Na podstawie grafu tworzone jest drzewo rozpinające możliwe ruch z początkowego, wybranego wierzchołka do kolejnych połączonych wierzchołków. Tak przygotowane drzewo nie może zawierać cykli. Wybór wierzchołka, który jest korzeniem drzewa nie jest narzucony, jednak dąży się do wyboru takich drzew, dla których liczba liści jest maksymalna. Przykładowo, jako węzeł początkowy można wybrać węzeł znajdujący się w środku mapy. Pierwsza faza algorytmu polega na stworzeniu planu przesuwającego wszystkie roboty do pozycji znajdujących się w liściach drzewa wzdłuż ścieżek nie powodujących kolizji. W drugiej fazie, roboty przesuwane są do pozycji, z których mogą osiągnąć pozycje końcową nie blokują innych robotów. Osiągane jest to dzięki rozpatrywaniu na początku robotów, których docelowa pozycja znajduje się najgłębiej w drzewie (najdalej od korzenia). Roboty, które nie znajdują się w pozycjach docelowych, przesuwane są do niej w trzeciej fazie algorytmu. Trzy fazy algorytmu pozwalają na poruszanie się tylko jednemu robotowi na raz. Ostatnią fazą jest optymalizacja planu, tak aby pozwalał on na poruszanie się kilku robotów w jednoczesnym czasie, jeżeli nie powoduje to kolizji.

Innym podejściem do problemu planowania ścieżek jest podejście zdecentralizowane [7]. Polega ono na planowaniu ścieżek robotów z wykorzystaniem priorytetów. Każdemu robotowi przydzielany jest zadany priorytet. Może się to odbywać na zasadzie losowania lub oceny mobilności robota (robot o niższej mobilności dostaje wyższy priorytet). Na

początku wyznaczana jest ścieżka dla robota o najwyższym priorytecie przy zastosowaniu dowolnej metody wyznaczania ścieżki dla pojedynczego robota. Następnie, dla kolejnych robotów o niższych wartościach priorytetu wyznacza się ścieżki biorąc pod uwagę wcześniej wyznaczone ścieżki dla robotów o wyższych priorytetach. Tak powstałe ścieżki traktuje się jako ruchome przeszkody. Ponieważ w opisywanym podejściu wymagane jest uwzględnienie czasu, przestrzeń reprezentowana jest jako lista przestrzeni konfiguracyjnych w zadanych chwilach czasu. Ścieżki w przestrzeni uwzględniającej czas są wyznaczane przy pomocy algorytmu przeszukiwania grafu widoczności.

Metoda zdecentralizowana opiera się na technice koordynowania ścieżek [12]. Polega ona na podziale problemu na problem planowania ścieżki oraz prędkości robota. Dekompozycja problemu pozwala na zmniejszenie złożoności obliczeniowej przez nie uwzględnianie dodatkowego wymiaru czasowego podczas planowania ścieżki robota. Algorytm składa się z dwóch faz. W pierwszej zostają zaplanowane ścieżki dla wszystkich robotów, niezależnie od siebie, na przykład z wykorzystaniem algorytmu przeszukiwania grafu widoczności. W fazie drugiej uwzględnia się czas, przez zaplanowanie prędkości robotów w kolejnych segmentach trasy, tak aby nie dochodziło do kolizji między robotami. W takim podejściu ścieżki ustalone w pierwszej fazie nie są już zmienianie w kolejnej.

Ściśle związanym z tematem planowania ścieżek dla wielu robotów jest kwestia koordynowania ruchu wielu robotów. W odróżnieniu od zadania planowania ścieżek dla wielu robotów lub podejścia opartego na koordynowaniu ścieżek, które skupiały się na tworzeniu kompletnego planu poruszania się dla wszystkich robotów, techniki koordynowania ruchu koncentrują się na podejściach zdecentralizowanych i on-line. Wykorzystują one na przykład reguły kontroli ruchu pozwalające robotom unikać lub rozwiązywać konflikty pojawiające się podczas realizacji ścieżki przez robota. W zastosowaniach kontroli ruchu wielu robotów, pojedyncze roboty nadal mają niezależne pozycje początkowe oraz docelowe i muszą poruszać się tak aby unikać kolizji. Grupa robotów musi jednocześnie poruszać się synchronicznie zgodnie z wcześniej zdefiniowanymi ograniczeniami ruchu dla całego zespołu [14].

Przykładową metodą wykorzystującą podejścia polegającego na koordynowaniu ruchu jest kontrola ruchu [9]. Opisywana problematyka dotyczy kontroli dużej liczby autonomicznych wózków transportowych wykorzystywanych w fabryce. Podejście oparte jest na określeniu reguł poruszania się wózków we wspólnym środowisku. Wózki posiadają różne priorytety, które są zmieniane w kolejnych chwilach czasu tak aby zapewnić średnio równy priorytet dla wszystkich pojazdów. W pierwszej kolejności planowany jest ruch dla wózków o wyższych priorytecie. Każdy pojazd wykonuje planowanie kroku niezależnie od innych. Jako krok uznaje się ruch lub jego brak (oczekiwanie). W takim podejściu, krok jaki zostanie zaplanowany wynika z ustalonych reguł kontroli ruchu. Dla jednakowego problemu można przyjąć różne polityki kontroli ruchu.

1.1 Teza pracy

Celem niniejszej pracy jest przedstawienie opracowanych algorytmów sterowania: algorytmu z heurystycznym dobieraniem ścieżek robotów i algorytmu opartego o optymalizację numeryczną oraz porównanie opracowanych algorytmów pod względem wydajności czasowej oraz długości przebytej drogi przez wszystkie roboty.

1.2 Opis zawartości pracy

W rozdziale 2 zostały opisane zagadnienia teoretyczne wykorzystywane w dalszych częściach pracy. Znajduje się w nim dokładny opis zadania przyjętego do realizacji, szczegóły dotyczące wykorzystanego algorytmu A* oraz metody Model Predictive Control (MPC) korzystającej z optymalizacji numerycznej. Kolejny rozdział został poświęcony opisowi oprogramowania wykorzystanego do implementacji oraz testowania rozwiązań postawionego zadania. W czwartym rozdziale zostały umieszczone informacje na temat sposobu działania sterownika pojedynczego robota. Kolejny rozdział prezentuje opis realizacji sterownika centralnego. Zostały w nim przedstawione funkcje realizowane przez sterownik centralny wraz z algorytmami wykorzystanymi do rozwiązania problemu planowania ścieżek dla roju robotów. W rozdziale zostały również przedstawione wynik badań wydajności poszczególnych rozwiązań oraz wnioski z nich wynikające. Ostatni rozdział przedstawia krótkie podsumowanie całej pracy.

Rozdział 2

Zagadnienia teoretyczne

W rozdziale zostały przedstawione zagadnienia teoretyczne wykorzystane w pracy. Na początku zostało szczegółowo opisane zadanie przyjęte do realizacji. Została podana treść podstawowego zadania oraz warunki poprawnego wykonania zadania. Następnie zostały wprowadzone dodatkowe założenia pozwalające uszczegółowić i uprościć podstawowe zadanie. W kolejnej części rozdziału zostanie opisany algorytm A*. W ostatniej części rozdziału został przedstawiony algorytm optymalizacji numerycznej opierający się na predykcji zachowania systemu.

Środowisko wykorzystane podczas badań zostało przedstawione na rysunku 2.1. Składa się ono z jednej strefy centralnej oraz czterech stref na obrzeżach przestrzeni roboczej. Celem podstawowego zadania jest przetransportowanie wszystkich ładunków l_i, dla $i = 0, 1, \ldots, n$, znajdujących się w centralnej strefie do stref znajdujących się na obrzeżach wyznaczonego obszaru. Strefy są identyfikowane za pomocą kolorów. Strefa centralna z_{black} została oznaczona czarnym kolorem natomiast strefy na obrzeżach z_c, dla $c \in \{red, green, blue, yellow\}$ zostały oznaczone odpowiednio kolorami czerwonym, zielonym, niebieskim i żółtym. Każdy z ładunków l_i jest opisany za pomocą współrzędnych położenia $p_i = (x, y)$ oraz koloru określającego jedną z czterech stref docelowych c_i. Transport ładunków l_i odbywa się za pomocą roju składającego się z robotów r_j, dla $j = 0, 1, \ldots, m$. Każdy z robotów r_j opisany jest za pomocą współrzędnych obecnego położenia (x, y), współrzędnych położenia końcowego (x_f, y_f) oraz informacji na temat przewożonego ładunku lub jego braku $c'_r = c \cup e$

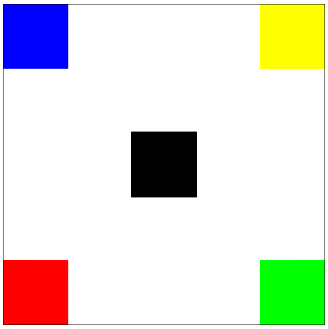
$$r_j = [x, y, x_f, y_f, c'_r].$$

Robot r_j w jednej chwili czasu może przewozić tylko jeden ładunek. Zadanie uznaje się za zakończone w momencie gdy wszystkie ładunki l_i ze strefy centralnej z_{black} zostaną przewiezione do stref o odpowiednim kolorze z_c i wszystkie roboty r_j osiągną wyznaczone pozycje końcowe x_f, y_f

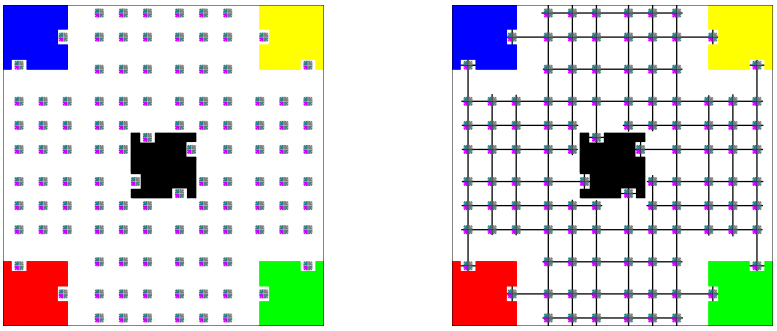
$$\begin{cases} p_i \in z_c \\ x = x_f \quad dla \quad c = c_i \\ y = y_f \end{cases} .$$

W celu uszczegółowienia i uproszczenia opisanego zadania przyjęto kilka dodatkowych założeni związanych z poruszaniem się robotów. Przyjęto, że roboty należą do klasy (2,0), mogą swobodnie obracać się w miejscu oraz poruszać się po linii prostej w przód i w tył. Ponadto, poza strefami, roboty mogą się przemieszczać tylko między wyznaczonymi węzłami P

$$(x, y) \in P = [p_1, p_2, \ldots, p_k] \vee (x, y) \in z,$$



Rysunek 2.1 Sceneria podstawowego zadania



(a) (b)

Rysunek 2.2 a) położenie wyznaczonych węzłów P, pomiędzy którym mogą poruszać się roboty, b) sceneria przyjętego zadania

gdzie $z = z_c \cup z_{black}$. Rozmieszczenie węzłów zostało przedstawione na rysunku 2.2a. Strefy zostały podzielone na sektory. Strefa identyfikowana czarnym kolorem z_{black} została podzielona na cztery części. Pozostałe strefy z_c zostały podzielone na dwie części $s \in \{black_1, \ldots, black_4, red_1, red_2, green_1, green_2, blue_1, blue_2, yellow_1, yellow_2\}$. W jednym sektorze nie mogą znajdować się jednocześnie dwa roboty

$$(x_a, y_a) \in s \wedge (x_b, y_b) \in s \;\; dla \;\; a = b.$$

Do każdego z sektorów strefy robot może wjechać tylko przez wyznaczony punkt węzłowy $p_s \in P$, znajdujący się na skraju sektora. Robot znajdujące się w jednym sektorze nie może bezpośrednio przejechać do drugiego sektora. Aby robot mógł wjechać do innej części strefy musi on wyjechać ze strefy w wyznaczonym miejscu, a następnie przemieścić się do punktu wjazdowego do innej części strefy. Między wyznaczonymi węzłami roboty mogą się poruszać jedynie po wyznaczonych liniach. Dodatkowo, robot nie może przemieścić się do następnego wyznaczonego węzła $p' \in P$ dopóki w tym węźle znajduje się inny robot

$$(x_a, y_a) = p' \wedge (x_b, y_b) = p' \;\; dla \;\; a = b.$$

Scenariusz przyjętego zadania przy uwzględnieniu wszystkich dodatkowych założeń został przedstawiony na rysunku 2.2b.

2.1 Algorytm A*

Algorytm A* jest algorytmem heurystyczny pozwalający na wyszukanie najkrótszej ścieżki w grafie. Graf składa się z wierzchołków u_i połączonych ze sobą krawędziami $c(u_i, u_j)$, dla

$i \neq j$. Każdej z krawędzi c odpowiada koszt przejścia między wierzchołkami $w(u_i, u_j)$, które są połączone przez daną krawędź. Algorytm opiera się na wykorzystaniu funkcji heurystycznej $h(u)$ pozwalającej na estymowanie kosztu ścieżki z obecnego wierzchołka u do wierzchołka docelowego u_g. Działanie algorytmu polega na iteracyjnym wyborze wierzchołków spośród zbioru wierzchołków dostępnych $\mathbb{O}$. Wierzchołek jest dostępny kiedy istnieje krawędź łącząca go z już przebadanym wierzchołkiem i została wyliczona dla niego wartość funkcji

$$f(u) = g(u) + h(u), \tag{2.1}$$

gdzie $g(u)$ określa koszt przebycia wyznaczonej ścieżki z wierzchołka początkowego u_s do wierzchołka u, natomiast $h(u)$ jest funkcją heurystyczną. W kolejnych iteracjach, spośród wierzchołków znajdujących się w zbiorze wierzchołków dostępnych wybiera się ten wierzchołek, dla którego wartość funkcji $f(u)$ jest najmniejsza i dodaje się go do zbioru wierzchołków przebadanych $\mathbb{C}$. Ponieważ koszty przejścia między kolejnymi wierzchołkami w są znane, wartość funkcji $g(u)$ można wyliczyć poprzesz sumowanie kosztów przejścia między wierzchołkami wzdłuż ścieżki wyznaczonej od wierzchołka początkowego u_s do bieżącego wierzchołka u. Postać funkcji heurystycznej $h(u)$ nie jest jednoznacznie określona i najczęściej zależy od rozpatrywanego problemu. Odpowiedni dobór funkcji heurystycznej $h(u)$ istotnie wpływa na wydajność i optymalność rozwiązania. Jeżeli funkcja heurystyczna $h(u)$ jest dopuszczalna, to znaczy spełnia warunek oszacowania od dołu

$$\forall u \; h(u) \leqslant h^*(u),$$

gdzie $h^*(u)$ jest funkcją określającą koszt dla optymalnej ścieżki między wierzchołkiem u, a wierzchołkiem końcowym u_g, to rozwiązanie zwrócone przez algorytm A* będzie optymalne. Jednak spełnienie tego warunku nie gwarantuje optymalnego czasu znalezienia rozwiązania [6].

Podczas inicjalizacji algorytm A* należy do zbioru wierzchołków dostępnych dodać wierzchołek startowy u_s, obliczyć wartość funkcji $f(u_s) = h(u_s)$ oraz przyjąć zbiór wierzchołków przebadanych jako zbiór pusty $\mathbb{C} \in \{\emptyset\}$.

1. Jeżeli $\mathbb{O} \in \{\emptyset\}$ to nie istnieje ścieżka między u_s i u_g. W innym przypadku należy wybrać wierzchołek

   $$u^* = \min_{u \in \mathbb{O}} f(u). \tag{2.2}$$

   Usunąć wierzchołek u^* ze zbioru wierzchołków dostępnych $\mathbb{O}$ i dodać do zbioru wierzchołków przebadanych $\mathbb{C}$.

2. Jeżeli wybrany wierzchołek u^* jest wierzchołkiem docelowym $u^* = u_g$, należy zwrócić ścieżkę między u_s a u^*. W innym wypadku należy uaktualnić stan wierzchołków u takich, że

   $$\exists c(u^*, u) \tag{2.3}$$

3. Mogą wystąpić trzy różne przypadki:

   (a) Jeżeli wierzchołek $u \in \mathbb{O}$ to należy sprawdzić czy wartość funkcji $g(u)$ wzdłuż obecnie znanej ścieżki z wierzchołka u_s do wierzchołka u nie jest większa od sumy wartości funkcji $g(u^*)$ i wartości kosztu przejścia $w(u^*, u)$ między wierzchołkami u^* i u

   $$g(u^*) + w(u^*, u) < g(u). \tag{2.4}$$

Jeśli warunek (2.4) jest spełniony to następuje zmiana ścieżki między u_s a u i zostają uaktualnione wartości funkcji

$$g(u) = g(u^*) + w(u^*, u) \tag{2.5}$$

oraz

$$f(u) = g(u^*) + w(u^*, u) + h(u). \tag{2.6}$$

(b) Jeżeli wierzchołek $u \in \mathbb{C}$ to należy sprawdzić warunek (2.4). Jeśli warunek (2.4) jest spełniony to następuje zmiana ścieżki między u_s a u oraz uaktualnienie są wartości funkcji zgodnie z równaniami (2.5) i (2.6). Dodatkowo wierzchołek u jest usuwany ze zbioru wierzchołków przebadanych $\mathbb{C}$ i dodawany do zbioru wierzchołków dostępnych $\mathbb{O}$.

(c) W pozostałych przypadkach wierzchołek u dodawany jest do zbioru wierzchołków dostępnych $\mathbb{O}$, zostają zainicjalizowane wartość funkcji zgodnie z równaniami (2.5) i (2.6) oraz zapisywana jest ścieżka między u_s i u.

2.2 Algorytm optymalizacji numerycznej

Zasada działania algorytmu optymalizacji numerycznej bazuje na metodzie Model Predictive Control [10] wykorzystywanej w procesie sterowania różnego typu systemami. W celu predykcji oraz optymalizacji przyszłego zachowania systemu wykorzystany jest model procesu. Predykcja i optymalizacja odbywa się w skończonym horyzoncie czasowym. Postać modelu można wyrazić wzorem

$$x(n+1) = f(x(n), u(n)), \tag{2.7}$$

gdzie $f : X \times U \to X$, $x(n)$ i $u(n)$ są odpowiednio stanem systemu i sygnałami sterującymi w chwili czasu t_n, natomiast $x(n+1)$ jest stanem systemu w następnej chwili czasu t_{n+1} wynikającym ze stanu $x(n)$ i wartości wejść $u(n)$. Zaczynając w stanie $x(n)$ i stosując dowolną sekwencję sterowań $u(0), \ldots, u(N-1)$ z horyzontem czasowym o długości $N \geqslant 2$ można, z wykorzystaniem modelu (2.7), wyliczyć przewidywaną trajektorię x_u zdefiniowaną jako

$$x_u(0) = x(n), \quad x_u(k+1) = f\left(x_u(k), u(k)\right), \quad k = 0, \ldots, N-1. \tag{2.8}$$

Postępowanie w taki sposób pozwala na szacowanie $x_u(k)$ dla stanu systemu $x(n+k)$ w przyszłej chwili czasu t_{n+k}. Umożliwia to przewidywanie zachowania systemu w dyskretnych chwilach czasu $t_n, \ldots, t_{n+N}$ w zależności od zastosowanej sekwencji sterowań $u(0), \ldots, u(N-1)$. Wykorzystując optymalizację numeryczną możliwe jest wyznaczenie ciągu sterowań $u(0), \ldots, u(N-1)$, którego zastosowanie pozwoli osiągnąć przewidywany stan systemu $x_u(k)$ możliwie zbliżony do żądanego stanu systemu x_*, dla $k = 0, \ldots, N-1$. W celu dokonania optymalizacji należy zdefiniować funkcję $l(x_u(k), u(k))$ pozwalającą na określenie kosztu wynikającego z niewłaściwych wartości stanu $x_u(k)$ w porównaniu do wartości oczekiwanego stanu x_*. Funkcja może również określać koszt związany z różnicą między wartościami sterowań $u(k)$, a wartością sterowania referencyjnego u_*. Rozwiązanie problemu sterowania optymalnego

$$\min_{u(0),\ldots,u(N-1)} J(x(n), u(\cdot)) = \sum_{k=0}^{N-1} l(x_{u^*}(k), u^*(k)) \tag{2.9}$$

dostarcza sekwencje sterowań optymalnych $u^*(0), \ldots, u^*(N-1)$. Ostatecznie sterowanie jakie należy zastosować w chwili t_n wynosi $u(n) = u^*(0)$ [10].

Podsumowując, algorytm optymalizacji numerycznej można przedstawić w trzech krokach. W każdych kolejnych chwilach czasu t_n, $n = 0, 1, 2, \ldots$ należy

1. Dokonać pomiaru stanu systemu $x(n)$.

2. Ustawić $x_0 = x(n)$, a następnie rozwiązać problem sterowania optymalnego

$$\min J_N(x_0, u(\cdot)) = \sum_{k=0}^{N-1} l(x_u(k, x_0), u(k)), \tag{2.10}$$

gdzie $u(\cdot) \in \mathbb{U}^N(x_0)$ oraz $x_u(0, x_0) = x_0$ i $x_u(k+1, x_0) = f\left(x_u(k, x_0), u(k)\right)$, przy ograniczeniach

$$g\left(x_u(k, x_0), u(k)\right) \leqslant 0. \tag{2.11}$$

Otrzymaną, optymalną sekwencje sterowań oznaczyć przez $u^*(\cdot) \in \mathbb{U}^N(x_0)$.

3. Zdefiniować wartość sterowań dla obecnej iteracji n

$$\mu_N(x(n)) = u^*(0) \tag{2.12}$$

oraz zastosować tak zdefiniowaną wartości sterowań w obecnej chwili czasu t_n.

Rozdział 3

Oprogramowanie

W rozdziale zostało przedstawione oprogramowanie wykorzystane do wytworzenia sterowników oraz ich symulacji. Pierwsza część rozdziału poświęcona jest Robot Operating System [1], [11]. Zostały w niej omówione podstawowe elementy z jakich może składać się system utworzony za pomocą biblioteki i narzędzia dostarczonych przez ROSa oraz opisane mechanizmy komunikacji między elementami. W kolejnej części został przedstawiony symulator ARGoS [17], [16], wykorzystany w badaniach symulacyjnych opracowanych algorytmów sterowania rojem robotów. Została omówiona architektura ARGoSa oraz opisane elementy wykorzystane w symulacjach. W ostatniej części został omówiony sposób komunikacji pomiędzy elementami ROSa oraz symulatorem ARGoS.

3.1 ROS

Robot Operating System (ROS) jest zbiorem narzędzi oraz bibliotek pozwalających na tworzenie aplikacji robotycznych. ROS jest rozpowszechniany na licencji wolnego oprogramowania. Zapewnia miedzy innym abstrakcje sprzętową, niskopoziomowe sterowniki urządzeń, narzędzia do pisania, debugowania i uruchamiania kodu na wielu różnych komputerach,a także zarządzanie pakietami. ROS wspiera różne języki programowania, między innymi C++ oraz Python. ROS posiada architekturę grafu, w którym poszczególne elementy połączone są ze sobą na zasadzie jeden do jednego. Komunikacja pomiędzy elementami systemu utworzonego za pomocą ROS-a może odbywać się na trzy sposoby: za pomocą usług, z wykorzystaniem współdzielonej pamięci lub poprzez wysyłania informacji przez strumień danych [1].

Podstawowymi elementami systemu utworzonego za pomocą ROSa są węzły (z ang. *Nodes*). Węzeł posiada własny przepływ sterowania i można go utożsamić z procesem w standardowym systemie operacyjnym. Każdy węzeł musi zostać napisany z wykorzystaniem biblioteki klienta ROS, w zależności od wybranego języka programowania mogą to być na przykład `roscpp` lub `rospy`. Wykorzystanie tych bibliotek pozwala na zastosowanie różnych sposobów komunikacji pomiędzy węzłami. Zaletą użycia węzłów jest możliwość wytworzenia systemu za pomocą wielu prostych węzłów realizujących podstawowe operacje oraz przesyłających między sobą niezbędne informacje zamiast tworzenia systemu składającego się z jednego dużego modułu. Użycie wielu prostych węzłów ułatwia proces debugowania oraz rozszerzenie lub modyfikację systemu [11].

Korzystanie z węzłów wymaga uprzedniego uruchomienia programu ROS Master, którego zadaniem jest zapewnienie rejestrowania oraz przechowywanie danych dotyczących wszystkich węzłów w systemie. ROS Master pośredniczy podczas wyszukiwania i łączenia

się ze sobą poszczególnych węzłów. Jedną z części ROS Mastera jest serwer parametrów (z ang. *Parameter Server*). Pozwala on na przechowywanie różnego typu danych w centralnym miejscu. Każdy węzeł może uzyskać dostęp do danych, odczytywać je, zapisywać, kasować oraz modyfikować. Jest to jeden ze sposobów komunikacji między węzłami. Taki sposób komunikacji nie jest jednak wydajny. Nie zaleca się korzystania z niego w wypadku konieczności częstej modyfikacji danych. Jednym z przykładowych zastosowań jest wykorzystanie serwera parametrów do przechowywania różnego rodzaju parametrów obecnej konfiguracji systemu [1].

Kolejnym sposobem komunikacji wykorzystywanym w ROSie jest mechanizm publikacji i subskrypcji. Polega on na przesyłaniu wiadomości (z ang. *Messages*) między węzłami za pomocą tematów (z ang. *Topics*). Wiadomości są prostymi strukturami zawierającymi określony typ danych, pozwalające na przesyłanie zbioru danych między węzłami. W wiadomościach można wykorzystać predefiniowane typy danych lub określić własny typ wiadomości poprzez stworzenie odpowiedniego pliku `msg`. Kiedy węzeł wysyła wiadomość powoduje to opublikowanie jej w odpowiednim temacie. Kiedy węzeł subskrybuje dany temat, może on odebrać wiadomość. Węzeł publikujący oraz węzeł subskrybujący dany temat nie są świadome swojego istnienia. Możliwa jest sytuacja kiedy węzeł publikuje wiadomości do tematu, który nie ma subskrybenta oraz kiedy węzeł subskrybuje temat, do którego żaden inny węzeł nie publikuje wiadomości. Brak konieczności posiadania przez węzły informacji o innych węzłach, które publikują/subskrybują dany temat pozwala na łatwe rozszerzanie systemu stworzonego przy użyciu ROSa. Na przykład, jeżeli w systemie znajduje się węzeł obsługujący czujnik i publikuje otrzymane pomiary do zdefiniowanego tematu, możliwe jest dodanie nowego węzła korzystającego z wyników pomiarów bez konieczności zmieniania kodu węzła obsługującego czujnik oraz węzłów, które wcześniej już subskrybowały temat zawierający pomiary z czujnika. Dodatkowo, taka własność pozwala na „podmienianie" węzłów czy to w przypadku modyfikacji ich działania, czy w wypadku uszkodzenia sprzętu, na którym działa dany węzeł bez konieczności restartowania i rekonfiguracji całego systemu. Warto również zauważyć, że model komunikacji typu publikacja/subskrypcja pozwala na tworzenie komunikacji typu wiele do wielu, natomiast architektura ROSa opiera się na komunikacji typu jeden do jednego.

Innym modelem komunikacji między węzłami są usług (z ang. *Services*). Ten model komunikacji polega na wysyłaniu zapytań przez węzeł klienta i odsyłaniu odpowiedzi przez węzeł pełniący rolę serwera. Pozwala to na uzyskanie mechanizmu wywoływania zdalnej procedury. Węzeł serwera udostępnia swoją usługę poprzez zgłoszenie możliwości jej wywołania do ROS Mastera. Kiedy węzeł klienta zgłosi potrzebę skorzystania z usługi, ROS Master inicjalizuje połączenie pomiędzy węzłem klienta a węzłem serwera na zasadzie jeden do jednego. Możliwe jest, że nie została jeszcze zgłoszona usługa, którą potrzebuje węzeł klienta, na przykład w sytuacji kiedy serwer jest wyłączony. W takim przypadku klient może zaczekać aż odpowiednia usługa zostanie utworzono i zgłoszona do ROS Mastera. Typy danych przesyłane przez klienta do serwera oraz zwracane przez serwer do klienta muszą być ściśle określone podczas zgłoszenia usługi do ROS Mastera. Możliwe jest wykorzystanie typów danych dostarczonych w pakietach ROSa lub zdefiniowanie własnych typów poprzez stworzenie odpowiedniego pliku `srv`. W odróżnieniu od asynchronicznej komunikacji typu publikacja/subskrypcja, usługi wymagają oczekiwania przez klienta na odpowiedz serwera. Może prowadzić to do zmniejszenia wydajności systemu, na przykład kiedy wiele węzłów chce w jednym czasie skorzystać z tej samej usługi.

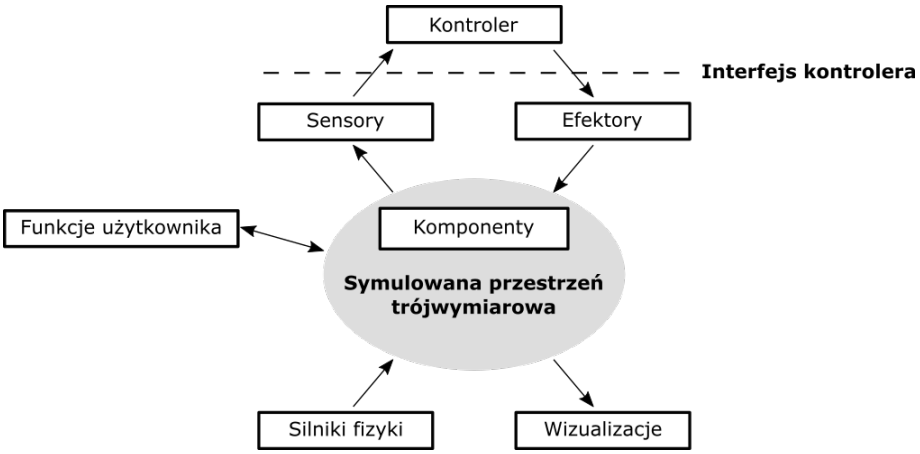
Podczas opracowywania oraz symulowania rozwiązań przedstawionych w niniejszej pracy został wykorzystany ROS w wersji Melodic 1.14.9 wraz z systemem operacyjnym Ubuntu 18.04.5 LTS. ROS został użyty do stworzenie węzłów odpowiadających za stero-

wanie poszczególnymi robotami oraz węzła ze sterownikiem centralnym. Dostarczył również mechanizmów pozwalających na komunikację sterowników robotów ze sterownikiem centralnym, jak również sterowników z symulatorem.

3.2 ARGoS

Autonomus Robots Go Swarming (ARGoS) jest symulatorem stworzonym z myślą o badaniu narzędzi i strategii kontroli heterogenicznych rojów robotów. ARGoS posiada architekturę modułową. Roboty, sensory, efektory, wizualizacje oraz silniki fizyczne są zaimplementowane w postać modułów definiowalnych przez użytkownika. Pozwala to na uzyskanie dużego stopnia rozszerzalności symulatora. ARGoS pozwala na wielokrotną implementację różnego rodzaju modułów oraz wyboru modułów, które zostaną wykorzystane w bieżącym eksperymencie symulacyjnym poprzez odpowiednią modyfikacje pliku konfiguracyjnego XML. Jedną z unikalnych cech ARGoSa jest możliwość podzielenia symulowanej przestrzeni na podprzestrzenie, dla których można zastosować różne silniki fizyki. Roboty mogą swobodnie przemieszczać się między poszczególnymi podprzestrzeniami [17].

Schemat architektury ARGoSa został przedstawiony na rysunku 3.1. Wszystkie białe bloki przedstawione na schemacie związane są z elementami architektury, które mogą zostać zdefiniowane przez użytkownika. Symulowana przestrzeń trójwymiarowa przedstawiona na środku schematu jest zbiorem struktur danych zawierających kompletny stan symulacji. Stan symulacji jest zbiorem danych, do których należą informacje na temat pozycji oraz orientacji wszystkich elementów w przestrzeni takich jak przeszkody czy roboty, jak również informacje na temat różnych części, wyposażenia oraz urządzeń, na przykład zbioru kolorowych diod LED. Wszystkie dane przechowywane w symulowanej przestrzeni są zorganizowane w formie komponentów. Istnieje kilka typów predefiniowanych komponentów, które mogą zostać dostosowane przez użytkownika. Jeśli zajdzie taka potrzeba, istnieje również możliwość dodania nowych komponentów. Każdy komponent przechowuje informacje na temat konkretnego aspektu symulacji. Sensory oraz układy wykonawcze są elementami architektury, które mają dostęp do stanu symulacji. Sensory posiadają możliwość odczytywania danych z symulowanej przestrzeni trójwymiarowej, natomiast układy wykonawcze posiadają możliwość modyfikacji danych w symulowanej przestrzeni. Sensory i wspomniane układy wykonawcze zostały zaprojektowane tak, aby miały dostęp tylko do niezbędnych komponentów. Silniki fizyki odpowiadają za aktualizacje, w kolejnych momentach czasu, danych związanych z poszczególnymi komponentami znajdującymi się w odpowiedniej podprzestrzeni. Do jednej wydzielonej podprzestrzeni może być przydzielony tylko jeden silnik fizyki. Nie ma możliwości nakładania się części podprzestrzeni. Wizualizacje są modułami odpowiedzialnym za wyświetlanie odpowiedniej reprezentacji graficznej danych odczytanych z symulowanej przestrzeni. ARGoS pozwala również na zdefiniowanie przez użytkownika funkcji, które będą wywoływane w pewnych punktach głównej pętli symulatora. Użytkownik może dodać własne funkcjonalności wykonywa po każdym kroku symulacji. Dodatkowo możliwe jest zdefiniowanie własnych warunków zakończenia się eksperymentu symulacyjnego. Wykorzystanie tego mechanizmu pozwala, na przykład generować różnego rodzaju wykresy oraz statystyki wykorzystywane w późniejszym etapie analizy wyników, jak również wpływanie na przebieg symulacji poprzez, na przykład dodawanie lub usuwanie różnych obiektów po wykonaniu różnych działań przez roboty. Ostatnim modułem są kontrolery robotów. Określają one w jaki sposób robot będzie się zachowywał podczas symulacji. ARGoS dostarcza własny, abstrakcyjny interfejs



Rysunek 3.1 Architektura ARGoS (na podstawie [16])

pozwalający na napisanie logiki kontrolera za pomocą języka C++ [16].

W badaniach symulacyjnych opracowanych algorytmów sterowania rojem zostały wykorzystane dwa elementy: ładunki oraz roboty. Ładunek został zamodelowany jako walec o niewielkich rozmiarach oraz masie, który może zostać złapany za pomocą chwytaka oraz przewieziony przez robota. Na górnej części walca znajduje się dioda LED, której kolor identyfikuje strefę docelową ładunku. Drugim elementem symulacji są roboty. Jako model robota został wykorzystany model nazwany w symulatorze `foot-bot`. Symulowany robota zawiera wiele różnych czujników, między innymi czujniki zbliżeniowe, LIDAR, czujnik koloru podłoża, kamerę wielokierunkową. Posiada również chwytak. Ponadto robot należy do klasy (2,0) co spełnia założenie przyjęte w zadaniu. Podczas eksperymentów symulacyjnych została wykorzystana kamera wielokierunkowa pozwalająca na określenie położenia ładunku względem robota oraz koloru ładunku. Został również wykorzystany chwytak, który umożliwił przewożenie ładunków między strefami. Dodatkowo symulator umożliwia uzyskanie globalnej lokalizacji robota, co pozwoliło na zaimplementowanie przemieszczania się robotów.

3.3 Komunikacja między ROS i ARGoS

W celu wykorzystania symulatora ARGoS do przetestowania sterowników napisanych z wykorzystaniem pakietów dostarczonych wraz z ROSem należało wykorzystać specjalnie przygotowany pakiet ROSowy dostępny na stronie poświęconej symulatorowi ARGoS. Pakiet zawiera kilka plików wymaganych przez ROS oraz kontroler napisany w języku C++ z wykorzystaniem interfejsu programowego wymaganego przez ARGoSa. Równocześnie logika kontrolera utworzona jest na zasadzie węzła ROSa, dzięki czemu możliwe jest wykorzystanie różnych mechanizmów komunikacji dostępnych z poziomu tego systemu. Głównymi zadaniami kontrolera są odczytywanie danych otrzymywanych z czujników wykorzystanych podczas symulacji i przesyłaniu ich do innych węzłów oraz sterowanie efektorami w sposób zgodny z otrzymanymi instrukcjami od innych węzłów. Domyślnie komunikacja między węzłem kontrolera, a innymi węzłami odbywa się za pomocą mechanizmu publikacji i subskrypcji. Dla każdego wykorzystanego czujnika oraz efektora tworzony jest odpowiedni temat. Umożliwia to odczyt danych oraz sterowanie robotem za pomocą dowolnego węzła i pozwala oddzielić logikę sterownika robota od interfejsu programistycznego wymaganego przez ARGoSa. Daje to możliwość łatwego przełączenia się

między robotem symulowanym przez ARGoS a rzeczywistym robotem, bez konieczności modyfikowania kodu węzła implementującego logikę sterownika robota. Z uwagi na wysoki stopień modułowości systemu ARGoS, każdy symulowany robot posiada dedykowany kontroler. Powoduje to, że dla każdego symulowanego robota tworzona jest oddzielna grupa tematów pozwalająca na jego sterowanie. Dla części robotów możliwe jest zastosowanie kontrolera napisanego z wykorzystaniem bibliotek ROSowych. Dla pozostałej części kontrolerów, które zawierają logikę sterowania robotem wykorzystywany jest wyłącznie tylko interfejs ARGoSa (bez możliwości komunikacji z węzłami ROSa).

Rozdział 4

Sterownik pojedynczego robota

Sterownik roju robotów może zostać przedstawiony za pomocą dwupoziomowej struktury. Pierwszy, wyższy poziom składa się ze sterownika centralnego, którego zadaniem jest planowanie oraz zarządzanie zadaniami wykonywanymi przez roboty. Drugi, niższy poziom składa się ze sterowników lokalnych odpowiedzialnych za właściwe wykonywanie zadań zleconych przez sterownik centralny. Schemat struktury został przedstawiony na rysunku 4.1. W rozdziale został dokładnie omówiony sterownik pojedynczego robota. Zostały opisane funkcje jakie są realizowane przez sterownik, następnie został przedstawiony sposób realizacji ruchu robota. Na końcu został omówiony sposób komunikacji między poszczególnymi poziomami architektury sterowania.

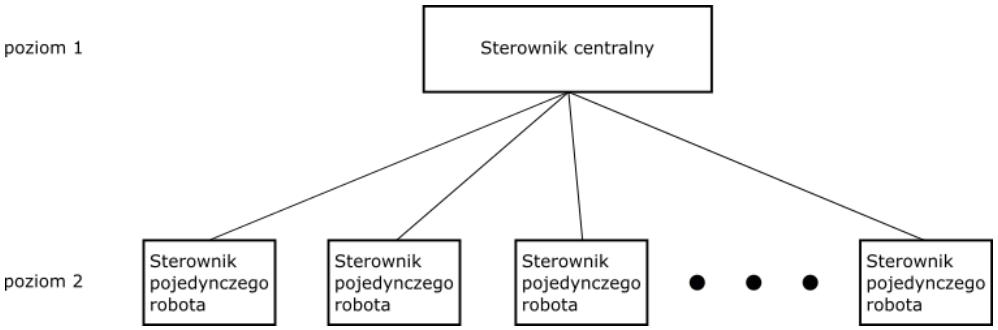
4.1 Funkcje realizowane przez sterownik

Głównym zadaniem sterownika pojedynczego robota jest odpowiednia realizacja zadań otrzymanych ze sterownika centralnego oraz informowanie sterownika centralnego o wyniku wykonanego działania. Polecenia otrzymywane ze sterownika centralnego mogą zostać pogrupowane na pięć kategorii:

- przemieszczenie się do wyznaczonego punktu węzłowego p,
- wjazd do wyznaczonej strefy,
- podniesienie najbliższego ładunku znajdującego się w danej części strefy lub zawiadomienie o braku ładunku w danej części strefy,
- odłożenie przewożonego ładunku na pozycje otrzymaną ze sterownika centralnego,
- opuszczenie części strefy, w której aktualnie znajduje się robot.

Dodatkowo sterownik centralny może zażądać przesłania obecnej pozycji robota wraz z orientacją lub statusu przewożonego ładunku (identyfikator) jeśli robot przewozi ładunek albo informacji o braku ładunku.

Przemieszczanie się między punktami węzłowymi p pozwala na poruszanie się robotom pomiędzy wyznaczonymi strefami. Roboty mają za zadanie przejechać z bieżącej pozycji do pozycji wyznaczonej przez sterownik centralny, przy czym wyznaczony węzeł docelowy musi być zgodny z przyjętą w założeniach siatką połączeń węzłów. Oznacza to, że roboty nie mogą poruszać się między dowolnymi węzłami, a jedynie najbliższymi, połączonymi czarnymi liniami. Ruch robota musi odbywać się wzdłuż założonych linii. Sterownik pojedynczego robota nie odpowiada za weryfikacje poprawności wyboru węzła docelowego czy



Rysunek 4.1 Schemat struktury sterownika roju robotów

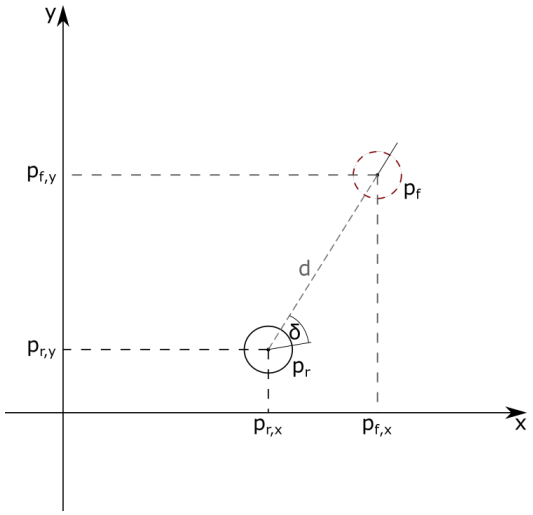
też sprawdzanie wystąpienia kolizji z innym robotem. Odpowiada jedynie za osiągnięcie wyznaczonego punktu z ustaloną dokładnością.

Wjazd do wyznaczonej strefy polega na rozpoznaniu przez robota numeru sektora strefy, do której może wjechać z aktualnie zajmowanego punku węzłowego oraz przemieszczeniu do punktu węzłowego znajdującego się wewnątrz rozpoznanej części strefy. Sterownik pojedynczego robota nie dokonuje sprawdzenia czy w sektorze strefy, do której obecnie wjeżdża jest inny robot. Sterownik centralny odpowiada za brak kolizji i odpowiednią liczbę robotów w strefach. Kontrola liczby robotów w danym sektorze odbywa się poprzez blokowania węzła wjazdowego, tak aby roboty nie mogły otrzymać polecenie przemieszczenia się do tego węzła dopóki dana część strefy jest zajęta. W celu ułatwienie rozpoznania przez sterownik pojedynczego robota dla jakiej strefy powinien sprawdzać sektor, jako argument polecenia przekazywany jest identyfikator strefy.

Kiedy robot przemieści się do węzła wewnątrz sektora centralnego możliwe jest wykonanie polecenia podniesienia najbliższego ładunku znajdującego się w danej części strefy. Jeżeli w obecnie zajmowanym przez robota sektorze nie ma już ładunków, robot zwraca informacje o braku ładunków do sterownika centralnego. Na podstawie tych informacji sterownik centralny określa, do których sektorów strefy centralnej należy jeszcze wysyłać roboty. Jeśli w sektorze znajduje się przynajmniej jeden ładunek, robot podjeżdża do niego oraz podnosi go. Po wykonaniu manewru zwraca informacje na temat identyfikatora podniesionego ładunku, potrzebnego sterownikowi centralnemu do późniejszego wyznaczenia ścieżki.

Roboty znajdujące się w sektorach docelowych stref mogą wykonać polecenie związane z odłożeniem ładunku na wyznaczoną pozycję. Sterownik centralny zlicza liczbę ładunków umieszczonych przez roboty w danym sektorze i na tej podstawie określa pozycję na jaką ma zostać odłożony ładunek. Jeżeli liczba ładunków w danej części strefy przekroczy wyznaczony próg to tą część strefy uznaje się za przepełnioną i roboty nie są już więcej do niej kierowane. Podczas wykonywania polecenia odłożenia ładunku, robot przemieszcza się do wyznaczonego miejsca oraz zwalnia ładunek. Po wykonaniu polecenia zwracana jest informacja przez robota o zakończeniu manewru.

Po wykonaniu komendy podniesienia lub odłożenia ładunku musi nastąpić polecenie opuszczenia sektora. Sterownik lokalny robota na podstawie zapisanego numeru sektora oraz identyfikatora strefy określa pozycję węzła wewnątrz sektora oraz pozycję węzła wyjazdowego. Następnie przemieszcza się po kolei do wcześniej określonych pozycji węzłów. Aby nie nastąpiła przypadkowa kolizja z ładunkami znajdującymi się w strefie, robot wycofuje się do punku węzłowego wewnątrz strefy nie wykonując żadnego obrotu. Dopiero kiedy znajdzie się na pozycji pierwszego z wyznaczonych węzłów wykonuje obrót. Następnie przemieszcza się po linii prostej do węzła wyjazdowego.



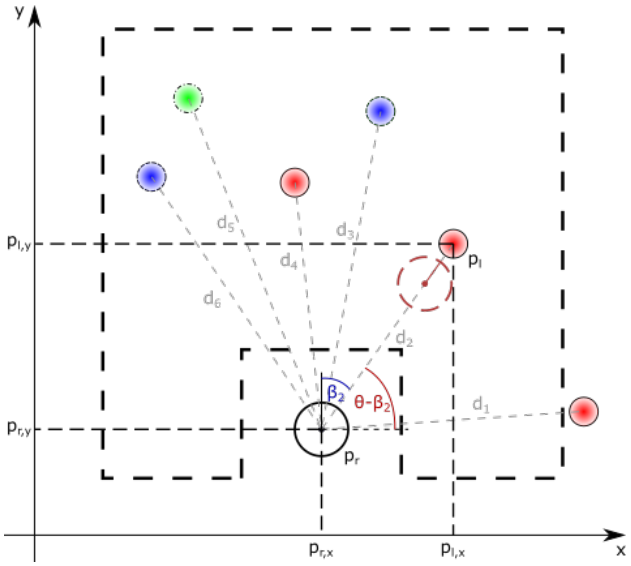
Rysunek 4.2 Pierwsza sytuacja podczas realizacji ruchu robota

4.2 Sposób realizacji ruchu robota

Każda z głównych funkcjonalności sterownika pojedynczego robota wymaga wykonania ruchu. Założenie, że roboty wykorzystane w zadaniu należą do klasy (2,0), pozwala na wykonywanie przez roboty obrotu w miejscu. Dzięki temu ruch robota można podzielić na dwie części. Pierwszy etap polega na znalezieniu kąta o jaki powinien obrócić się robot, aby być na wprost punktu docelowego oraz dokonania obrotu zadany kąt.Następnie wymagane jest wyznaczenie odległości do zadanego punktu oraz przemieszczenie robota w linii prostej do wyznaczonej pozycji. Model ruchu robota został uproszczony. Nie jest rozpatrywana dynamika robota, a jedynie wraz ze zmniejszaniem odległości kątowej lub liniowej robota do celu, zmniejszana jest prędkość kątowa lub liniowa robota, zgodnie z przyjętymi progami.

Funkcje realizowane przez sterownik robota wymagają rozpatrzenia dwóch sytuacji. Pierwsza z nich polega na przemieszczeniu się do punktu wyznaczonego przez sterownik centralny. Taka sytuacja została przedstawiona na rysunku 4.2. Zakładając, że pomiar lokalizacji globalnej robota nie zawiera znaczących błędów, możliwe jest wyznaczenie odległości z obecnej pozycji robota do punktu docelowego jako długość między obecną pozycją robota a wyznaczoną pozycją końcową $d = ||p_f - p_r||$. Dzięki wyznaczeniu różnic położenia dla poszczególnych współrzędnych możliwe jest wyznaczenie kąta obrotu robota $\delta = arctg\left(\frac{\delta y}{\delta x}\right)$.

Druga sytuacja, przedstawiona na rysunku 4.3, wynika z konieczności podjechaniu przez robota do najbliższego ładunku, którego pozycja musi zostać określona na podstawie czujników. Do lokalizacji i identyfikacji ładunku, symulowane roboty wykorzystują kamerę wielokierunkową zwracającą kolor ładunku, odległość d_i do poszczególnych ładunków oraz kąt β_i wyrażony względem robota a pozycją ładunku. Warto zauważyć, że jeśli w danej części strefy znajduje się więcej niż jeden ładunek danego koloru, kamera nie dokonuje ich bezpośredniego rozróżnienia. W celu podniesienie przez robota ładunku znajdującego się w wewnątrz danej części strefy, najbliżej węzła wjazdowego, najpierw należy sprawdzić czy ładunek, dla którego kamera zwróciła najmniejszą odległość, znajduje się w odpowiedniej części strefy. Następnie, po wybraniu ładunku do podniesienia a jeszcze przed wykonaniem ruchu przez robota, należy określić pozycję ładunku względem globalnego układu współrzędnych. Globalne współrzędne ładunku można obliczyć poprzez dodanie do poszczególnych współrzędnych robota odpowiedniego przesunięcia



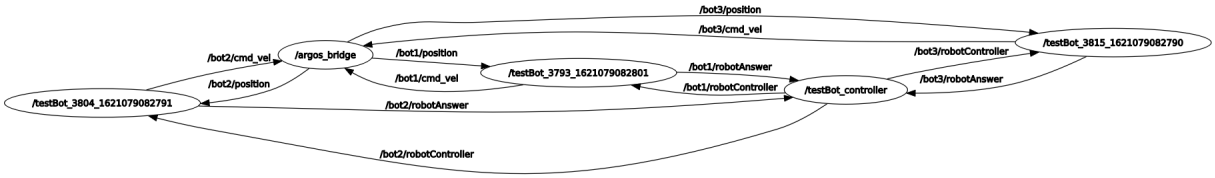
Rysunek 4.3 Druga sytuacja podczas realizacji ruchu robota

$(cos(\theta-\beta_l)\cdot d_l, sin(\theta-\beta_l)\cdot d_l)$. Wykorzystanie współrzędnych globalnych ładunku pozwoli na uniknięcie błędu związanego ze zmianą ładunku, który jest podnoszony, po rozpoczęciu ruchu robota oraz konieczności implementowania dodatkowego algorytmu śledzenia ładunków w celu ich rozróżnienia podczas zmiany pozycji robota. Ponadto określenie pozycji ładunku względem globalnego układu współrzędnych daje możliwość zastosowania algorytmu ruchu przedstawionego w pierwszej rozpatrywanej sytuacji z uwzględnieniem dodatkowej konieczności zatrzymania się robota w pewnej odległości przed ładunkiem, na tyle dużej aby zapobiec kolizji, ale jednocześnie na tyle małej aby robot miał możliwość złapania ładunku.

4.3 Komunikacja

Sposób wymiany informacji między poszczególnymi sterownikami odpowiada strukturze sterownika roju robotów. Oznacza to, że komunikacja występuje tylko między sterownikiem centralny a sterownikami poszczególnych robotów. Roboty nie mogą przekazywać informacji bezpośrednio między sobą. Możliwa jest jednak sytuacja kiedy dane otrzymane przez sterownik centralny od jednego robota wpłyną na polecenie wysłane przez sterownik centralny do innego robota, na przykład zostanie wysłane polecenie pozostania w obecnym punkcie węzłowym.

Komunikacja między poszczególnymi elementami sterownika roju robotów odbywa się z wykorzystaniem mechanizmu publikuj/subskrybuj dostarczonego wraz z pakietem ROS. Każdy z robotów posiada dwa niezależne tematy. Pierwszy z tematów, którego sterownik pojedynczego robot jest tylko subskrybentem, służy do publikowania przez sterownik centralny poleceń do wykonania przez danego robota. Drugi temat służy do publikowania przez robota wyników po wykonaniu ostatniego polecenia. Może do niego publikować wiadomości tylko robot, do którego temat jest przypisany, a subskrybować tylko sterownik centralny. Jak było wcześniej wspomniane, model komunikacja nie zakłada żadnego tematu wspólnego dla wszystkich sterowników pojedynczego robota. Każdy robot subskrybuje również dodatkowy temat pozwalający przesłać informacje na temat awaryjnego zatrzymania działania robota. Wiadomości wysyłane i obierane za pomocą tematów mają postać ciągów znaków w postaci formatu JSON. Taki format pozwala w łatwy sposób wysłać li-



Rysunek 4.4 Graf połączeń między poszczególnymi elementami sterownika roju robotów oraz symulatorem

stę wartości argumentów poleceń o różnej długości dla różnych poleceń. Graf połączeń między sterownikiem centralnym (węzeł o nazwie `testBot_controller`) a poszczególnymi sterownikami robotów dla 3 robotów został przedstawiony na rysunku 4.4. Na grafie uwzględniono również węzeł o nazwie `argos_bridge`, który odpowiada za komunikacje sterowników z symulatorem.

Rozdział 5

Sterownik centralny

Architektura sterownika roju robotów, zaprezentowana w poprzednim rozdziale, składała się z dwóch poziomów. Niższy poziom, na który składały się sterowniki robotów został omówiony w rozdziale 4. W niniejszym rozdziale został natomiast opisany wyższy poziom składający się ze sterownika centralnego. W pierwszej części rozdziału została pokrótce przedstawiona funkcjonalność sterownika centralnego. Następnie w kolejnych dwóch częściach rozdziału zostały opisane dwie opracowane metody planowania ścieżek robotów. Zostały omówione zasady ich działania oraz przedstawiony sposób badania metod wraz z wynikami. W ostatniej części zostało wykonane porównanie wyników uzyskanych dla obydwóch metod.

5.1 Funkcje realizowane przez sterownik

Sterownik centralny ma za zadanie zarządzać zachowaniem całego roju robotów w celu zrealizowania wyznaczonego zadania. Do głównych funkcji realizowanych przez sterownik należą:

- śledzenie położeń i orientacji robotów,
- wysyłanie poleceń dotyczących ruchu do poszczególnych sterowników robotów,
- wykrywanie i zapobieganie występowania kolizji między robotami,
- wyznaczanie ścieżki ruchu między obecną pozycją robota a pozycją docelową.

Zarządzanie rojem robotów przez sterownik centralny odbywa się poprzez wysyłanie do poszczególnych robotów odpowiednie polecenia. Głównymi czynnikami wpływającymi na rodzaj polecenia oraz wartości w nim przesłane (na przykład wartością mogą być współrzędne, do których robot ma się przemieścić) są zaplanowane ścieżki robotów i obecne położenie poszczególnych robotów. Od obecnego położenia robota zależy między innymi czy robot może, czy nie wjechać do strefy, jakiego koloru jest to strefa lub czy robot będzie miał za zadanie podnieś ładunek, czy go odłożyć. Z tego powodu sterownik centralny przechowuje informacje o położeniu oraz orientacji wszystkich robotów oraz aktualizuje te dane za każdym razem gdy lokalne sterowniki robotów zwrócą nowe wartości po poprawnym wykonaniu ruchu.

Położenia robotów są również wykorzystywane podczas sprawdzania czy robot może wykonać następny krok zgodnie z zaplanowaną ścieżką. Jeżeli na jego drodze znajduje się inny robot, polecenie ruchu do następnej wyznaczonej pozycji w ścieżce nie zostaje

przesłane do sterownika robota. Pozwala to na unikanie kolizji, jednak może prowadzić do blokady robotów. Wraz z poszczególnymi metodami planowania ścieżki zostały wprowadzone pewne mechanizmy unikania blokady robotów, omówione dokładniej w dalszych częściach rozdziału.

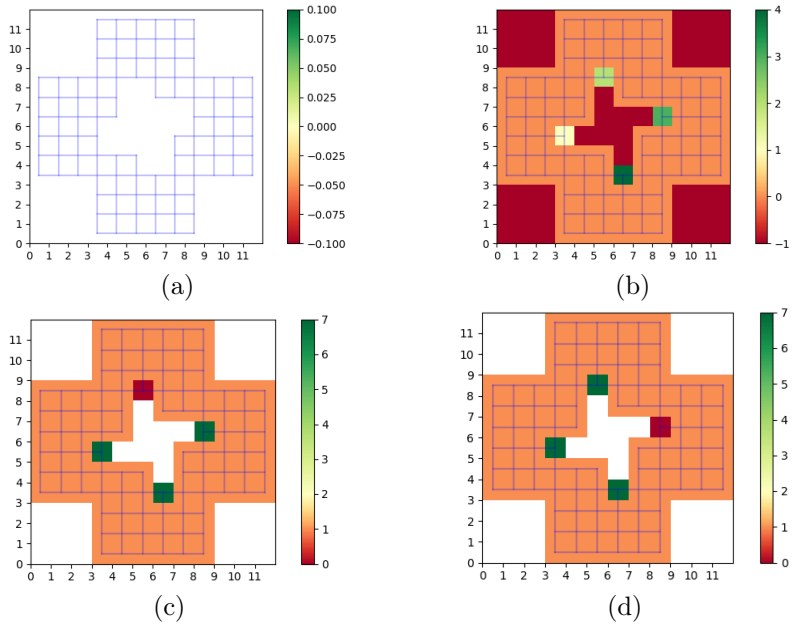
Najistotniejszą funkcją sterownika centralnego jest planowanie ścieżek dla poszczególnych robotów. Wyznaczenie odpowiednich ścieżek wpływa na wydajność całego roju oraz zmniejsza prawdopodobieństwo wystąpienia kolizji i zakleszczeń. W idealnym przypadku długość poszczególnych ścieżek jest minimalna, a ich przebieg w pełni zapobiega kolizją między robotami oraz robotami a przeszkodami. Jednak planowanie ścieżek, szczególnie dla rojów o dużej liczbie robotów, jest bardzo złożonym zadaniem. W dalszej części rozdziału zostaną omówione dwie metody planowania ścieżek: metoda oparta o zmodyfikowany algorytm A* oraz metoda oparta o algorytm optymalizacji numerycznej.

5.2 Planowanie ścieżek oparte o zmodyfikowany algorytm A*

Pierwsza opracowana metoda planowania ścieżek robotów wykorzystuje zmodyfikowany algorytm A*. Algorytm A* w podstawowej wersji został omówiony w rozdziale 2.1. Wykorzystanie algorytmu A* jako metody planowania ścieżek robotów dla zadania przedstawionego w rozdziale 2 wymagało dostosowania algorytmu. Punkty węzłowe P zostały utożsamione z wierzchołkami grafu. Połączenia między poszczególnymi punktami węzłowymi (przedstawione za pomocą czarnych linii) określały krawędzie grafu. Koszty przejścia między poszczególnymi punktami węzłowymi zostały wyrażone za pomocą mapy kosztów. Modyfikacja algorytmu A* polegała na wprowadzeniu mapy kosztów, której wartości zmieniały się w trakcie działania sterownika. Dodatkowo każdy robot posiada własną mapę kosztów. Jako funkcja heurystyczna została wykorzystana norma taksówkowa $h(u) = |x_g - x| + |y_g - y|$, gdzie (x_g, y_g) są współrzędnymi położenia docelowego punktu węzłowego u_g, natomiast (x, y) współrzędnymi położenia punktu węzłowego u. Planowanie ścieżek odbywało się po każdym wykonanym ruchu przez robota. Jako punkt węzłowy, do którego ma się przemieścić robot w danym momencie czasu uznawany był pierwszy punkt ścieżki. Metoda planowania ścieżki wybierała również część strefy, do której robot ma się udać, na podstawie odległości do poszczególnych węzłów wjazdowych oraz liczby robotów zmierzających do danego węzła wjazdowego.

5.2.1 Mapa kosztów

Przed każdym wyszukiwaniem ścieżki między obecnie zajmowanym przez robota punktem węzłowym a punktem docelowym obliczana jest mapa kosztów dla danego robota. Wartości kosztów zależą od położenia robotów oraz zaplanowanych ścieżek. Wartość kosztu przejścia do zablokowanych puntów mapy (wyjechanie poza dozwolony obszar lub punkty znajdujące się w strefach) wynosi nieskończoność. Podstawowy koszt przejścia między dwoma węzłami wynosi w_n. Jeżeli przez punkt węzłowy przechodzą zaplanowane ścieżki innych robotów, to wartości kosztu przejścia do tego punktu z węzłów sąsiednich zwiększana jest o liczbę przecinanych ścieżek. Jeżeli inny robot znajduje się w punkcie węzłowym to do wartości kosztu przejścia do tego węzła dodawana jest wartość w_c. Dodatkowo podnoszona jest wartość kosztu przejścia dla dwóch następujących po sobie punktach węzłowych w wyznaczonych ścieżkach tych robotów. Wartość kosztu punktu, do którego robot ma się przemieścić w następnym kroku zwiększana jest o w_{o1}, natomiast w drugim kroku

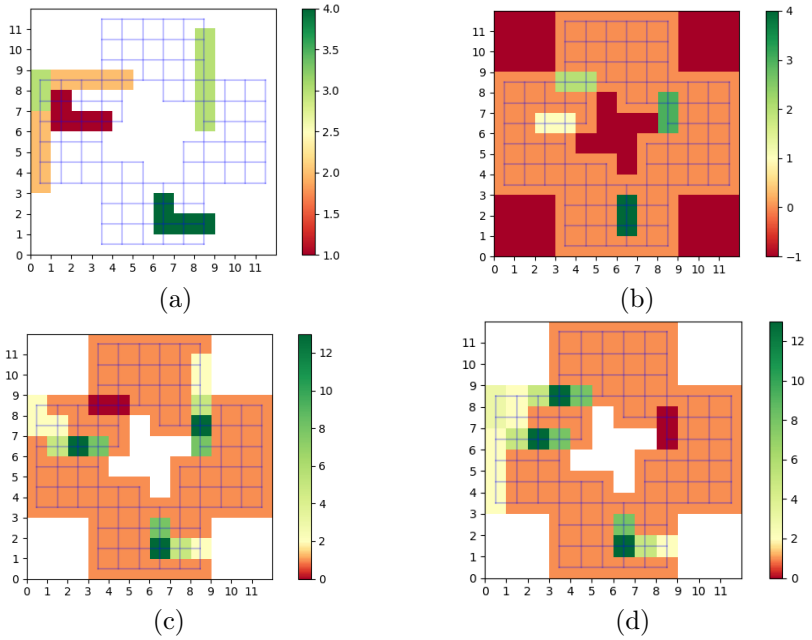


Rysunek 5.1 a) mapa z zaznaczonymi ścieżki robotów, b) mapa z zaznaczonymi pozycjami robotów, c) mapa kosztu dla robota numer 2, d) mapa kosztu dla robota numer 3

o w_{o2}. Poszczególne wartości dodawane do mapy kosztów w zależności od zaplanowanych ścieżek i pozycji robotów zostały dobrane eksperymentalnie.

Przykładowe mapy kosztów dla dwóch robotów wraz z zaznaczonymi zajmowanymi pozycjami oraz zaplanowanymi ścieżkami zostały przedstawione na rysunkach 5.1 i 5.2. W obydwu przestawionych przypadkach łączna suma robotów wynosiła cztery. Rysunek 5.1 przedstawia początkową fazę działania sterownika roju robotów, kiedy ścieżki nie są jeszcze wyznaczone, a wszystkie roboty znajdują się w strefie centralnej. Mapy kosztów zostały przedstawione dla robota 2 i 3. Przyglądając się mapie dla robota 3 można zauważyć, że wartość kosztu przejścia do pól zajmowanych przez inne roboty wynosi 7, do pól pustych wynosi 1, a do pola zajmowanego przez robota 3 wynosi 0. Rysunek 5.2 przedstawia ten sam eksperyment symulacyjny po upływie kilku cykli planowania. Można zauważyć, że dla wszystkich robotów zostały zaplanowane ścieżki. Ścieżki na mapie zostały zaznaczone kolorami odpowiadającymi numerom robotów. Przecięcia ścieżek zostały oznaczone kolorami odpowiadającymi sumie numerów robotów, których ścieżki się przecięły. Porównując mapę kosztów dla robota o numerze 3 z rysunków 5.1d i 5.2d, łatwo spostrzec, że wartości na mapie uległy zmianie. Widać również w jaki sposób zaplanowane ścieżki i pozycje robotów wpływają na wartości kosztów przejścia między poszczególnymi punktami węzłowymi. Ścieżka robota, dla którego wyliczana jest mapa kosztu nie jest brana pod uwagę podczas wyliczania kosztów.

Należy również zauważyć, że każdy z robotów na mapie położenia robotów zajmuje dwa pola. Pierwsze z nich to pole, na którym aktualnie się znajduje. Drugie pole przedstawia natomiast punkt węzłowy, do którego dany robot przemieści się w najbliższym ruchu. Mechanizm blokowania punktów, do których roboty mają przejechać pozwala na unikanie kolizji między robotami. Robot przed wykonaniem ruchu sprawdza czy dane pole nie jest zajęte przez innego robota.



Rysunek 5.2 a) mapa z zaznaczonymi ścieżki robotów, b) mapa z zaznaczonymi pozycjami robotów, c) mapa kosztu dla robota numer 2, d) mapa kosztu dla robota numer 3

5.2.2 Problem zakleszczeń

Wykorzystanie wielu robotów pracujących w tej samej przestrzeni roboczej może prowadzi do blokowania się na wzajem poszczególnych robotów. Wraz ze wzrostem liczby zastosowanych robotów do wykonywania zadania rośnie prawdopodobieństwo występowania blokad nazywanych zakleszczeniami. Do zakleszczenia dochodzi, kiedy dwa roboty do wykonania swojego ruchu potrzebują zwolnienia punktu węzłowego, który jest zajmowany przez drugiego robota. Przykładowo do zakleszczenie może dość wtedy kiedy jeden z robotów chce jechać w lewo, a wyznaczona ścieżka prowadzi przez węzeł zajmowany przez drugiego robota. Drugi robot natomiast chce jechać w prawo oraz zaplanowana ścieżka przechodzi przez węzeł zajmowany przez pierwszego robota. Ponieważ wyznaczone ścieżki były najkrótsze, nie dojdzie do ich zmiany i przez to obydwa roboty będą oczekiwać na wykonanie ruchu przez pozostałego robota i odblokowanie zajętego węzła. W przypadku tak powstałej blokady metoda doboru ścieżek korzystająca ze zmodyfikowanego algorytm A* posiada możliwość tymczasowego zwiększania kosztu przejścia dla pól, zajmowanych przez inne roboty. W celu wykrycia powstania zakleszczenia, sprawdzana jest możliwość wykonania przez danego robota ruchu. Jeżeli przez określony czas robot nie może wykonać ruchu, w pierwszym kroku wykonuje się ponowne planowanie ścieżki bez modyfikacji parametrów mapy kosztu. Jeżeli takie działanie nie przyniesie skutku, zwiększa się najpierw dwukrotnie wartości kosztu przejścia, a następnie sześciokrotnie. Tak duże zwiększenie wartości kosztu zazwyczaj prowadzi do zmiany wcześnie wyznaczonej ścieżki i ominięcia zatoru.

Przy większej liczbie robotów (podczas przeprowadzonych testów dla 12 i szczególnie 16 robotów) pojawiały się zakleszczenia związane ze zbyt dużą liczbą robotów, które miały jednakowy punkt docelowy. Zdarzało się to w końcowych fazach testów, kiedy ładunki pozostały tylko w jednej części centralnej strefy. W takiej sytuacji wszystkie roboty, które nie przewoził ładunku, były kierowane do wjazdu. Jednocześnie robot z ładunkiem miał za zadanie opuścić strefę centralną i przemieścić się do odpowiedniej strefy identyfikowanej kolorem ładunku. Ponieważ liczba robotów zmierzających do punktu wjazdowego (będącego jednocześnie punktem wyjazdowym), była większa od liczby punktów węzłowych

sąsiadujących z puntem wjazdowym, została zablokowana możliwość wyjazdu ze strefy. Zwiększanie parametrów związanych z kosztem przejścia w takiej sytuacji nie przyniesie oczekiwanego rezultatu z uwagi na to, że pola ze sobą sąsiadują. Powstanie takiej sytuacji doprowadziło do całkowitego zakleszczenia robotów.

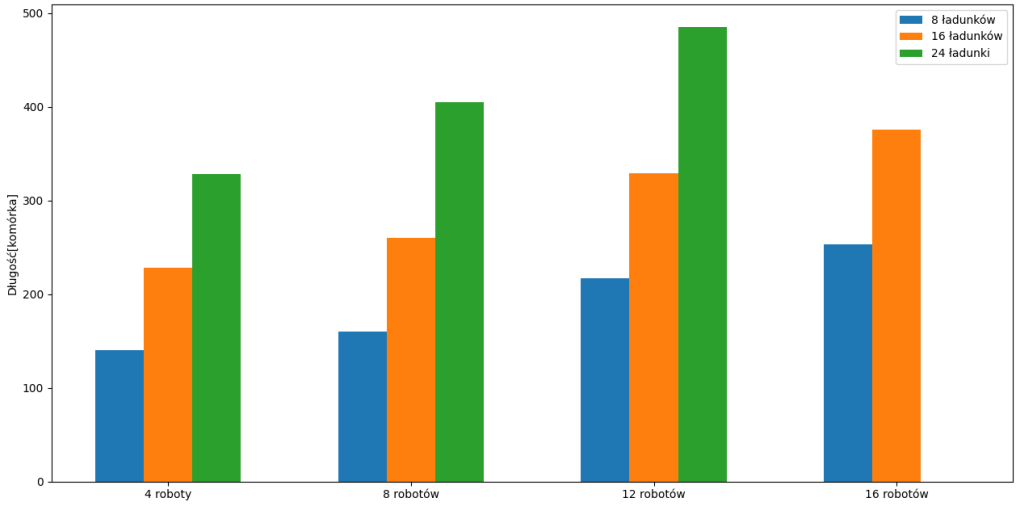
5.2.3 Badania wydajności

W celu sprawdzenia wydajności opracowanej metody zostało wykonanych kilka testów dla różnej liczby robotów oraz dla różnej liczby ładunków. Testy zostały wykonane dla roju robotów składających się z 4, 8, 12 i 16 robotów. Każdy rój robotów miał za zadanie przetransportować w kolejnych testach 8, 16 i 24 ładunki. Badaniu podlegały:

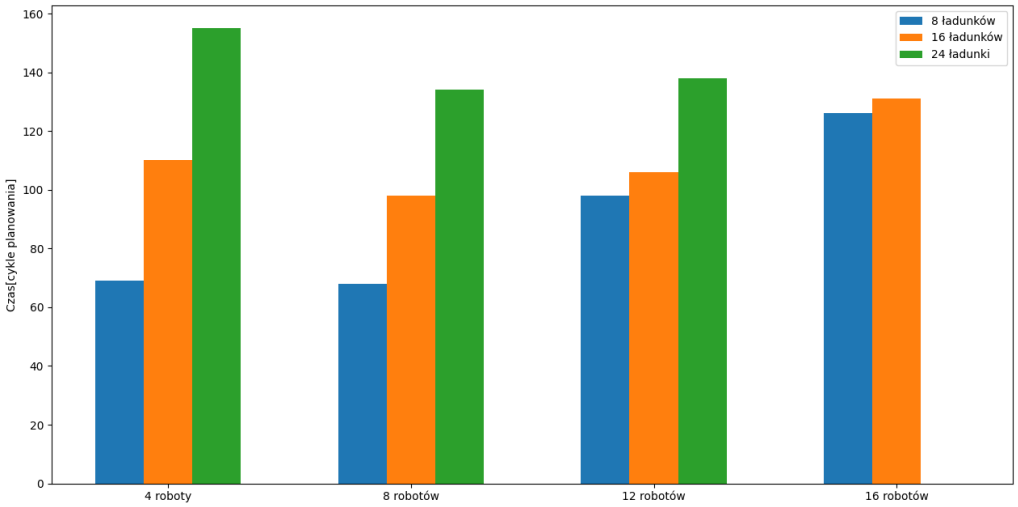
- łączna droga przebyta przez roboty liczona w komórkach,
- czas potrzebny do realizacji zadania w cyklach planowania,
- najkrótszy, najdłuższy oraz średni czas potrzebny na przetransportowanie pojedynczego ładunku ze strefy centralnej do strefy docelowej, wyrażony w cyklach planowania,
- procentowe obciążenie robotów w kolejnych cyklach planowania.

Droga przebyta przez roboty został wyrażona w komórkach. Przebycie jednej komórki oznacza przejazd przez jednego robota z obecnie zajmowanego punktu węzłowego do jednego z sąsiednich punktów węzłowych. Jeden cykl planowania dla metody planowania ścieżek z wykorzystaniem zmodyfikowanego algorytmu A* określa czas między kolejnymi planowaniami ścieżki dla dowolnego robota. Oznacza to, że wartość czasu wyznaczona w cyklach planowania jest inkrementowana w momencie, w którym którykolwiek z robotów zakończy planowanie ścieżki, niezależnie od statusu pozostałych robotów. Z uwagi na wcześniej opisywany problem zakleszczeń, nie udało się pomyślnie zakończyć eksperymentów symulacyjnych dla najbardziej wymagającego scenariusza składającego się z 16 robotów i 24 ładunków.

Wyniki badania łącznej przebytej drogi dla wszystkich prób zostały przedstawione na rysunku 5.3a. Można zauważyć, że wraz ze wzrostem liczby ładunków i liczbą robotów rośnie łączna przebyta droga. Taki wzrost wydaje się zgodny z intuicją, wzrost łącznej przebytej drogi wraz ze wzrostem liczby robotów wynika między innymi z konieczności powrotu na pozycje początkowe po zakończeniu zadania. Wykres czasu potrzebnego do realizacji zadania, w zależności od liczby robotów oraz ładunków został przedstawiony na rysunku 5.3b. Analizując wykresy można spostrzec, że najkrótszy czas realizacji zadania, nie zależnie od liczby ładunków, osiągną rój składający się z 8 robotów. Drugi najlepszy czas dla 8 ładunków osiągnął rój składający się z 4 robotów. W przypadku scenariusza dla 16 i 24 ładunków drugi najlepszy czas uzyskał rój składający się z 12 robotów. Rój składający się z 16 robotów osiągnął najgorsze wyniki, a w przypadku dla 24 paczek dochodziło do zakleszczeń uniemożliwiających zakończenie testu. Przyczyną takich wyników są najprawdopodobniej ruchy robotów wokół strefy centralnej w celu znalezienia nowego ładunku. Takie ruchy powodują dodatkowe utrudnienia dla robotów przenoszących ładunek. Lepszy obraz sytuacji można uzyskać analizując dodatkowo wykresy procentowego obciążenia robotów w kolejnych cyklach czasu przedstawionych na rysunku 5.4. Można zauważyć, że dla roju składającego się z 4 robotów, niezależnie od liczby ładunków, w dużej części czasu wszystkie lub prawie wszystkie roboty przewożą ładunki. Oznacza to, że nie pojawia się wiele ruchów robotów bez ładunku, które przeszkadzałyby w przewożeniu

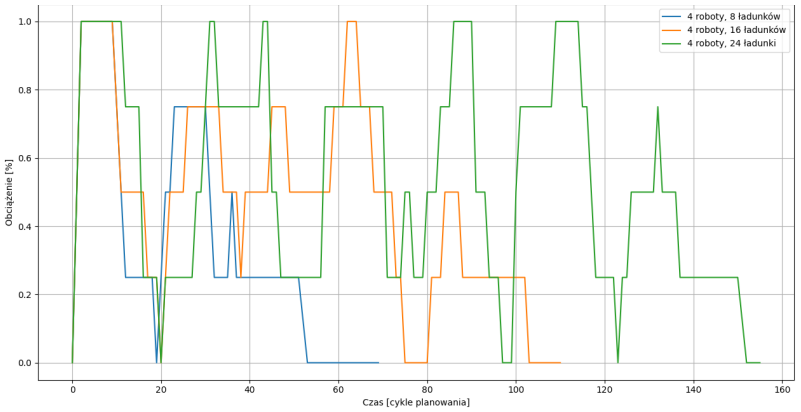


(a)

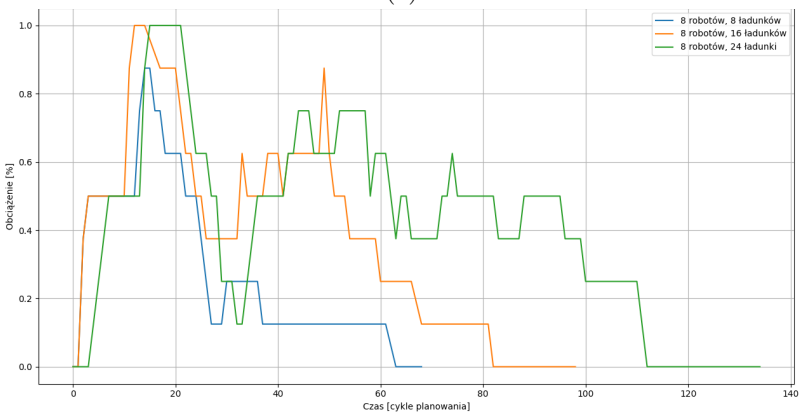


(b)

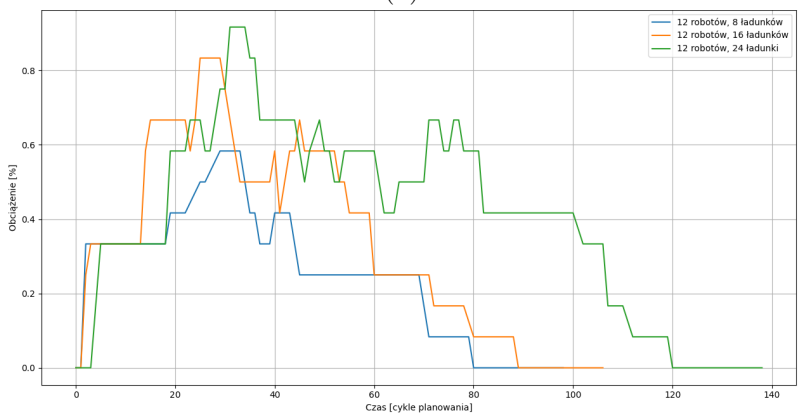
Rysunek 5.3 a) Łączna przebyta droga, b) czas potrzebny do realizacji zadania



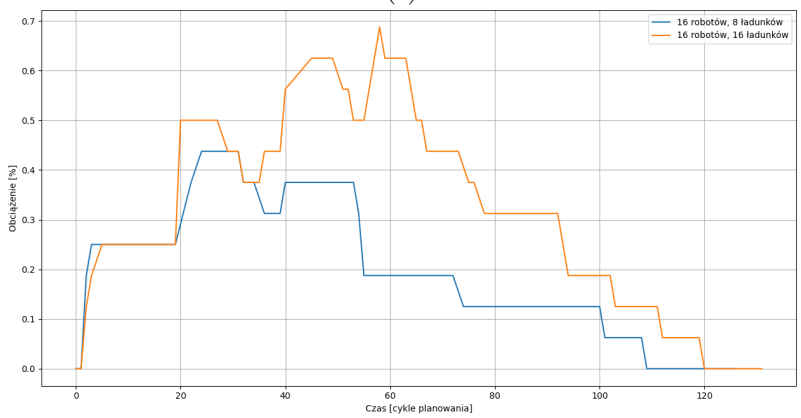
(a)



(b)



(c)



(d)

Rysunek 5.4 Obciążenie robotów dla roju składającego się z a) 4 robotów, b) 8 robotów, c) 12 robotów, d) 16 robotów

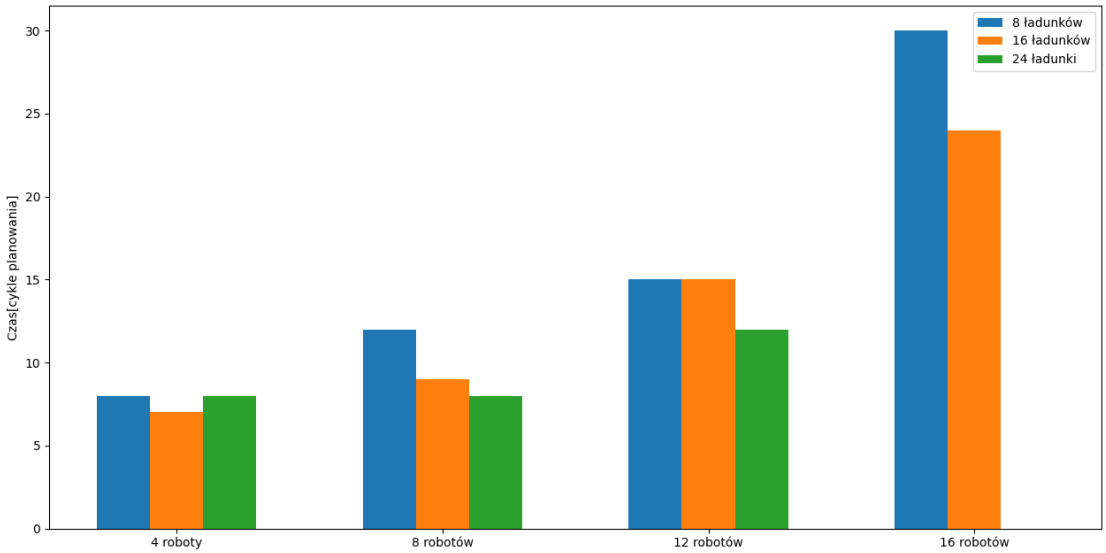
ładunków przez inne roboty. Natomiast system jest znacząco obciążony i dodanie kolejnych robotów do roju mogłoby polepszyć jego działanie. W przypadku 8 robotów w roju, na początku następuje gwałtowny wzrost obciążenia, wszystkie lub prawie wszystkie roboty przewożą ładunek. Następnie widać schodkowy spadek obciążenia, pierwsze 4 roboty odłożyły ładunek, następnie pozostałe roboty pozbyły się ładunku. W kolejnych cyklach widać kolejny schodkowy wzrost obciążenia. Oznacza to, że roboty wymijały się w środkowych częściach drogi między poszczególnymi wjazdami do stref. Nie powstawały przez to przestoje przy wjazdach do stref i ruch robotów z ładunkami mógł odbywać się płynnie. Zwiększenie liczby robotów w roju do 12 zaburzyło równowagę. Na wykresie (rys. 5.4c) można spostrzec, że początkowy wzrost obciążenia, przy 12 robotach w roju, nie prowadzi do maksymalnego obciążenia robotów. Dodatkowo ustabilizowanie się poziomu obciążenia w okolicach 60% oznacza, że średnio 7 robotów przewozi ładunek natomiast pozostałe 5 robotów wraca bądź oczekuje na możliwość pobrania ładunku. Ponieważ są tylko 4 części strefy centralnej, powstają pewne blokady wokół wjazdów spowodowane robotami oczekującymi na możliwość wjazdu i pobrania ładunku. Wydłuża to wzrost czasu potrzebnego do realizacji zadania. Podobną tendencje jak w przypadku dla 12 robotów widać na wykresie obciążenia dla 16 robotów. Przy czym dla 16 robotów można zauważyć pojedynczy wzrost obciążenia oraz późniejszy spadek. Oznacza to, że roboty zaczęły gromadzić się wokół wjazdów do strefy centralnej. Dodatkowo roboty, które jako pierwsze podjęły ładunek, zdążyły wrócić do wjazdów strefy centralnej zanim pozostałe roboty wjechały do strefy.

Innym aspektem zwiększania liczebności roju jest wzrost liczby przecięć ścieżek robotów oraz konieczność prowadzenia ścieżek w sposób pozwalający na ominięcie pozostałych robotów. Na rysunku 5.5 zostały przedstawione wykresy najkrótszego, najdłuższego oraz średniego czasu potrzebnego na dostarczenie pojedynczego ładunku. Analiza wykresów pozwala zauważyć, że wraz ze wzrostem liczby wykorzystanych robotów rośnie najkrótszy, średni oraz najdłuższy czas przewozu ładunku. Wynika to najprawdopodobniej z faktu, że wzrost liczby robotów powoduje konieczność prowadzenia dłuższych ścieżek pozwalających na ominiecie pozostałych robotów oraz powstawania przecięć ścieżek prowadzących do oczekiwania na możliwość przejazdu.

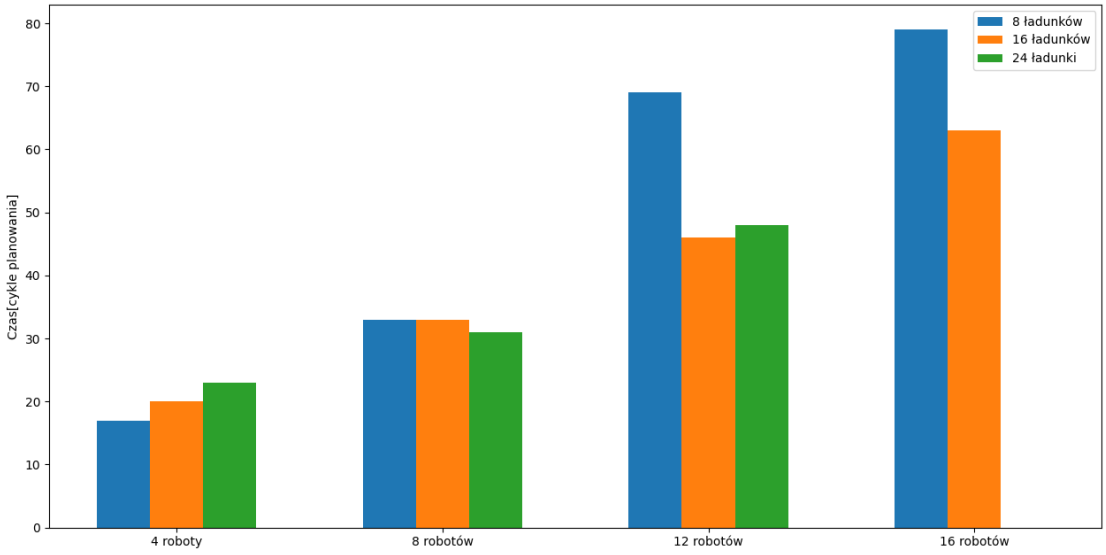
Wszystkie poruszone aspekty prowadzą do wniosku, że zastosowanie coraz to większej liczby robotów nie musi przyczyniać się do polepszania czasu potrzebnego do wykonania zadania. Wykres łącznej przejechanej przez roboty drogi wskazuje, że wraz ze wzrostem liczby robotów rośnie przejechana droga, a co za tym idzie koszty związane z pracą roju.

5.2.4 Wnioski

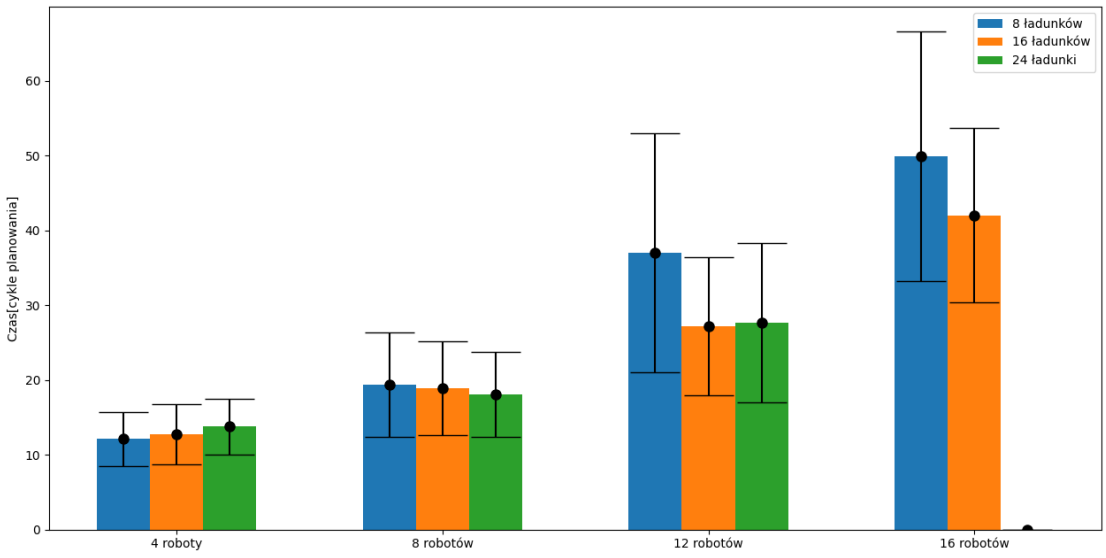
Metoda planowania ścieżek z wykorzystaniem zmodyfikowanego algorytmu A* pozwala na rozwiązanie postawionego zadania. Z uwagi na heurystyczny charakter algorytmu A* metoda może prowadzić do optymalnego rozwiązania, jednak nie jest to zagwarantowane. Skuteczność metody zależy od kilku czynników. Pierwszy z nich jest odpowiedni dobór parametrów związanych z metodą, między innymi wartości kosztów przejścia między kolejnymi węzłami oraz sposób ich modyfikacji, jak również dobór sposobu obliczania wartości heurystycznej w funkcji oceny węzła. Kolejnym czynnikiem jest sposób rozwiązywania problemu zakleszczeń. Implementacja odpowiednich mechanizmów zapobiegających oraz eliminujących zakleszczenia wpływa na jakość oraz niezawodność metody. Wzrost liczby robotów może prowadzić do częstszego występowania problemu zakleszczeń i wymagać coraz to bardziej skomplikowanych mechanizmów ich eliminacji. Jednocześnie odpowiedni dobór liczebności roju jest kolejnym czynnikiem wpływającym na efektywność metody.



(a)



(b)



(c)

Rysunek 5.5 a) Najkrótszy czas dostarczenia pojedynczego ładunku, b) najdłuższy czas dostarczenia pojedynczego ładunku, c) średni czas dostarczenia pojedynczego ładunku

Mała liczba robotów wykorzystanych w roju, pozwala na zmniejszenie kosztów związanych z przebytą przez nie drogą, jednocześnie powodując przeciążenie systemu i gorsze wyniki związane z czasem realizacji zadania. Zbyt duża liczba robotów prowadzi z jednej stron do dużych kosztów związanych z ruchem robotów, z drugiej strony zwiększa prawdopodobieństwo zakleszczenia oraz oczekiwania poszczególnych robotów na wykonanie ruchu przez inne roboty. Prowadzi to do wzrostu czasu potrzebnego na wykonanie zadania. Tak więc istotnym czynnikiem jest odpowiedni dobór liczby robotów. Przeprowadzone testy wskazały, że dla metody wykorzystującej zmodyfikowany algorytm A*, najlepsze wyniki zostały uzyskane dla 8 robotów, czyli podwojoną liczbę części strefy centralnej.

5.3 Planowanie ścieżek oparte o algorytm optymalizacji numerycznej

Druga opracowana metoda planowania ścieżek robotów zakłada wykorzystanie optymalizacji numerycznej. Opis algorytmu optymalizacji numerycznej w ogólnej formie został przedstawiony w rozdziale 2.2. Wykorzystanie algorytmu do planowania ścieżek, oprócz zdefiniowania wartości odpowiednich parametrów oraz postaci kryterium jakości, wymagało opracowania dodatkowej metody interpretacji otrzymanej predykcji w celu wyznaczenia kolejnych ruchów robotów. Jako zmienne stanu zostały przyjęte położenia wszystkich robotów oraz wartość przejechanej drogi przez roboty między celami pośrednimi w chwili czasu n, $x(n) = [x_1, y_1, d_1, x_2, y_2, d_2, \ldots, x_m, y_m, d_m]^T$. Wartości przejechanej drogi były zerowane za każdym razem gdy dany robot osiągnął cel pośredni, na przykład przejechał z jednej strefy do drugiej. Jako sygnały sterujące zostały przyjęte prędkości liniowe oraz orientacje robotów w chwili czasu n, $u(n) = [v_1, \theta_1, v_2, \theta_2, \ldots, v_m, \theta_m]^T$. Został również zdefiniowany wektor współrzędnych pozycji końcowych wszystkich robotów $x_f = [x_f^1, y_f^1, x_f^2, y_f^2, \ldots, x_f^m, y_f^m]^T$. Wykorzystany do predykcji ruchu robotów model został wyrażony wzorem

$$\begin{cases} \dot{x} = v\cos(\theta) \\ \dot{y} = v\sin(\theta) \end{cases}, \tag{5.1}$$

gdzie v to prędkość liniowa robota, a θ to kąt pod jakim skierowany jest robot. Postać wykorzystanego modelu wynika z założenia, że w zadaniu zostały użyte roboty należące do klasy (2,0) i mające zdolność obrotu w miejscu. Stały horyzont czasowy podczas przeprowadzonych eksperymentów został ustawiony na $N = 12$ kroków czasu. Wartość horyzontu czasu została wyznaczona na podstawie testów, tak aby zapewniała satysfakcjonującą jakość predykcji przy jednoczesnym niezbyt długim czasie obliczeń.

Kryterium jakości składa się z sumy odległość poszczególnych pozycji robotów od punktów docelowych, długości przejechanej drogi przez roboty oraz sumy funkcji nasycenia poszczególnych odległości między robotami. Pierwszy ze składników sumy odpowiada za wytyczanie najkrótszej ścieżki, drugi pozwala uniknąć nadmiarowych ruchów, szczególnie powtarzania przez roboty kilka razy ruchu w jedną i w drugą stronę. Trzeci składnik ma na celu zapewnić utrzymanie odpowiedniego dystansu między robotami, aby nie dochodziło do kolizji. Postać kryterium jakości wykorzystanego w zadaniu planowania ścieżek robotów została wyrażona wzorem

$$J_N\left(x(n), u(\cdot)\right) = m(x_u(N, x(n))) + \sum_{k=0}^{N-1} l(x_u(k, x(n)), u(k), x_f), \tag{5.2}$$

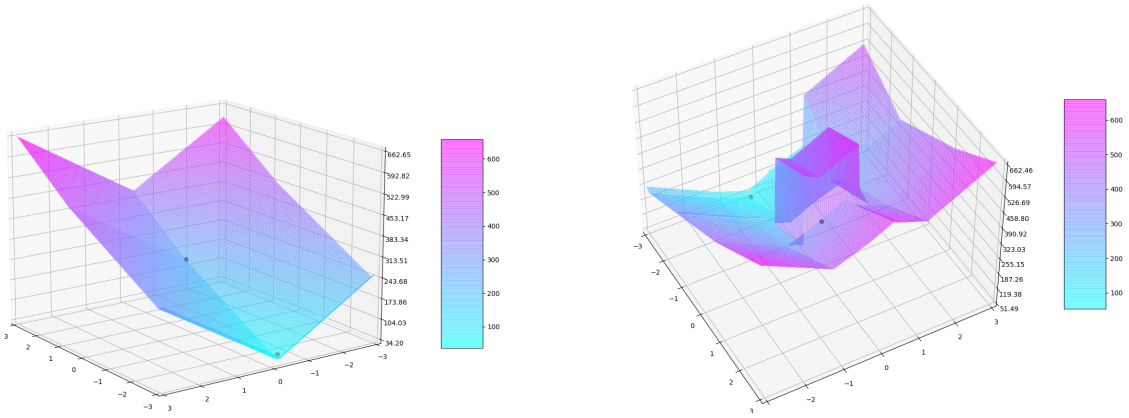
gdzie $m(x_u(N, x(n))) = \alpha \cdot f_o(x_u(N, x(n)), x_f)$ i $f_o(\cdot)$ to funkcja odległości, natomiast

$$l(x_u(k, x(n)), u(k), x_f) = \beta \cdot f_o(x_u(k, x(n)), x_f) + \gamma \cdot \sum_{i=1}^{m} \sum_{j=1}^{m} (\varphi(|x_i - x_j|) + \varphi(|y_i - y_j|)) + \delta \cdot \sum_{i=1}^{m} d_i \tag{5.3}$$

i $\varphi(x) = 1 - \frac{2x}{\sqrt{1+4x^2}}$ jest funkcją nasycenia. Wartości parametrów α, β, γ i δ zostały dobrane na podstawie testów tak, aby zapewnić pożądane zachowanie robotów. Ograniczenia przestrzeni stanu wynikały z konieczność przemieszczania się przez roboty między punktami węzłowymi wzdłuż wyznaczonych linii oraz wyznaczonego obszaru przestrzeni roboczej.

Początkowo jako funkcja odległości była wykorzystywana norma taksówkowa. Jednak wykorzystanie normy taksówkowej prowadziło do powstania minimów lokalnych dla punktów wokół strefy centralnej w momencie kiedy roboty miały za zadanie przejechać na jej drugą stronię. Prowadziło to do blokowania się robotów oraz braku możliwości zrealizowania zadania. Częściowa funkcja kosztu wykorzystująca normę taksówkową dla jednego z minimów lokalnych została przedstawiona na rysunku 5.6a. W celu wyeliminowania problemu z występowaniem minimów lokalnych została zmodyfikowana funkcja odległości. Główna idea liczenia odległości polegała na podziale strefy roboczej wokół strefy centralnej na cztery sektory. Przejazd robota z jednego sektora do drugiego mógł odbyć się tylko przez dwa wyznaczone punkty węzłowe. Jeśli punkt początkowy i docelowy znajdował się w tym samym sektorze, odległość liczona była bezpośrednio za pomocą normy taksówkowej. Jeżeli punkt początkowy znajdował się w jednym sektorze a punkt docelowy w części sąsiadującej, to odległość obliczana była jako suma odległości od punktu początkowego do najbliższego punktu węzłowego wyznaczonego na granicy dwóch części oraz z punktu na granicy do punktu docelowego. Analogicznie obliczana była odległość dla przypadku kiedy punkt początkowy znajdował się w sektorze leżącym na przeciwko sektora z punktem docelowym (sektory nie sąsiadują ze sobą). Suma dodatkowo składała się z odległości między punktami na granicach sektorów sąsiadujących z sektorem, w którym znajdował się punkt początkowy oraz sektorem z punktem docelowym. Taki sposób liczenia odległości wymagał odpowiedniego mechanizmu wyboru poszczególnych węzłów na granicach sektorów, które gwarantowałyby najkrótszą drogę. Częściowa funkcja kosztu wykorzystująca zmodyfikowaną metodę miary odległości została przedstawiona na rysunku 5.6b. Jak można zauważyć jest to identyczna sytuacja jak na rysunku 5.6a. Zmodyfikowana funkcja odległości nie powoduje powstania minimum lokalnego.

Z uwagi na fakt, że do planowania ścieżek robotów została wykorzystana optymalizacja z ciągłą przestrzenią stanu, natomiast możliwe do osiągnięcia przez roboty położenia poza strefami są określone za pomocą zdyskretyzowanych punktów węzłowych, należało opracować sposób interpretacji otrzymanych predykcji. Interpretacja polegała na dopasowaniu punktów uzyskanych w procesie optymalizacji do punktu węzłowego w taki sposób, aby określić następny ruch robota (lub jego brak). Metoda wyznaczania kolejnego ruchu robota jest realizowana w kilku krokach. Na początku wybierana jest współrzędna, dla której wartość bezwzględna różnicy badanej współrzędnej obecnie sprawdzanego punktu zwróconego w predykcji oraz położenia robota jest największa. W kolejnym kroku sprawdzane jest, względem wybranej współrzędnej, po której stronie obecnie zajmowanego przez robota punktu węzłowego znajduje się badany punkt z predykcji. Na przykład, jeżeli została wybrana współrzędna x oraz współrzędna x punktu z predykcji miała mniejszą wartość od współrzędnej x położenia robota, to punkt z predykcji znajduje się po lewej stronie. Następnie wybierany jest taki węzeł sąsiadujący z punktem węzłowym obecnie zajmowanym przez robot, który znajduje się po tej samej stronie co punkt z predykcji. Jeżeli po



(a) (b)

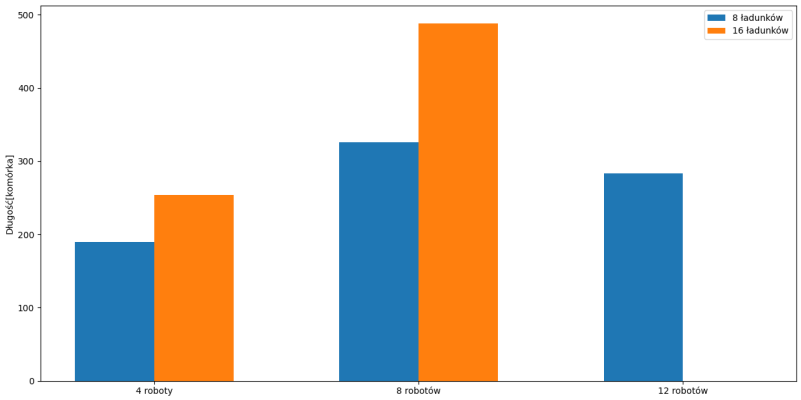
Rysunek 5.6 Częściowa funkcja kosztu wykorzystująca a) normę taksówkową , b) zmodyfikowaną metodę miary odległości

wyznaczonej stronie nie ma sąsiadującego węzła to metoda przechodziła do sprawdzania kolejnego punktu zwróconego w predykcji. Za pomocą normy taksówkowej obliczana jest odległość między sąsiadującymi ze sobą punktami węzłowymi. Połowę tej odległości stanowi próg. Jeżeli różnica dla wybranej współrzędnej między aktualnie badanym punktem z predykcji oraz położeniem robota jest większa od progu to kolejny ruch odbywa się do wybranego, sąsiedniego węzła. Jeżeli próg nie został przekroczony to procedura jest powtarzana dla kolejnych punktów z predykcji, aż wartość różnicy przekroczy próg lub zostały przebadane wszystkie punkty zwróconego w predykcji.

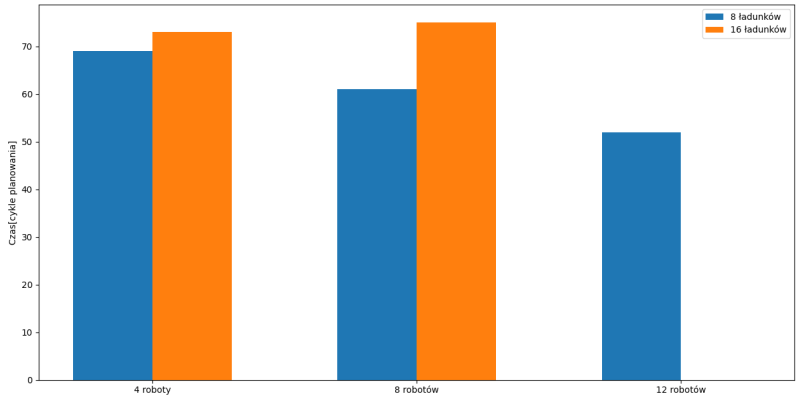
5.3.1 Badania wydajności

Badanie wydajności metody planowania ścieżek robotów wykorzystującej optymalizację numeryczną zostało wykonane z wykorzystaniem identycznych statystyk jak przy badaniu metody opartej o zmodyfikowany algorytm A*. Ze względu na dużą złożoność obliczeniową badania zostały przeprowadzone dla 4, 8 i 12 robotów oraz 8 i 16 ładunków. W trakcie badań pojawiły się problemy związane ze zbyt długiego czasu obliczeń oraz blokowaniem się robotów poprzez cykliczne powtarzanie identycznych ruchów. Dodatkowo zostały zmienione położenia początkowe (i tym samym końcowe) robotów w porównaniu do testów przeprowadzonych dla metody opartej o zmodyfikowany algorytm A*. Roboty zostały rozmieszczone na brzegach obszaru roboczego. Zmiana początkowego i zarazem końcowego umiejscowienia robotów była spowodowana faktem częstego występowania blokad między robotami, które zakończyły swoje działanie i dotarły na pozycje końcową (położoną przy wjeździe do strefy centralnej) a robotami aktywnie realizującymi zadania. Blokady uniemożliwiały poprawne zakończenie symulacji.

Wykres przedstawiający łączną drogę przejechaną przez roboty w poszczególnych testach został umieszczony na rysunku 5.7a. Można zauważyć, że dla 4 i 8 robotów, wzrost liczby robotów oraz ładunków powodował zwiększenie łącznej przejechanej drogi. Taki wzrost jest zgodny z intuicją, większa liczba ładunków wymusza większą liczę przejazdów między strefami, natomiast większa liczba robotów prowadzi do konieczność częstszego wymijania się robotów. Inaczej jednak przedstawiają się wyniki dla przypadku 12 robotów i 8 ładunków. Łączna przebyta droga jest mniejsza od łącznej przebytej drogi dla 8 robotów i 8 ładunków. W celu wyjaśnienia takiego wyniku, należy przeanalizować wykresy przedstawiające obciążenie robotów zamieszczony na rysunku 5.8. Porównując ze sobą



(a)

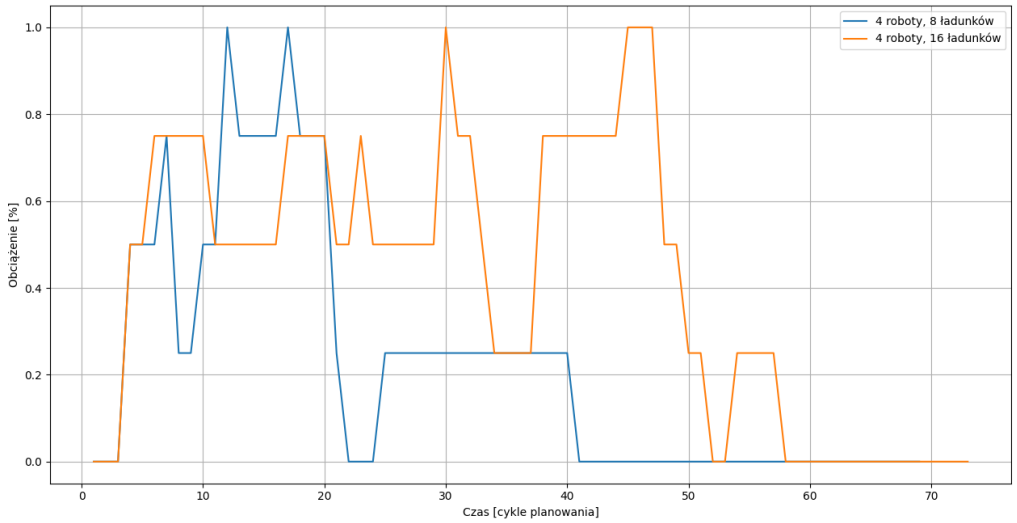


(b)

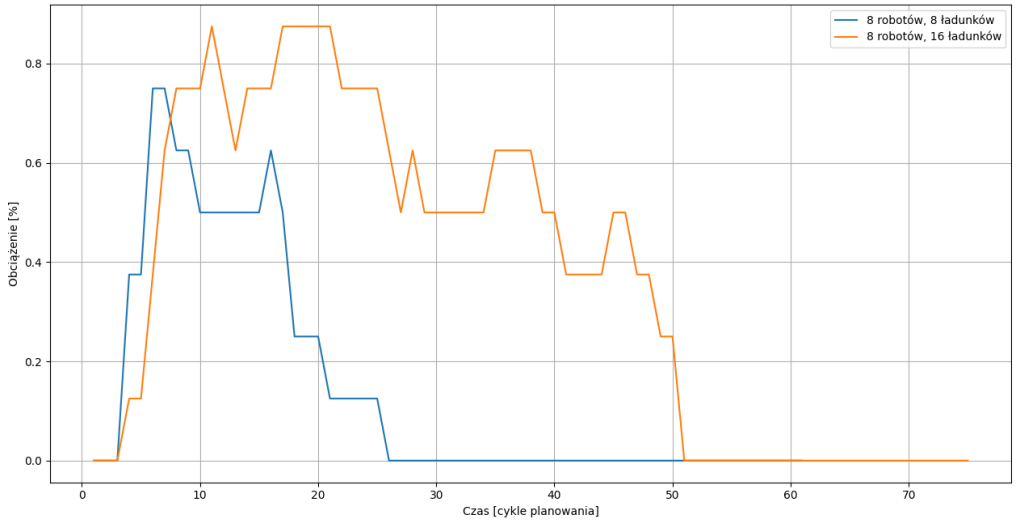
Rysunek 5.7 a) Łączna przebyta droga, b) czas potrzebny do realizacji zadania

przebiegi dla 8 i dla 12 robotów w przypadku przewozu 8 ładunków, można zauważyć, że dla 8 robotów sam czas przewozu ładunków jest znacznie krótszy niż dla przypadku 12 robotów. Po przetransportowaniu ładunków, roboty muszą wrócić na pozycje początkowe. Przyglądając się wspomnianym przebiegom można spostrzec, że dla 8 robotów ta faza zadania trwa dłużej niż faza przewozu ładunków. Oznacza to, że roboty po zakończeniu fazy transportu ładunków musiały znajdować się w bardzo niekorzystnych pozycjach względem pozycji końcowych, na przykład po przeciwległej stronie strefy centralnej. W skutek tego znacząco wydłużył się czas realizacji zadania i jednocześnie łączna przejechana droga.

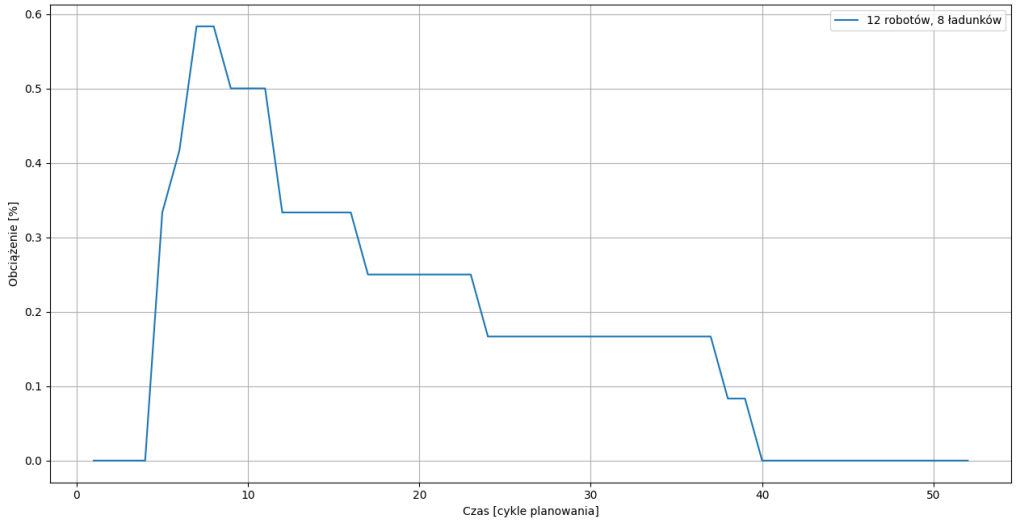
Wykres zawierający czas wykonania zdania dla poszczególnych testów został przedstawia na rysunku 5.7b. Intuicyjnie możnaby zakłożyć, że wraz z zwiększeniem liczby robotów skróci się czas potrzebny do wykonania zadania. Jednak podczas przewozu 16 ładunków, czas wykonania zadania dla 8 robotów jest większy niż dla 4 robotów. Dodatkowo, na podstawie rezultatów uzyskanych dla metody opartej o zmodyfikowany algorytm A*, możnaby przypuszczać, że czas wykonania zadania przy użyciu 8 robotów będzie najkrótszy. Ponownie, w celu uzyskania lepszego obrazu sytuacji należy przyjrzeć się przebiegom obciążeń robotów przedstawionych na rysunku 5.8. W przypadku użycia 4 robotów, można zobaczyć, że obciążenie wszystkich robotów dochodzi w niektórych momentach czasu do 100%. Można wyciągnąć z tego wniosek, że system jest zbyt obciążony i warto użyć większej liczby robotów. W przypadku 8 robotów, obciążenie robotów nie osiąga 100%. Widoczny jest również początkowy gwałtowny wzrost obciążenia i późniejszy stopniowy jego spadek. Oznacza to, że wokół wjazdów strefy centralnej nie powstają skupiska robotów oczekujących na możliwość wjazdu do strefy i przez to nie występują dodatkowe utrudnienia w ruchu robotów związane z oczekującymi robotami. Jak już wcześniej



(a)



(b)



(c)

Rysunek 5.8 Obciążenie robotów dla roju składającego się z a) 4 robotów, b) 8 robotów, c) 12 robotów

zostało zauważone, faza przewozu ładunków jest najkrótsza dla testów z wykorzystaniem 8 robotów, natomiast faza powrotu na pozycje końcowe znacząco wydłuża czas realizacji całego zadania. Testy wykazały, że opracowana metoda ma problemy w przypadku kiedy kilka robotów (3-4) w tej samej chwili czasu próbuje się minąć jadąc w przeciwnych kierunkach. Wynika to zapewne z przyjętego sposobu interpretacji predykcji.

Statystyki dotyczące najkrótszego, najdłuższego oraz średniego czasu przewozu pojedynczego ładunku zostały pokazane na rysunku 5.9. Analizując wykres przedstawiający średni czas przewozu ładunku można zauważyć, że wraz ze wzrostem liczby robotów, wydłuża się czas potrzebny na przetransportowanie ładunku. Wynika to z faktu, że użycie w zadaniu większej liczby robotów powoduje częstsze powstawanie sytuacji kiedy roboty muszą się wymijać lub oczekiwać na możliwość przejazdu. Powoduje to wydłużenie czasu przewozu ładunków.

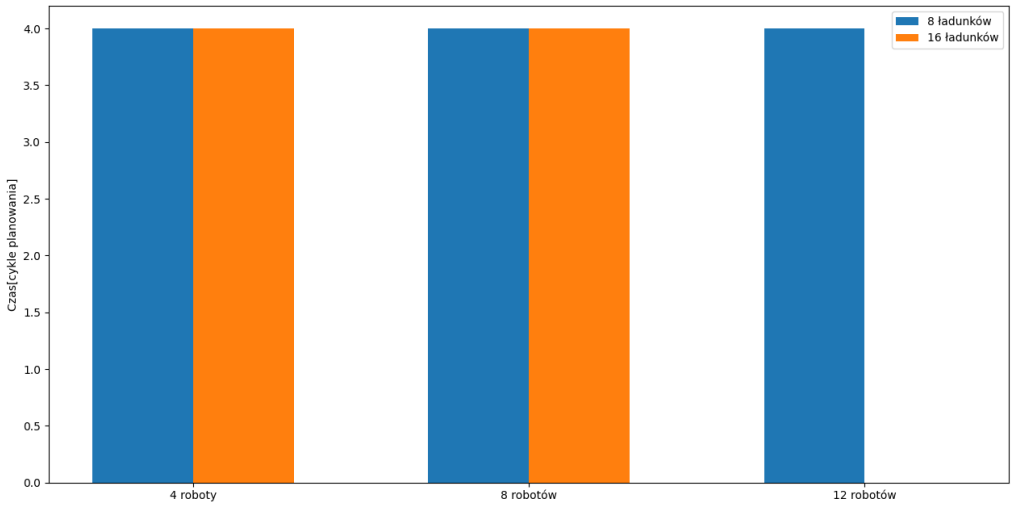
5.3.2 Wnioski

Przeprowadzone badania wskazują, że metoda planowania ścieżek z wykorzystaniem optymalizacji numerycznej pozwala na wykonanie poprawnie przyjętego zadania. Z wykorzystaniem metody wiążą się jednak pewne problemy, których sposób rozwiązania wpływa na wydajność i możliwości samej metody. Pierwszą istotną trudność stanowi złożoność obliczeniowa związana z dużą liczba zmiennych wynikającą z samego problemu, jak również z rozpatrywania predykcji ruchu w kolejnych momentach czasu. Duża złożoność obliczeniowa powodowała długi czas obliczeń pojedynczego cyklu planowania, przez co został zastosowany mechanizm przerywania obliczeń, jeżeli nie zostało znalezione optymalne rozwiązanie (spełniające przyjęte założenie dokładności) w odpowiedniej liczbie iteracji. Takie podejście zredukowało czas obliczeń, jednak prowadziło do uzyskania częściowych rozwiązań problemu w kolejnych cyklach planowania. Długi czas obliczeń i duża złożoność obliczeniowa były głównymi przyczynami ograniczenia testów do przypadków przewozu 8 i 16 ładunków.

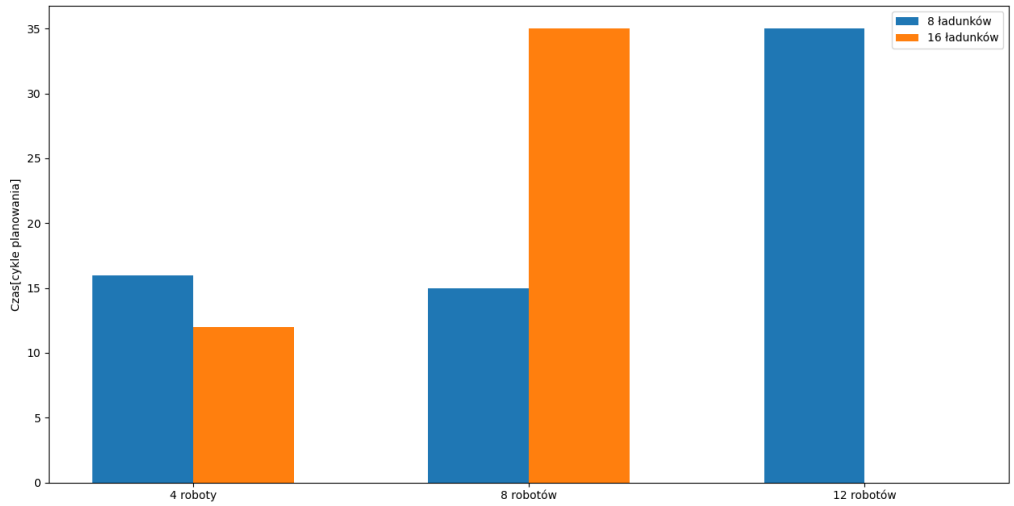
Kolejna trudność związana jest z doborem funkcji kosztu o odpowiednim kształcie. Jest to problem związany z metodami optymalizacji numerycznej i ich wrażliwości na lokalne minima funkcji kosztu. Od kształtu funkcji kosztu znacząco zależy czy zostanie znalezione odpowiednie rozwiązania oraz wpływa on na jakość rozwiązania. Dodatkowo, funkcja kosztu najczęściej musi zostać indywidualnie dobrana do danego problemu. W przypadku przyjętego zadania pojawiała się potrzeba zmiany sposobu liczenia odległości z powodu powstawania minimów lokalnych wokół strefy centralnej oraz blokowania się w nich robotów. Z funkcją kosztu związane są również różne parametry, których wartości często można odpowiednio dobrać jedynie na podstawie testów.

Z uwagi na zastosowanie metody optymalizacji numerycznej wykorzystującej ciągłą przestrzeń stanu do zadania planowania ścieżek, które reprezentowane są za pomocą dyskretnej przestrzeni stanu (punktów węzłowych) pojawiła się trudność ze sposobem interpretacji predykcji. W wykorzystanej metodzie, ruch robota odbywał się w momencie kiedy jeden z punktów uzyskanych jako predykcja ruchu robota przekroczył odpowiedni próg. Takie podejście daje pewność, że nawet dla częściowych predykcji (gdy punkty predykcji nie łączą obecnej pozycji robota z punktem docelowym) zostanie wykonany ruch. Przyjęty sposób wymusza jednocześnie w pewnych sytuacjach ruch robota, który według predykcji powinien odbyć się w następnym kroku czasu. Może to prowadzi, na przykład do konieczności wykonania ruchu powrotnego w celu umożliwiania przejazdu innemu robotowi.

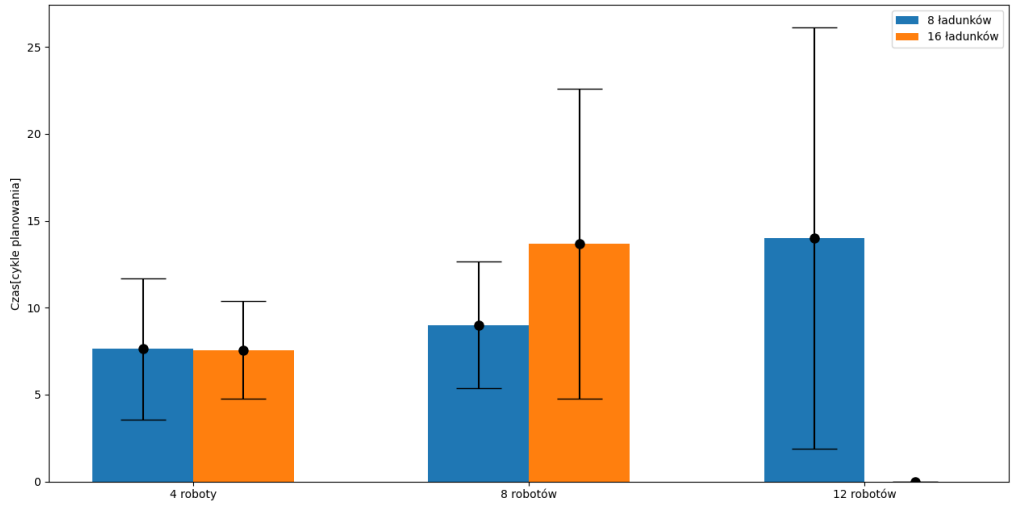
Metoda planowania ścieżki z wykorzystaniem optymalizacji numerycznej nie wymaga-



(a)



(b)



(c)

Rysunek 5.9 a) Najkrótszy czas dostarczenia pojedynczego ładunku, b) najdłuższy czas dostarczenia pojedynczego ładunku, c) średni czas dostarczenia pojedynczego ładunku

ła implementacji specjalnych mechanizmów pozwalających na rozwiązywanie zakleszczeń. Wyliczane predykcje ruchu pozwalały odpowiednio wymijać się robotom, nawet w sytuacji gdy kilka z nich znajdowało się w otoczeniu wjazdu do strefy centralnej. Problematyczne okazała się jedynie sytuacje kiedy jeden z robotów zakończył już swoje działanie, jednak jego pozycja końcowa znajdowała się na ścieżce innego działającego robota. Predykcja wymuszająca odsunięcie się robota, który zakończył działanie nie mogła zostać wykonana i przez to dochodziło do blokady. Próbą rozwiązania takiego problemu może być uznanie robotów, które zakończyły działanie za statyczne przeszkody i uwzględnienie tego faktu podczas rozwiązywania zadania optymalizacji. Dzięki takiemu podejściu możliwe byłoby również zredukowanie liczby optymalizowanych zmiennych, a tym samym, najprawdopodobniej, skrócenie czasu obliczeń.

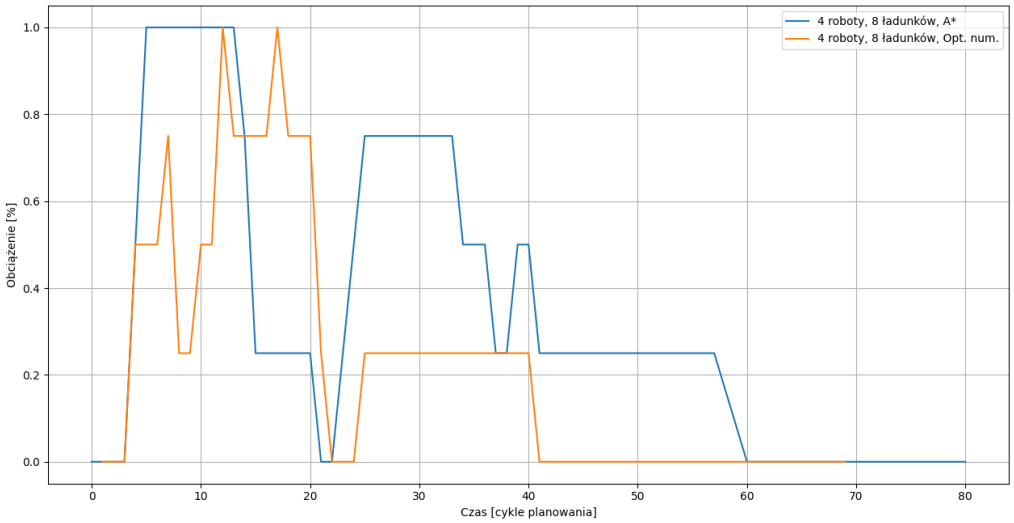
5.4 Porównanie algorytmów

W celu porównania wydajności obydwu algorytmów należało wykonać dodatkowe testy dla metody opartej o zmodyfikowany algorytm A*, w których pozycje startowe i jednocześnie pozycje końcowe robotów były takie same jak podczas testów dla metody wykorzystującej optymalizację numeryczną. Jednakowe badania, dla przedstawionych metod, zostały przeprowadzone dla 4, 8, 12 robotów, które miały przetransportować 8 oraz 16 ładunków. Do porównania algorytmów zostały wykorzystane takie same statystyki jak przy badaniu wydajności poszczególnych algorytmów. Zestawienia wyników dla poszczególnej liczby robotów zostały przedstawione w tabeli 5.1. Analizując wyniki można zauważyć, że dla wszystkich przypadków metoda wykorzystująca optymalizację numeryczną kończyła zadanie w mniejszej liczbie cykli planowania, przy czym w każdym przypadku prowadziło to do przejechania przez roboty dłuższej, łącznej drogi. Należy tu jednak zauważyć, że krótszy czas realizacji zadania wyrażony w cyklach planowania nie oznacza krótszego, rzeczywistego czasu obliczeń. Metoda wykorzystująca zmodyfikowany algorytm A* jest mniej złożona obliczeniowo i wymaga nawet o rząd krótszego czasu obliczeń w porównaniu do metody wykorzystującej optymalizację numeryczną.

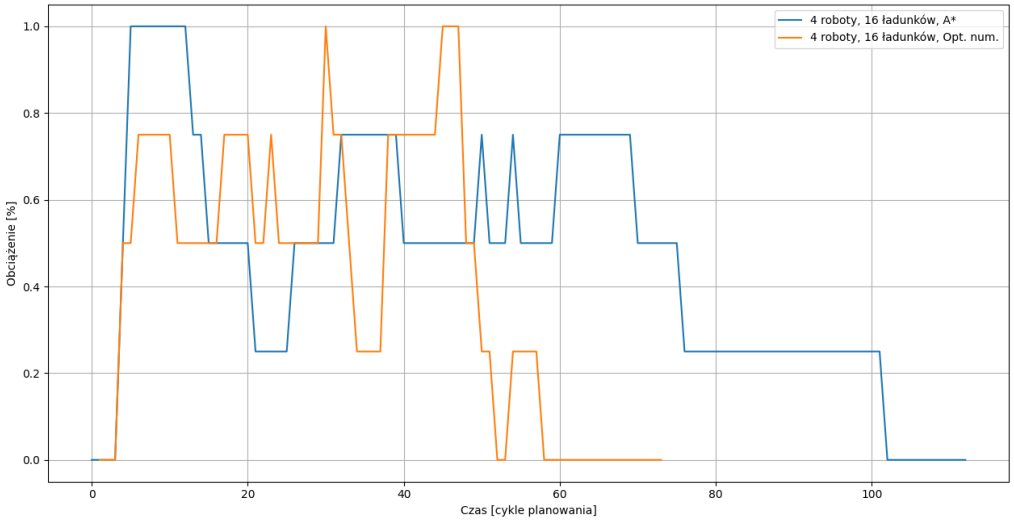
Wykresy obciążenia robotów zostały przedstawione na rysunkach 5.10 i 5.11. Przyglądając się przebiegom obciążenia dla 4 robotów, można spostrzec, że w przypadku metody opartej o zmodyfikowany algorytm A* następuje początkowy, gwałtowny wzrost obciążenia do 100%, wszystkie roboty podnoszą ładunek, później obciążenie mocno spada i ponownie wzrasta. Roboty podnoszą i odkładają ładunki falami. W przypadku metody wykorzystującej optymalizację numeryczną wygląda to inaczej. Początkowy wzrost obciążenia nie dochodzi do poziomu 100%. Roboty stopniowo podnoszą ładunki przez co obciążenie oscyluje na poziomie 50%. Dopiero w późniejszym okresie następują chwilowe skoki obciążenia do 100%. Nieco inaczej przedstawiają się wykresy obciążeń dla 8 i 12 robotów. Dla obydwu metod można zauważyć gwałtowny wzrost obciążenia na początku. W przypadku metody wykorzystującej optymalizację numeryczną, po wzroście następuje stopniowy, sukcesywny spadek obciążenia z małymi, chwilowymi wzrostami. W przypadku metody opartej o zmodyfikowany algorytm A*, przebiegi dla przypadku 8 ładunków, 8 i 12 robotów wyglądają podobnie jak w przypadku drugiej metody. Warto natomiast zwrócić uwagę na przebieg obciążenia w przypadku przewożenia 16 ładunków przez 8 robotów. Można zauważyć, że po dużym wzroście obciążenia, następuje duży spadek i ponowny duży wzrost. Roboty podjeżdżają do strefy centralnej w większej ilości, falami. Tym różni się metoda oparta o zmodyfikowany algorytm A* od metody wykorzystującej optymalizację numeryczną, w której roboty podjeżdżają do wjazdów stopniowo.

Tabela. 5.1 Porównanie wyników badań dla dwóch metod

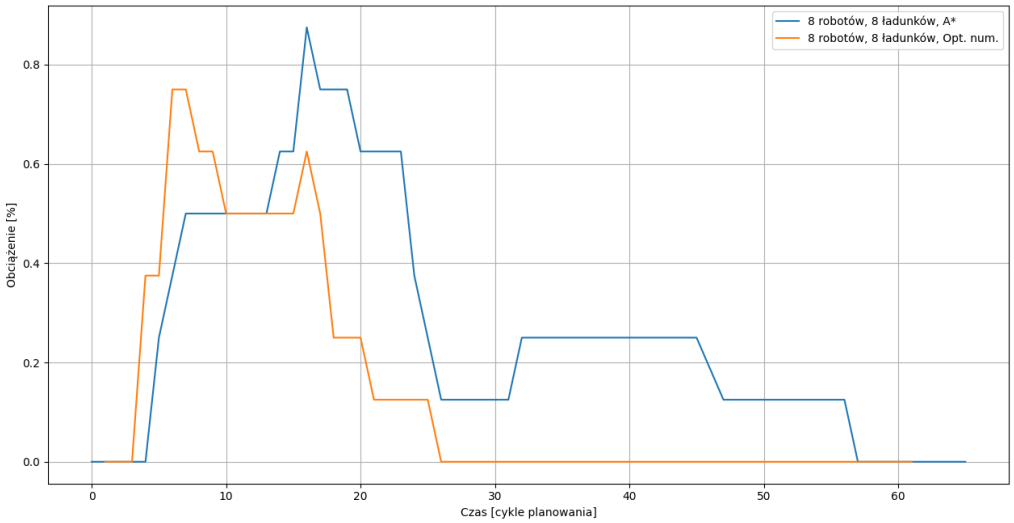
statystyki	4 robotów			
	8 ładunków		16 ładunków	
	A*	Opt. num.	A*	Opt. num.
Czas	80	69	112	73
Łączna droga	175	190	243	254
Najkrótszy czas na ładunek	9	4	8	4
Najdłuższy czas na ładunek	21	16	17	12
Średni czas na ładunek	13.25 ± 4.21	7.63 ± 4.06	12.56 ± 3.45	7.56 ± 2.81
	8 robotów			
	8 ładunków		16 ładunków	
	A*	Opt. num.	A*	Opt. num.
Czas	65	61	108	75
Łączna droga	209	326	341	488
Najkrótszy czas na ładunek	10	4	10	4
Najdłuższy czas na ładunek	31	15	27	35
Średni czas na ładunek	17.25 ± 6.98	9.00 ± 3.64	18.50 ± 5.14	13.69 ± 8.91
	12 robotów			
	8 ładunków		16 ładunków	
	A*	Opt. num.	A*	Opt. num.
Czas	77	52	105	–
Łączna droga	245	283	409	–
Najkrótszy czas na ładunek	12	4	12	–
Najdłuższy czas na ładunek	36	35	42	–
Średni czas na ładunek	20.63 ± 8.63	14.00 ± 12.11	23.13 ± 8.84	–



(a)

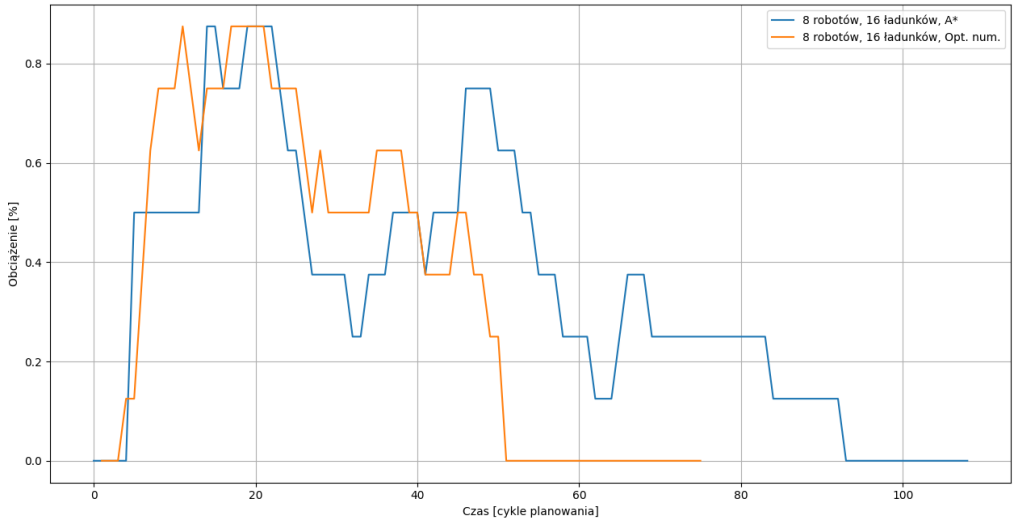


(b)

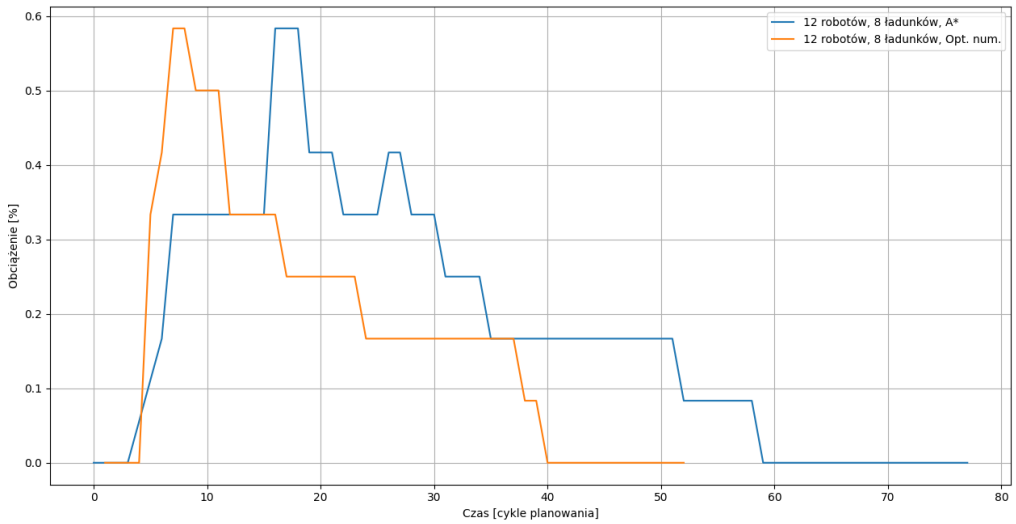


(c)

Rysunek 5.10 Porównanie przebiegów obciążenia robotów dla dwóch różnych metod planowania ścieżek robotów



(a)



(b)

Rysunek 5.11 Porównanie przebiegów obciążenia robotów dla dwóch różnych metod planowania ścieżek robotów

Metoda oparta o zmodyfikowany algorytm A* wymaga zastosowania odpowiednich mechanizmów zapobiegania i rozwiązywania zakleszczeń. Zwiększenie liczby robotów prowadzi do większego prawdopodobieństwa wystąpienia blokad i wymaga lepszych mechanizmów radzenia sobie z nimi. Brak odpowiedniego sposobu rozwiązywania zakleszczeń może spowodować, że metoda nie doprowadzi do rozwiązanie zadania. Z drugiej strony, metoda wykorzystująca optymalizację numeryczną nie wymaga dodatkowych mechanizmów rozwiązujących problem blokowania się robotów. Odpowiedni wybór funkcji kosztów daje możliwość skutecznego radzenia sobie z zakleszczeniami. Natomiast wykorzystanie optymalizacji numerycznej prowadzi do innych problemów, między innymi wynikających z możliwości występowania minimów lokalnych, które mogą doprowadzić do niemożliwości poprawnego zakończenia zadania.

Rozdział 6

Podsumowanie

Celem pracy było zaprezentowanie dwóch opracowanych algorytmów sterowania: algorytmu wykorzystujący zmodyfikowany algorytm A* pozwalający na heurystyczny dobór ścieżek robotów i algorytm oparty o optymalizację numeryczną oraz ich porównaniu. Porównanie dotyczyło wydajności czasowej oraz długości przebytej drogi przez wszystkie roboty. Wszystkie wymienione zadania zostały w pełni zrealizowane.

We wstępie do pracy zostały opisane różne rodzaje zadań związanych z wykorzystaniem roju robotów oraz został wykonany przegląd różnych metod i podejść stosowanych do sterowania rojem robotów. Następnie w rozdziale 2 zostało przedstawione zadanie przyjęte do realizacji. Podane zostały warunki określające poprawne wykonanie zadania oraz dodatkowe założenia pozwalające na uszczegółowienie przyjętego zadania. W dalszej kolejności zostały zaprezentowane dwa algorytmy: algorytm A* oraz algorytm optymalizacji numerycznej bazujący na metodzie MPC, które zostały wykorzystane w metodach sterowania rojem. Dalej zostało pokrótce opisane wykorzystane oprogramowanie. Jako pierwszy został przedstawiony ROS użyty do stworzenia sterowników robotów, następnie symulator ARGoS pozwalający na wykonanie eksperymentów symulacyjnych i badań nad opracowanymi metodami sterowania. W rozdziale 4 został przestawiony niższy z poziomów struktury sterownika rojem, sterownik pojedynczego robota. Opisane zostały funkcje realizowane przez lokalny sterownik robota, między innymi sposób realizowania ruchu przez roboty. Zaprezentowany został również sposób komunikacji między sterownikami lokalnymi a sterownikiem centralnym. Szczegółowy opis sposobu działania sterownika centralnego został przedstawiony w rozdziale 5. W pierwszej części rozdziału zostały przedstawione funkcjonalności sterownika. Następnie zostały przedstawione dwie opracowane metody planowania ścieżek robotów: opartą o zmodyfikowany algorytm A* oraz opartą o optymalizację numeryczną. Zostały przedstawione wynik badań wydajności poszczególnych metod oraz ich porównanie.

Obydwie metody sterowania rojem mogą zostać wykorzystane jako metody offline. W przypadku znajomości położenia ładunków i początkowej pozycji robotów oraz założeniu, że w przestrzeni roboczej nie znajdują się ruchome przeszkody, można dokonać symulacji działania roju. W wyniku symulacji uzyskane zostały kolejne sekwencje ruchu robotów, które następnie mogą zostać odtworzone w rzeczywistym systemie. Jako przykładowe zastosowanie przedstawionych metod w trybie offline można podać przypadek kiedy kilka robotów cyklicznie przewozi ładunki między stanowiskami produkcyjnymi. Roboty transportowe pracują w tej samej przestrzeni roboczej, przy czym środowisko pracy jest statyczne i nie znajdują się w nim inne obiekty. Roboty w takim przypadku mogą poruszać się cyklicznie po wcześniej wyliczonych ścieżkach. W takim przypadku istotne jest jednak zachowanie odpowiedniej synchronizacji ruchu, kolejne ruchy robotów mogą

zostać wykonane dopiero kiedy zostaną zakończone wszystkie ruchy robotów wynikające z poprzedniego kroku.

Wspomniane metody mogą również pracować w trybie online. Zastosowanie metody w trybie online jest jednak mocno uzależnione od wymaganych reżimów czasowych. Metoda wykorzystująca zmodyfikowany algorytm A* wymaga znacznie krótszego czasu obliczeń niż metoda oparta o optymalizację numeryczną. W przypadku ograniczeń sprzętowych oraz długości czasu obliczeń bardziej odpowiednią metodą będzie metoda wykorzystująca zmodyfikowany algorytm A*. Czas potrzebny do wykonania kolejnych cykli planowania w metodzie opartej o optymalizację numeryczną może być regulowany poprzez maksymalną liczbę iteracji algorytmu optymalizacji, jednak zmniejszanie limitu iteracji prowadzi do pogorszenia wyników poprzez uzyskiwanie częściowych predykcji (w najgorszym przypadku wszystkie punkty predykcji znajdują się w jednym punkcie węzłowym). Dzięki zastosowaniu struktury sterowania podzielonej na lokalne sterowniki robotów oraz sterownik centralny możliwe jest uniknięcie problemów związanych z czasem obliczeń wynikających z ograniczeń sprzętowych. Sterownik centralny może zostać uruchomiony na jednostce stacjonarnej posiadającej dużą moc obliczeniową, a polecenia mogą być przesyłane za pomocą sieci. Przykładowym zastosowaniem metod w trybie online może być wykorzystanie ich do sterowania automatycznymi wózkami transportowymi w magazynie. Wózki mogą zajmować się przewożeniem ładunków, na przykład ze strefy rozładunku samochodów ciężarowych do poszczególnych stref w magazynie wynikających z rodzaju ładunku obecnie przewożonego przez robota.

Wpływ na działanie przedstawionych metod sterowania ma środowisko pracy robotów. W przyjętym zadaniu przestrzeń robocza nie zawiera żadnych innych obiektów poza robotami. W rzeczywistych zastosowaniach możliwe jest występowanie statycznych oraz dynamicznych przeszkód. Położenie przeszkód statycznych może być znane przed rozpoczęciem zadania sterowania. W przypadku przeszkód statycznych, których położenie jest znane możliwe jest dostosowanie opracowanych metod sterowania poprzez odpowiednie dopasowanie siatki punktów węzłowych. W przypadku metody opartej o optymalizację numeryczną zmiana siatki punktów węzłowych może prowadzić do powstania nowych minimów lokalnych. W takim przypadku może zaistnieć dodatkowa potrzeba zmodyfikowania postaci funkcji kosztów. Właściwe wyznaczenie siatki punktów węzłowych wymaga uwzględnienia rozmiaru oraz możliwości ruchowych robotów oraz odpowiedniego dystansu od przeszkód i stref z ładunkami. Sam podział przestrzeni roboczej na siatkę może odbywać się automatycznie. Prowadzi to do możliwości modyfikacji siatki punktów węzłowych w przypadku nowo wykrytej przeszkody statycznej. Takie podejście daje możliwość przystosowania przedstawionych metod w przypadku środowiska, w którym są nieznane położenia statycznych przeszkód. W przypadku metody wykorzystującej zmodyfikowany algorytm A* zmiana siatki w trakcie działania algorytmu nie powinna prowadzić do znaczącego pogorszenia się jakość sterowania. W przypadku metody opartej o optymalizację numeryczną może zajść wcześniej opisany problem powstania minimów lokalnych prowadzących do uniemożliwienia poprawnego działania metody. Występowanie przeszkód dynamicznych w środowisku pracy robota może prowadzić do pogorszenia się jakości działania opracowanych metod. Metody sterowania mogą blokować pewne punkty węzłowe w pobliżu, których wykryto przeszkodę dynamiczną i przez to unikać kolizji. Takie zachowanie nie prowadzi natomiast do specjalnego uwzględniania dynamicznych przeszkód w przypadku planowania ścieżek (przeszkoda dynamiczna będzie traktowana, w danym cyklu planowania, tak jakby w danych węzłach znajdowały się inne roboty).

Na działanie metod, oprócz przeszkód, może mieć wpływ również rozmiar mapy przestrzeni roboczej i siatki punktów węzłowych. W przypadku realizowanego zadania liczba

punktów węzłowych wynosiła 100 (z pominięciem węzłów wewnątrz stref). Nie jest to duża liczba. Większa siatka węzłów oznaczałaby zwiększenie liczby węzłów w grafie przeszukiwanym w przypadku metody wykorzystującej zmodyfikowany algorytm A*. Dodatkowo większe środowisko pracy robotów może prowadzić do wydłużenia się odległości (zwiększenie liczby punktów węzłowych) między strefami. Obydwa czynniki mają wpływ na działanie metody i najprawdopodobniej doprowadziłyby do wydłużenia czasu obliczeń cyklu planowanie. Jednocześnie, dzięki wykorzystaniu heurystycznego algorytmu A*, metoda powinna ciągle zachowywać pierwotną złożoność obliczeniową pozwalającą na jej wykorzystanie w zadaniu planowania ścieżek. Skuteczność metod jednak w dużym stopniu będzie zależeć od zastosowanej funkcji heurystycznej. W przypadku metody opartej o optymalizację numeryczną zwiększenie rozmiaru mapy oraz siatki punktów węzłowych może wpłynąć na czas obliczeń. W przypadku wykorzystania mechanizmu wczesnego przerwania obliczeń po wykonaniu określonej liczby iteracji, czas obliczeń nie powinien znacząco wzrosnąć, natomiast uzyskane predykcje będą częściowe.

Innym zagadnieniem związanym ze środowiskiem roboczym jest sposób rozmieszczenia stref załadunkowych i rozładunkowych oraz ewentualnych przeszkód statycznych znanych przed rozpoczęciem zadania sterowania. Rozmieszczenie stref oraz przeszkód może przyczyniać się do powstania „wąskich gardeł" czyli punktów węzłowych, przez które roboty muszą przejechać żeby przemieścić się z jednego sektora przestrzeni roboczej do innego sektora. W najgorszym przypadku może istnieć tylko jeden punkt węzłowy pozwalający na przejazd między dwoma sektorami. W takich przypadkach rośnie prawdopodobieństwo występowania blokad robotów. Jeżeli to tylko możliwe należy projektować rozmieszczenie stref załadunku i rozładunku w taki sposób aby minimalizować liczbę „wąskich gardeł". Przyjęte w pracy zadanie nie jest odpowiednim przykładem takiego projektowania. Umieszczenie strefy załadunku w centralnej części przestrzeni roboczej przyczyniło się do powstania czterech miejsc, w których utrudniony jest ruch robotów. Taki układ stref pozwolił jednak na lepsze badanie opracowanych metod pod względem unikania i rozwiązywania zakleszczeń.

Badania metody opartej o optymalizację numeryczną pokazały, że metoda nie potrzebuje specjalnych mechanizmów pozwalających na rozwiązywanie problemu zakleszczeń. Inne wyniki zostały uzyskane dla metody wykorzystującej zmodyfikowany algorytm A*. Szczególnie w przypadku użycia roju składającego się z 16 robotów dochodziło do sytuacji, w których roboty blokowały możliwość wyjazdu ze strefy centralnej. Jako dalszy rozwój pracy można wskazać implementację różnych sposobów rozwiązania takiego problemu oraz ich przetestowanie. Pierwszym z rozwiązań może być wyznaczenie specjalnych stref, w których roboty mogą oczekiwać na wjazd do danej części strefy, jednocześnie nie blokując przejazdu między wjazdami poszczególnych stref. Takie rozwiązanie mogłoby okazać się skuteczne dla średniej liczby robotów (12), jednak z uwagi na ograniczone miejsce z każdej stron strefy centralnej mogłoby prowadzić do problemów związanych z brakiem miejsca przy większej liczbie robotów (16 - 20). Możliwe byłoby wykorzystanie miejsca z różnych stron strefy centralnej, jednak prowadziłoby to do konieczności wprowadzenia dodatkowego algorytmu kolejkowania i przemieszczania się między strefami przeznaczonymi dla oczekujących robotów. Inne podejście mogłoby polegać na stopniowym kierowaniu robotów na pozycje końcowe (zgodne z warunkiem zakończenia zadania) wraz z opróżnianiem z ładunków kolejnych części strefy centralnej. Takie rozwiązanie nie wymaga wprowadzenia dodatkowych mechanizmów. Można jednak zastanawiać się czy takie podejście nie prowadzić do obniżenia wydajności roju. Przykładowo, jeśli w jednej części strefy znajduje się wiele ładunków, a w pozostałych liczba ta jest mniejsza, ładunki z pozostałych części strefy zostaną szybko przewiezione co spowoduje, że trzy czwarte roju

zakończy swoją pracę. Prowadzi to do zmniejszenia liczby robotów mogących przewozić dane ładunki, ale jednocześnie zmniejsza liczbę robotów poruszających się wokół wjazdu do ostatniej części strefy i możliwość występowania blokad. Kolejny sposób rozwiązania problemu zakleszczeń mógłby polegać na nadaniu różnych priorytetów robotom. Roboty z wyższym priorytetem miałyby pierwszeństwo w przejeździe. Oznacza to, że w przypadku kiedy dwa roboty blokują swoje ruchy nawzajem poprzez wyznaczenie swoich ścieżek w przeciwnych kierunkach, robot o niższym priorytecie musi przerwać dążenie do swojego celu i wykonać ruch do sąsiedniego punktu węzłowego tak, aby umożliwić przejazd robotowi o wyższym priorytecie. Jeżeli wszystkie sąsiednie węzły są zajęte przez inne roboty to następuje sprawdzenie, który z sąsiadujących robotów może wykonać ruch pozwalający na zwolnienie węzła potrzebnego do odblokowania ścieżki. Jeżeli ponownie żaden ze sprawdzanych robotów nie może wykonać ruch to sprawdzani są kolejni sąsiedzi. Procedura trwa tak długo aż nie zostanie znaleziona kombinacja sekwencji dla robotów, których kolejne ruchy pozwolą na odblokowanie ścieżki przejazdu robota o wyższym priorytecie. Jeżeli zostanie znalezionych kilka różnych sekwencji zmian pozycji robotów wymagających tyle samo ruchów to wybierana jest ta, dla której suma priorytetów uczestniczących w niej robotów jest mniejsza. Dodatkowo odrzucane są kombinacje sekwencji zawierające roboty, które wykonały już ruch w danym cyklu planowania. W metodzie zakłada się, że robot o najwyższym priorytecie planuje swoją ścieżkę i rozpoczyna wykonanie zaplanowanego ruchu jako pierwszy. Jeżeli zaplanowany ruch może zostać wykonany bez przeszkód to kolejny robot o niższym priorytecie zaczyna planować swoją ścieżkę i wykonuje ruch. W przypadku kiedy ruch nie może zostać wykonany, wstrzymywane jest planowanie dla pozostałych robotów i badana jest kombinacja sekwencji ruchów robotów pozwalająca na odblokowywanie ruchu. W dalszej kolejności, jeżeli została znaleziona kombinacja sekwencji ruchów to jest ona wykonywana i następnie rozpoczynany jest ruch w zaplanowanej ścieżce. Jeżeli nie znaleziona została dozwolona kombinacja sekwencji ruchów (wszystkie znalezione kombinacje zakładają przesunięcie robota, który już zaplanował swoją ścieżkę i wykonał względem ścieżki ruch) to ruch robota, dla którego aktualnie planowana była ścieżka zostaje pominięty i rozpoczyna się planowanie dla robota o niższym priorytecie. Sama metoda rozwiązywania zakleszczeń zakłada, że przynajmniej jeden punkt węzłowy w całej siatce punktów węzłowych jest wolny. Jeżeli to założenie jest niespełnione to zadanie z założenia nie może być rozwiązane, żaden robot nie może wykonać ruchu. Taka metoda powinna pozwolić na poprawne działanie roju nawet w przypadku kiedy liczba robotów użytych w roju jest tylko o jeden mniejsza od liczby wszystkich punktów węzłowych w siatce. Słabą stroną zaproponowanej metody może być duża złożoność podczas wyszukiwania kombinacji sekwencji ruchów przy dużej liczbie robotów zajmujących węzły ze sobą sąsiadujące.

Każda z opracowanych metod sterowania rojem posiada różne charakterystyki. Metoda wykorzystująca zmodyfikowany algorytm A* pozwala na osiągnięcie dobrego czasu poszczególnych cykli planowania. Wykorzystanie w algorytmie funkcji heurystycznej pozwala na poprawienie wydajności obliczeniowej. Jednocześnie prowadzi do konieczności wyboru właściwej postaci funkcji. W przypadku siatki punktów węzłowych jako funkcja heurystyczna została wykorzystana norma taksówkowa. Wybór funkcji heurystycznej jest uwarunkowany rodzajem zadania oraz zastosowanej reprezentacji środowiska i może wymagać wykonania serii testów w celu ustalenia najlepszej postaci funkcji. Algorytm może wymagać zaimplementowania dodatkowych mechanizmów rozwiązujących problem zakleszczeń, szczególnie w przypadku zastosowania większej liczby robotów. Badania wydajności wskazały, że w porównaniu do metody opartej o optymalizację numeryczną, metoda wykorzystująca zmodyfikowany algorytm A* realizuje zadania w dłuższym czasie

mierzonym w cyklach planowania. Równocześnie łączna przebyta droga przez roboty jest krótsza w przypadku metody wykorzystującej zmodyfikowany algorytm A*. Metoda oparta o optymalizację numeryczną w porównaniu do drugiej metody wymaga dłuższego czasu obliczeń pojedynczego cyklu planowania. Zaletom metody jest brat konieczności implementacji specjalnych mechanizmów rozwiązywania problemu zakleszczeń. Trudności z zastosowaniem metody wiążą się z wykorzystaniem algorytmu optymalizacji numerycznej. Podstawowy problem wynika z konieczności wykorzystania odpowiedniej funkcji kosztu. Funkcja ta nie powinna zawierać minimów lokalnych, które mogą powodować blokowanie się robotów i uniemożliwiać poprawne wykonanie zadania. Wybranie odpowiedniej funkcji dla bardziej skomplikowanego zadania, w którym przestrzeń robocza posiada wiele zaułków, może być problematyczne. Dodatkowo wykorzystanie w zadaniu optymalizacji numerycznej ciągłej przestrzeni stanu wymusiło opracowanie sposobu interpretacji otrzymanych predykcji i zamiany ich na kolejne ruchy robotów.

Bibliografia

[1] Dokumentacja ROS. `http://wiki.ros.org/ROS`. Dostęp: 2021-05-14.

[2] R. Alami, S. Fleury, M. Herrb, F. Ingrand, F. Robert. Multi-robot cooperation in the martha project. *IEEE Robotics and Automation Magazine*, 5(1):36–47, 1998.

[3] T. Balch, R. Arkin. Behavior-based formation control for multi-robot teams. *Robotics and Automation, IEEE Transactions on*, 14:926 – 939, 01 1999.

[4] G. Beni, J. Wang. Swarm intelligence in cellular robotics systems. *Proceed. NATO Advanced Workshop on Robots and Biological Systems*, 1989.

[5] R. Brown, J. Jennings. A pusher/steerer model for strongly cooperative mobile robot manipulation. *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, wolumen 3, strony 562–568 vol.3, 1995.

[6] S. Edelkamp, S. Schrödl. *Heuristic Search - Theory and Applications.* Morgan Kaufmann, 2012.

[7] M. Erdmann, T. Lozano-Perez. On multiple moving objects. *IEEE Transactions on Robotics and Automation*, 2:477–521, 1987.

[8] T. Fukuda, S. Nakagawa, Y. Kawauchi, M. Buss. Self organizing robots based on cell structures - ckbot. *IEEE International Workshop on Intelligent Robots*, strony 145–150, 1988.

[9] D. D. Grossman. Traffic control of multiple robot vehicles. *IEEE Transactions on Robotics and Automation*, 5(5):491–497, 1988.

[10] L. Grüne, J. Pannek. *Nonlinear Model Predictive Control - Theory and Algorithms.* Springer-Verlag London, 2011.

[11] L. Joseph. *Mastering ROS for Robotics Programming - Design, Build, and Simulate Complex Robots Using the Robot Operating System and master its out-of-the-box functionalites.* Packt Publishing, 2014.

[12] K. Kant, S. W. Zucker. Toward efficient trajectory planning: the path-velocity decomposition. *The International Journal of Robotics Research*, 5(3):72–89, 1986.

[13] R. R. Murphy. Marsupial and shape-shifting robots for urban search and rescue. *IEEE Intelligent Systems*, 15(2):14–19, 2000.

[14] L. E. Parker. *Multiple Mobile Robot Teams, Path Planning and Motion Coordination in*, strony 5783–5800. Springer New York, 2009.

[15] M. Peasgood, C. Clark, , J. McPhee. A complete and scalable strategy for coordinating multiplerobots within roadmaps. *IEEE Transactions on Robotics*, 2008.

[16] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. D. Caro, F. Ducatelle, M.Birattari, L.Gambardella, M. Dorigo. Argos: a modular, parallel, multi-engine simulator for multi-robot systems. *Swarm Intelligence*, 6:271–295, 2012.

[17] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, T. Stirling, A. Gutiérrez, L. M. Gambardella, M. Dorigo. Argos: A modular, multi-engine simulator for heterogeneous swarm robotics. *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, strony 5027–5034, 2011.

[18] B. Siciliano, O. Khatib. *Handbook of Robotics*. Springer, 2008.

[19] A. Stroupe, A. Okon, M. Robinson, T. Huntsberger, H. Aghazarian, E. Baumgartner. Sustainable cooperative robotic technologies for human and robotic outpost infrastructure construction and maintenance. *Autonomous Robots*, 20(2):113–123, 2006.

[20] K. Sugawara, T. Watanabe. A study on foraging behavior of simple multi-robot system. *IEEE 2002 28th Annual Conference of the Industrial Electronics Society. IECON 02*, wolumen 4, strony 3085–3090, 2002.

Dodatek A

Załącznik do pracy

Do pracy została załączono płyta CD zawierającą w poszczególnych katalogach:

- Praca_magisterska.pdf — wersja cyfrowa pracy,
- Kod_zrodlowy/ros_workspace/src/ros_argos3 — paczka ROSowa zapewniająca komunikacje między węzłami ROSa a symulatorem ARGOS.
- Kod_zrodlowy/ros_workspace/src/testBot – paczka ROSowa zawierająca kod źródłowy sterowników lokalnych robotów oraz sterowników roju robotów, scenariusze testów, skrypty pozwalające na uruchomienie testów oraz pliki ze statystykami uzyskanymi podczas testów.