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

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 wykruciu kolizcji (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 warunkacj 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. [6]

Problem kooperacyjnego znajdowania tras pojawia się często m.in. w grach komputerowych, gdzie należy wyznaczyć drogi dla wielu jednostek, które mogą blokować się wzajemnie. Algorytmy 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 [7]) oraz magazynach (np. roboty transportowe w magazynach firmy Amazon [8]).

3.2 Cel pracy

Przedmiotem niniejszej pracy jest przegląd metod rozwiązujacych 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 sią 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: [4], [5]

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 zostaje znalezione.

3.5 Metody planowania tras

Spośród metod wykorzystywanych do planowania tras dla wielu robotów można wyróżnić dwie zasadnicze grupy [3]:

- **zcentralizowane** - drogi wyznaczane są dla wszystkich agentów na raz (jednocześnie). Metody te potrafią znaleźć wszystkie możliwe rozwiązania (w szczególności to optymalne), ale mają bardzo dużą złożoność obliczeniową ze względu na ogromną przestrzeń przeszukiwania. Z tego powodu wykorzystywane są heurystyki przyspieszające proces obliczania rozwiązania.

- **rozproszone** (ang. decoupled lub distributed) - Podejście to dekomponuje zadanie na niezależne lub zależne w niewielkim stopniu problemy dla każdego agenta. 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 ich wyboru może zależeć zupełność algorytmu. Nie należy mylić tej metody z zagadnieniem typu Non-Cooperative Pathfinding, w którym agenci nie mają wiedzy na temat pozostałych planów i muszą przewidywać przyszłe ruchy pozostałych robotów. W podejściu rozproszonym agenci mają pełną informację na temat stanu pozostałych robotów, lecz wyznaczanie dróg odbywa się w określonej kolejności.

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 zaimplemenotwana i przetestowana na prawdziwych robotach oraz w rozległych testach symulacyjnych. Wyniki eksperyentu pokazały, że metoda potrafi znacząco zmniejszyć liczbę niepowodzeń i znacznie zmniejszyć całkowitą długość tras dla różnych priorytetowanych i rozproszoncyh 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*

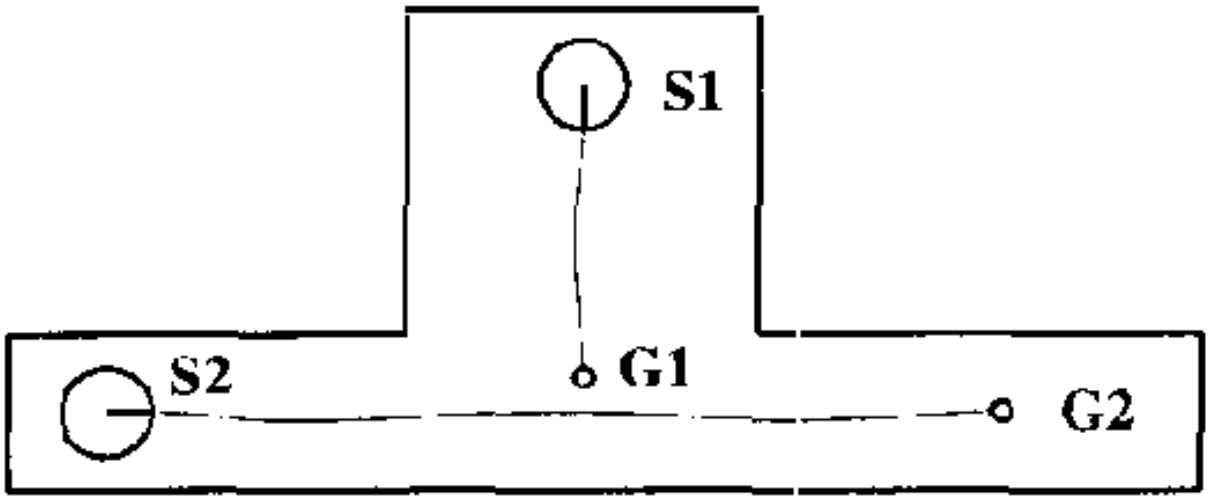
Path coordination

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



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

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 zajętości pola w danej chwili

Złożoność:

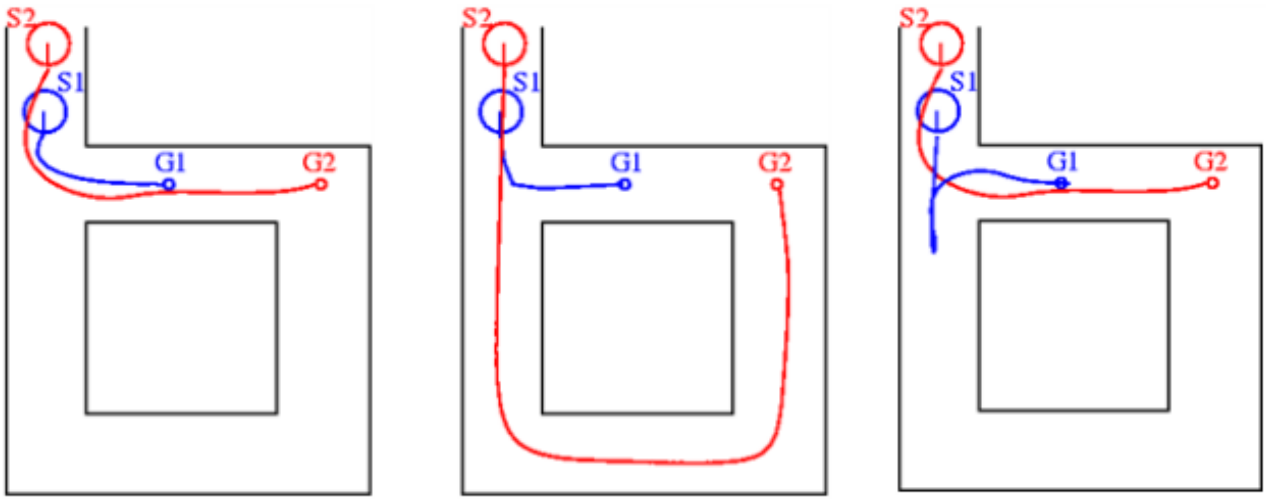
- $O(n \cdot m \cdot log(m))$, m - maksymalna liczba stanów podczas planowania (lista otwartych)

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.



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

3.7 Podusmowanie

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 eksperyentu pokazały, że metoda potrafi znacząco zmniejszyć liczbę niepowodzeń (gdy żadne rozwiązanie nie zostaje znalezione) i znacznie zmniejszyć całkowitą długość tras.

Rozdział 4

Algorytmy oparte na A*

Kiedy pojedynczy agent dokonuje znalezienia drogi do wyznaczonego celu, prosty algorytm A* sprawdza się bardzo dobrze. Jednak w przypadku, gdy 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. [6]

Zagadnienie znajdowania drogi jest ważnym elementem sztucznej inteligencji zaimplementowanej w wielu grach komputerowych. Chociaż klasyczny algorytm A* potrafi doprowadzić pojedynczego agenta do celu, to jednak dla wielu agentów wymagane jest zastosowanie innego podejścia w celu unikania kolizji. Algorytm A* może zostać zaadaptowany do replanowania trasy na żądanie, w przypadku wykrycia kolizji tras (Local Repair A* lub D*). Jednak takie podejście nie jest zadowalające pod wieloma względami. Na trudnych mapach z wieloma wąskimi korytarzami i wieloma agentami może to prowadzić do zakleszczenia agentów w wąskich gardłach lub do cyklicznego zapętlenia ruchu agentów. [6]

W systemach czasu rzeczywistego istotne jest również, aby rozwiązanie problemu planowania tras uzyskać w określonym czasie. Z tego powodu nie są wykorzystywane techniki zcentralizowanego planowania tras, ze względu na zbyt dużą złożoność obliczeniową.

4.1 Algorytm A*

TODO

4.2 Local Repair A*

W algorytmie Local Repair A* (LRA*) każdy z agentów znajduje drogę do celu, używając algorytmu A*, ignorując pozostałe roboty oprócz ich obecnych sąsiadów. Roboty zaczynają podążać wyznaczonymi ścieżkami do momentu, aż kolizja z innym robotem jest nieuchronna. Wtedy następuje ponowne przeliczenie drogi pozostałej do przebycia, z uwzględnieniem nowo napotkanej przeszkody.

Możliwe (i całkiem powszechne [6]) jest uzyskanie cykli (tych samych sekwencji ruchów powtarzających się w nieskończoność), dlatego zazwyczaj wprowadzane są pewne modyfikacje, aby rozwiązać ten problem. Jedną z możliwości jest zwiększanie wpływu losowego szumu na wartość heurystyki. Kiedy agenci zachowują się bardziej losowo, prawdopodobne jest, że wydostaną się z problematycznego położenia i spróbują podążać innymi ścieżkami.

Algorytm ten ma jednak sporo poważnych wad, które szczególnie ujawniają się w trudnych środowiskach z dużą liczbą przeszkód. Wydostanie się z zatłoczonego wąskiego gardła może trwać bardzo długo. Prowadzi to również do ponownego przeliczania trasy w prawie każdym kroku. Wciąż możliwe jest również odwiedzanie tych samych lokalizacji w wyniku zapętleń.

4.3 Algorytm D*

D* (Dynamic A* Search) jest przyrostowym algorytmem przeszukiwania. Jest modyfikacją algorytmu A* pozwalającą na szybsze replanowanie trasy w wyniku zmiany otoczenia (np. zajmowania wolnego pola przez innego robota). Wykorzystywany jest m.in. w nawigacji robota do określonego celu w nieznanym terenie. Początkowo robot planuje drogę na podstawie pewnych założeń (np. nieznany teren nie zawiera przeszkód). Podążając wyznaczoną ścieżką, robot odkrywa rzeczywisty stan mapy i jeśli to konieczne, wykonuje ponowne planowanie trasy na podstawie nowych informacji. Często wykorzystywaną implementacją (z uwagi na zoptymalizowaną złożoność obliczeniową) jest algorytm D* Lite [9].

4.4 Cooperative A*

4.5 Hierarchical Cooperative A*

4.6 Windowed Hierarchical Cooperative A*

TODO

Rozdział 5

Podsumowanie

Bibliografia

[1] Bennewitz M.; Burgard W.; Thrun S. *Optimizing Schedules for Prioritized Path Planning of Multi-Robot Systems*. 2001.

[2] B. Hart P. E.; Nilsson N. J.; Raphael. *A Formal Basis for the Heuristic Determination of Minimum Cost Paths*. IEEE Transactions on Systems Science and Cybernetics SSC4, 1968.

[3] Latombe J. *Robot Motion Planning*. Boston, MA: Kluwer Academic, 1991.

[4] Mówiński K.; Roszkowska E. *Sterowanie hybrydowe ruchem robotów mobilnych w systemach wielorobotycznych*. Postępy Robotyki, 2016.

[5] Siemiątkowska B. *Uniwersalna metoda modelowania zachowań robota mobilnego wykorzystująca architekturę uogólnionych sieci komórkowych*. 2009.

[6] Silver D. *Cooperative Pathfinding*. Proceedings of the First Artificial Intelligence and Interactive Digital Entertainment Conference, 2005.

[7] Roboty TUG i HOMER firmy Aethon. `http://www.aethon.com/tug/tughealthcare/`. Dostęp: 2018-01-02.

[8] As amazon pushes forward with robots. `https://www.nytimes.com/2017/09/10/technology/amazon-robots-workers.html`. Dostęp: 2018-01-02.

[9] S. Koenig; M. Likhachev. *D* Lite*. Proceedings of the AAAI Conference of Artificial Intelligence, 2002.

Wykaz skrótów

API	Application Programming Interface
SDK	Software Development Kit

Spis rysunków