



POLITECHNIKA WARSZAWSKA

Wydział Mechatroniki

Praca przejściowa

Ireneusz Szulc

# Planowanie bezkolizyjnych tras dla zespołu robotów mobilnych

Opiekun pracy:  
prof. dr hab. Barbara Siemiątkowska

Warszawa, 2018

# Spis treści

<b>Spis treści</b>	<b>2</b>
<b>1 TODO</b>	<b>3</b>
<b>2 Konspekt pracy</b>	<b>4</b>
<b>3 Wstęp teoretyczny</b>	<b>6</b>
3.1 Koordynacja ruchu robotów . . . . .	6
3.2 Cel pracy . . . . .	6
3.3 Przestrzeń konfiguracyjna . . . . .	7
3.4 Metoda hill-climbing . . . . .	7
3.5 Metody planowania tras . . . . .	7
3.5.1 Metody zcentralizowane . . . . .	8
3.5.2 Metoda potencjałowa . . . . .	8
3.5.3 Podejście "Decoupled" . . . . .	8
3.6 Wybór priorytetów . . . . .	11
3.7 Podsumowanie . . . . .	11
3.7.1 Wnioski . . . . .	11
<b>4 Algorytmy wywodzące się z A*</b>	<b>12</b>
4.1 Algorytm A* . . . . .	12
<b>5 Podsumowanie</b>	<b>13</b>
<b>Bibliografia</b>	<b>14</b>
<b>Wykaz skrótów</b>	<b>15</b>
<b>Spis rysunków</b>	<b>15</b>

# Rozdział 1

## TODO

Karta tematu:

Temat pracy: Planowanie bezkolizyjnych tras dla zespołu robotów mobilnych

Temat pracy (w jęz. ang.): Path planning for a group of mobile robots

Zakres pracy:

- Projekt algorytmu wyznaczania trajektorii dla pojedynczego robota
- Algorytm detekcji i zapobiegania kolizjom między robotami
- Implementacja oprogramowania symulacyjnego
- Przeprowadzenie testów symulacyjnych

Podstawowe wymagania:

- Aplikacja powinna umożliwiać symulację ruchu robotów oraz definiowanie położenia przeszkód przez użytkownika.
- Planowanie tras dotyczy robotów holonomicznych.

# Rozdział 2

## Konspekt pracy

- Wstęp teoretyczny:
  - Cooperative Pathfinding
  - algorytm  $A^*$  - szczegółowo
  - przegląd metod planowania tras dla wielu robotów
  - artykuł o Cooperative Pathfinding, time-space  $A^*$
  - artykuł o wyznaczaniu priorytetów i metodach planowania tras (prezentacja): Path Coordination, time-space  $A^*$
  - metoda ładunków - problem minimów lokalnych
  - replanowanie po wykrciu kolizji (algorytm  $D^*$ )
  - algorytmy WHCA\* i IADPP
  - Reciprocal Collision Avoidance
  - metody przydziału priorytetów - zwiększanie i przeliczanie
  - metody zcentralizowane vs rozproszone (porównanie)
  - time-space  $A^*$ , heurystyki, Reservation Table
- generowanie mapy - labiryntu do testów: własny algorytm, teoria grafów, własności mapy
- metoda przydziału / zmiany priorytetów
- obszerne testy, porównanie wyników metod przy tych samych warunkach początkowych
- zastosowanie: ciasne korytarze, częsty problem kolizji, szpitale, transport dokumentów, paczek

- time-space  $A^*$  - pseudokod, schemat blokowy, własne heurystyki, modyfikacje
- ograniczenia - nałożone uproszczenia: ruch skośny, czas dyskretny, brak czasu na obrót
- Implementacja aplikacji - stack technologiczny: Java 8, Java FX, Spring, Spring Boot, testy jednostkowe junit, git, IntelliJ, Maven, Linux; schemat klas aplikacji

# Rozdział 3

## Wstęp teoretyczny

### 3.1 Koordynacja ruchu robotów

Koordynacja ruchu robotów jest jednym z fundamentalnych problemów w systemach wielu robotów. [1]

Kooperacyjne znajdowanie tras (ang. Cooperative Pathfinding) jest problemem planowania w układzie wieloagentowym, w którym to agenci mają za zadanie znaleźć bezkolizyjne drogi do swoich, osobnych celów. Planowanie to odbywa się w oparciu o pełną informację o środowisku oraz trasach pozostałych agentów. [4]

Problem kooperacyjnego znajdowania tras pojawia się często m.in. w grach komputerowych, gdzie należy wyznaczyć drogi dla wielu jednostek, które mogą blokować się wzajemnie. Algoritmy do wyznaczania bezkolizyjnych tras dla wielu agentów (robotów) mogą znaleźć również zastosowanie w szpitalach (np. roboty TUG i HOMER do dostarczania sprzętu na wyposażeniu szpitala [5]) oraz magazynach (np. roboty transportowe w magazynach firmy Amazon [6]).

### 3.2 Cel pracy

Przedmiotem niniejszej pracy jest przegląd metod rozwiązujących zagadnienie planowania bezkolizyjnych tras dla wielu robotów. Stanowi to również wstęp teoretyczny do zaprojektowania algorytmu i implementacji oprogramowania pozwalającego na symulację działania systemu planowania tras.

Praca skupia się na przypadkach, w których mamy do czynienia ze środowiskiem z dużą liczbą przeszkód (np. zamknięty budynek z licznymi ciasnymi korytarzami), gdzie problem blokowania się agentów prowadzi często do zakleszczenia. Należy wtedy zastosować nieco inne podejścia niż te, które sprawdzają się w przypadku otwartych środowisk, a które zostały opisane

np. w pracach: [2], [3]

W niniejszej pracy zajmować będziemy się rozwiązaniem problemu, w którym mamy pełną informację o mapie otoczenia oraz o określonym położeniu początkowym i celu dla każdego z robotów. Zadaniem algorytmu będzie wyznaczenie możliwie najkrótszej bezkolizyjnej trasy dla wszystkich robotów. Należy jednak zaznaczyć, że priorytetem jest dotarcie każdego z robotów do celu bez kolizji z innymi robotami, dopiero później chcemy, aby wyznaczone drogi były możliwie najkrótsze.

### 3.3 Przestrzeń konfiguracyjna

Przestrzeń konfiguracyjna to formalna, matematyczna przestrzeń będąca zbiorem możliwych stanów danego układu fizycznego. W zależności od rodzaju i liczby wyróżnionych parametrów stanu przestrzenie konfiguracyjne mogą mieć wiele wymiarów.

### 3.4 Metoda hill-climbing

Metoda hill-climbing jest rodzajem matematycznej optymalizacji, lokalną metodą przeszukiwania. Jest to iteracyjny algorytm, który zaczyna w wybranym rozwiązaniu problemu, następnie próbuje znaleźć lepsze rozwiązanie poprzez przyrostowe zmiany pojedynczych elementów rozwiązania. Jeśli zmiana przynosi lepsze rozwiązanie, jest wprowadzana do nowego rozwiązania. Kroki algorytmu powtarzane są dotąd, aż żadne “udoskonalenie” nie zostanie znalezione.

### 3.5 Metody planowania tras

Spośród metod wykorzystywanych do planowania tras dla wielu robotów można wyróżnić dwie zasadnicze grupy:

- **zcentralizowane** - drogi wyznaczane są dla wszystkich agentów na raz (jednocześnie). Metody te potrafią znaleźć optymalne rozwiązanie, ale często mają bardzo dużą złożoność obliczeniową. Z tego powodu wykorzystywane są heurystyki przyspieszające proces obliczania rozwiązania.
- **rozproszone** (ang. decoupled) - dla każdego robota droga wyznaczana jest osobno, w określonej kolejności, następnie rozwiązywane są konflikty (kolizje dróg). W pewnych przypadkach rozwiązanie może nie zostać znalezione, mimo, iż istnieje. Metody te najczęściej wiążą się z koniecznością przydzielenia priorytetów, co stanowi istotny problem, gdyż od wyboru priorytetów może zależeć zupełność algorytmu.

### 3.5.1 Metody zcentralizowane

Zaletami metod zcentralizowanych są:

- Planowanie w zbiorowej przestrzeni konfiguracyjnej
- Wyznaczenie **optymalnego** rozwiązania
- W praktyce: Heurystyka radzi sobie z ogromną złożonością przestrzeni konfiguracyjnej

Podejścia (bez gwarancji optymalności):

- Potential field techniques - Metoda potencjałowa
- Roadmap methods - przeszukiwanie grafu połączeń algorytmem Dijkstry

### 3.5.2 Metoda potencjałowa

Metoda potencjałowa (ang. Potential field techniques) *TODO*

### 3.5.3 Podejście "Decoupled"

Popularne podejścia omijające planowanie w wysoko wymiarowej zbiorowej przestrzeni konfiguracyjnej to techniki rozproszone i priorytetowane. Pomimo, że te metody są bardzo efektywne, mają dwie główne wady:

1. Nie są zupełne - czasami nie udaje się znaleźć rozwiązania, nawet gdy istnieje.
2. Wynikowe rozwiązania są często nieoptymalne.

Ponadto nie mówią, jak przypisywać priorytety do pojedynczych robotów.

W artykule [1] przedstawiono metodę do optymalizowania układu priorytetów dla rozproszonych i priorytetowanych technik planowania. Proponowana metoda wykonuje randomizowane przeszukiwanie z techniką hill-climbing do znalezienia rozwiązania i do skrócenia całkowitej długości ścieżek. Technika została zaimplementowana i przetestowana na prawdziwych robotach oraz w rozległych testach symulacyjnych. Wyniki eksperymentu pokazały, że metoda potrafi znacząco zmniejszyć liczbę niepowodzeń i znacznie zmniejszyć całkowitą długość tras dla różnych priorytetowanych i rozproszonych metod planowania dróg, nawet dla dużych zespołów robotów.

Algorytm (niezupełny):

1. Wyznaczenie optymalnej ścieżki dla każdego robota **niezależnie**



2. Przydział priorytetów (opcjonalne)
3. Próba rozwiązania możliwych konfliktów między ścieżkami

Podejścia:

- Path coordination
- Planning in the configuration time-space:
  - V-Graph algorithm
  - Potential fields
  - A\*

### Path coordination

Idea:

- Utrzymanie robotów na ich indywidualnych, optymalnych ścieżkach
- Pozwolenie na zatrzymanie się, ruch naprzód, a nawet cofanie się, ale tylko **wzdłuż trajektorii** w celu uniknięcia kolizji

W praktyce:

- Wymagany wariant z ustaleniem priorytetów
- Złożoność  $O(n \cdot m \cdot \log(m))$ ,  $m$  - maksymalna liczba stanów podczas planowania

### Konieczność wyboru priorytetów

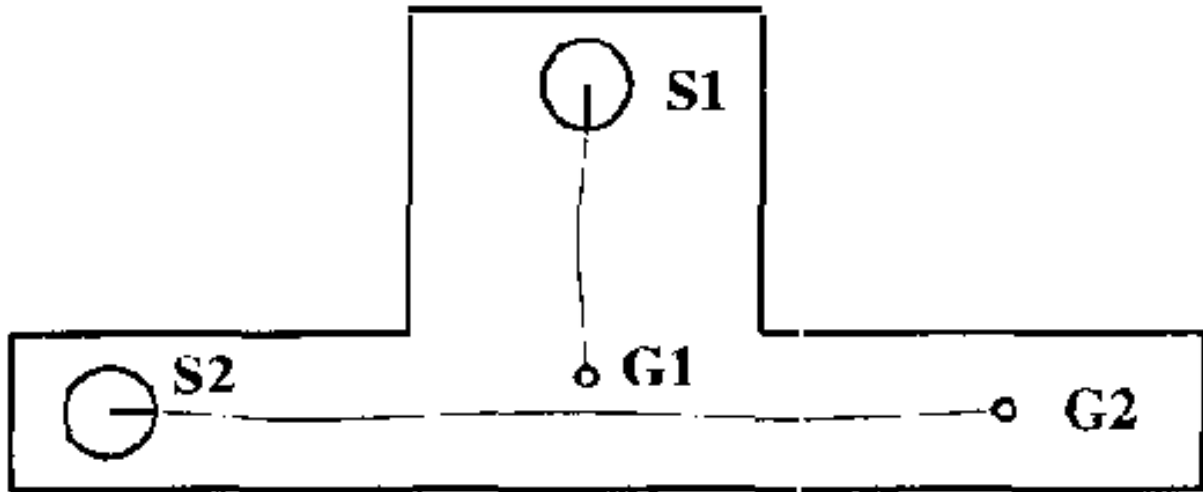
### Zastosowanie A\* w planowaniu dróg dla wielu robotów

Algorytm:

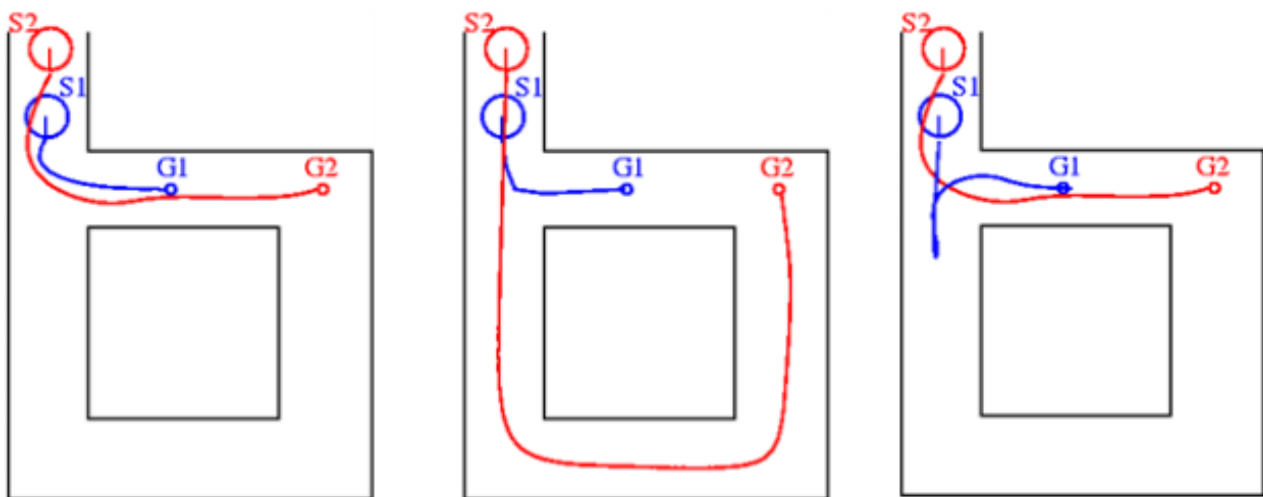
- Przypisanie priorytetów do poszczególnych robotów
- Wykonywanie kroków A\* z rozpatrywaniem czasu i przestrzeni - podział otoczenia na siatkę pól, zapisywanie prawdopodobieństwa zajęcia pola w danej chwili

Złożoność:

- $O(n \cdot m \cdot \log(m))$ ,  $m$  - maksymalna liczba stanów podczas planowania (lista otwartych)



Rysunek 3.1: Sytuacja, w której żadne rozwiązanie nie zostanie znalezione, jeśli robot 1 ma wyższy priorytet niż robot 2



Rysunek 3.2: Niezależne planowanie optymalnych tras dla 2 robotów; suboptymalne rozwiązanie, gdy robot 1 ma wyższy priorytet; rozwiązanie, gdy robot 2 ma wyższy priorytet

## Wpływ układu priorytetów na długość tras

### 3.6 Wybór priorytetów

#### Elastyczny dobór priorytetów

Obecne techniki pozostawiają dowolny wybór priorytetów lub korzystają z ustalonego z góry stałego układu kolejności robotów.

### 3.7 Podsumowanie

#### 3.7.1 Wnioski

- Testy przeprowadzono w dwóch różnych środowiskach (acykliczne / cykliczne). Losowo generowano punktów startu i celu
- Zaprezentowano metodę doboru priorytetów dla rozproszonych metod planowania dróg dla grupy robotów mobilnych.
- Zaproponowane podejście to randomizowana metoda, która cyklicznie zamienia kolejność robotów w celu znalezienia sekwencji, dla której można wyznaczyć plan dróg oraz w celu minimalizacji całkowitej długości tras.
- Jest to algorytm, który może być zatrzymany w dowolnym momencie i może zawsze zwrócić obecnie najlepsze rozwiązanie.
- Metoda została zaimplementowana i przetestowana na rzeczywistych robotach i w rozległych testach symulacyjnych dla dwóch różnych metod planowania dróg oraz dla dużej liczby robotów.
- Wyniki eksperymentu pokazały, że metoda potrafi znacząco zmniejszyć liczbę niepowodzeń (gdy żadne rozwiązanie nie zostaje znalezione) i znacznie zmniejszyć całkowitą długość tras.

## Rozdział 4

# Algorytmy wywodzące się z $A^*$

### 4.1 Algorytm $A^*$

Kiedy pojedynczy agent dokonuje znalezienia drogi do wyznaczonego celu, prosty algorytm  $A^*$  sprawdza się bardzo dobrze. Jednak kiedy wiele agentów porusza się w tym samym czasie, to podejście może się nie sprawdzić, powodując wzajemne blokowanie się i zakleszczenie jednostek. Rozwiązaniem tego problemu może być kooperacyjne znajdowanie tras. Jednostki będą mogły skutecznie przemieszczać się przez mapę omijając trasy wyznaczone przez inne jednostki oraz schodząc innym jednostkom z drogi, jeśli to konieczne. [4]

*TODO*

## Rozdział 5

## Podsumowanie

# Bibliografia

- [1] Thrun S. Bennewitz M., Burgard W. *Optimizing Schedules for Prioritized Path Planning of Multi-Robot Systems*. 2001.
- [2] Roszkowska E. Mówiński K. *Sterowanie hybrydowe ruchem robotów mobilnych w systemach wielorobotycznych*. Postępy Robotyki, 2016.
- [3] Siemiątkowska B. *Uniwersalna metoda modelowania zachowań robota mobilnego wykorzystująca architekturę uogólnionych sieci komórkowych*. 2009.
- [4] Silver D. *Cooperative Pathfinding*. 2005.
- [5] Roboty TUG i HOMER firmy Aethon. <http://www.aethon.com/tug/tughealthcare/>. Dostęp: 2018-01-02.
- [6] As amazon pushes forward with robots. <https://www.nytimes.com/2017/09/10/technology/amazon-robots-workers.html>. Dostęp: 2018-01-02.

# Wykaz skrótów

API    Application Programming Interface

SDK    Software Development Kit

# Spis rysunków

3.1	Sytuacja, w której żadne rozwiązanie nie zostanie znalezione, jeśli robot 1 ma wyższy priorytet niż robot 2 . . . . .	10
3.2	Niezależne planowanie optymalnych tras dla 2 robotów; suboptymalne rozwiązanie, gdy robot 1 ma wyższy priorytet; rozwiązanie, gdy robot 2 ma wyższy priorytet . . . . .	10