I Krajowa Konferencja
Algorytmy Ewolucyjne
Murzasichle, 12-15.06.1996

Planowanie ścieżki mobilnego robota

Krzysztof Trojanowski[1] Zbigniew Michalewicz,[1,2]

[1] Instytut Podstaw Informatyki Polskiej Akademii Nauk,
ul. Ordona 21, 01-237 Warszawa, Polska
[2] Department of Computer Science, University of North Carolina,
Charlotte, NC 28223, USA

1 Wstęp

Jednym z głownych nurtów w dziedzinie robotyki jest zagadnienie autonomicznych, mobilnych robotów. Najogólniej mówiąc, ich działanie sprowadza się do przyjęcia zadania i samodzielnego wykonania go. Takie wymaganie stawia przed projektantami robotów wiele problemów do rozwiązania. Pośród nich problem planowania ścieżki i nawigacji jest jednym z najistotniejszych.

Niniejsza praca będzie miała za cel przedstawić problem ruchu robota, pokazać dotychczasowe jego rozwiązania i zaprezentować nowe.

2 Definicja zagadnienia

Dla zdefiniowania problemu potrzebne jest określenie celu jaki został wskazany i warunków przy jakich ma zostać osiągnięty. Problem został podzielony na podstawowe składniki dla jego lepszego zrozumienia i analizy.

2.1 Symbole

Poniżej przedstawiony jest zbiór symboli, jakie będą wykorzystywane w dalszej części tekstu; jest to jednocześnie podstawowy zbiór pojęć, wykorzystywanych w dziedzinie robotyki [6]:

A — *mobilny* robot;
W — *przestrzeń* robota;
$B_i(i = 1,..,n)$ — i-ta przeszkoda z n przęszkód znajdujących się w przestrzeni W;
q — *konfiguracja* obiektu w przestrzeni, zawiera pozycję w przyjętym układzie współrzędnych, oraz orientację — kierunek (przyjmujemy, że każdy obiekt posiada swoją oś; kierunkiem jest kąt między tą osią a osiami układu współrzędnych przestrzeni);
q_{init} — konfiguracja startu;
q_{goal} — konfiguracja celu;
$A(q)$ — podzbiór przestrzeni W zajęty przez robota A w konfiguracji q;
C — *przestrzeń konfiguracji* robota A, jest sumą wszystkich możliwych

konfiguracji q tego robota w przestrzeni W;
τ — *ścieżka* robota, wiodącą od q_{init} do q_{goal} , taka że:

$$\tau : [0, 1] \rightarrow C,$$

przy czym: $\tau(0) = q_{init}$ i $\tau(1) = q_{goal}$.

Dodatkowo wszystkie przeszkody B, oraz robot A mogą być określane wspólnym mianem *obiektów.*

2.2 Założenia

Rozważając najprostszy model problemu możemy przyjąć założenie, że robot jest jedynym ruchomym obiektem w środowisku, oraz ignorowane są ograniczenia wynikające z fizycznej natury środowiska (odbicia w wyniku zderzeń, czas i droga hamowania, ograniczenia na kąt zakrętów etc). Dla tak przyjętych założeń mówimy o robocie, że jest *obiektem swobodnym (free-flying object).*

Upraszczamy również opis przestrzeni, przyjmując, że robot jest obiektem o niezmienialnym kształcie, a wszystkie obiekty, występujące w przestrzeni, są wielokątami, oraz są nieruchome (za wyjątkiem robota oczywiście).

2.3 Podstawowy problem planowania ruchu robota

Mając dany punkt wyjścia i kierunek, oraz punkt docelowy i kierunek ścieżki robota A w przestrzeni W, określić ścieżkę τ , zaczynającą się od punktu wyjścia i kończącą w punkcie docelowym i będącą ciągiem pozycji i kierunków robota A, takich że A nie wchodzi w kontakt z przeszkodami B_i.

2.4 Reprezentacja przeszkody w przestrzeni konfiguracji

Rozwiązania dla postawionego problemu poszukujemy w przestrzeni konfiguracji robota C. Dla odnalezienia w przestrzeni C ścieżki poprawnej, tj. takiej, która nie wchodzi w kontakt z istniejącymi przeszkodami, należy odwzorować przeszkody B z przestrzeni W do przestrzeni konfiguracji C i określić podzbiór tej przestrzeni, mogący zawierać ścieżki poprawne (podzbiór nie zawierający przeszkód).

Każdą przeszkodę $B_i (i = 1, .., n)$ z przestrzeni W odwzorowujemy na przestrzeń C, tworząc C-przeszkody:

$$CB_i = \{q \in C : A(q) \cap B_i \neq 0\}.$$

Wówczas:

$$C_{free} = C - \bigcup_{i=1}^{q} CB_i = \{q \in C : A(q) \cap (\bigcup_{i=1}^{q} B_i) = 0\}$$

jest określany jako *wolna przestrzeń.* Każda konfiguracja q w przestrzeni C_{free} jest nazywana *konfiguracją poprawną.*

Ścieżką poprawną w przestrzeni C pomiędzy dwoma poprawnymi konfiguracjami q_{init} i q_{goal} jest ścieżka τ taka, że:

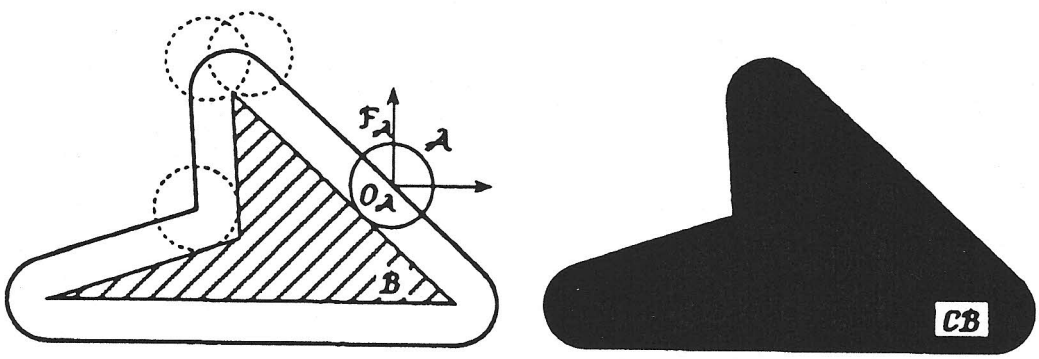
$$\tau : [0,1] \rightarrow C_{free}$$

przy czym: $\tau(0) = q_{init}$ i $\tau(1) = q_{goal}$.

2.5 Założenia dotyczące kształtu i orientacji robota

Jeżeli przyjmiemy założenie, że robot A jest pojedynczym punktem, wtedy jego orientacja w przestrzenie jest bez znaczenia. Przestrzeń konfiguracji C dokładnie odpowiada przestrzeni W. C-przeszkody są identyczne jak przeszkody B_i z przestrzeni W.

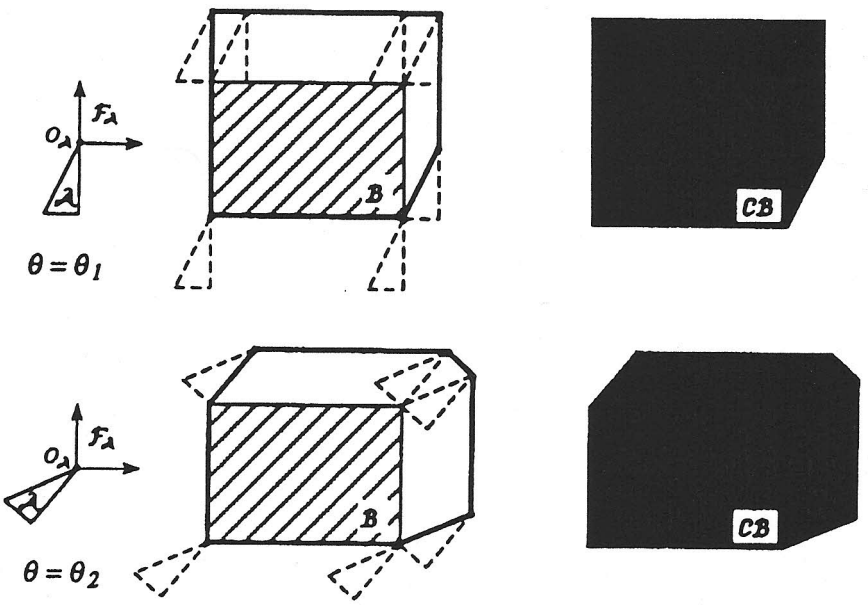
Drugim interesującym nas przypadkiem jest ten, gdy robot jest dyskiem lub obiektem nie mogącym się obracać. Gdy robot jest dyskiem a przeszkoda B jest wielokątem, to odwzorowaniem tej przeszkody w C jest podobna figura, pogrubiona o średnicę robota. Przedstawia to rysunek 1.



Rysunek 1. Odwzorowanie przeszkody B z przestrzeni W do C, przy założeniu, że robot jest dyskiem. $O(A)$ jest obszarem przestrzeni W zajmowanym przez robota. $F(A)$ — układ odniesienia dla orientacji robota A.

Gdy robot może jedynie przesuwać się w przestrzeni bez możliwości obrotu, odwzorowanie przeszkód jest również niezbyt złożone. Nowa figura jest również pogrubieniem poprzedniej, ale o obszar zakreślany przez robota, sunącego wokół figury i jednocześnie pozostającego z nią w kontakcie. Przedstawia to rysunek 2.

Dla dalszych rozważań przyjmiemy, że zachodzi przypadek pierwszy (robot jest punktem) lub drugi (robot jest dyskiem lub ma dowolny kształt, ale nie ma możliwości obrotu). Będzie to podstawą do przyjęcia założenia, że po odwzorowaniu przestrzeni W na C wszystkie przeszkody są wielokątami, a robot jest reprezentowany przez pojedynczy punkt. Ponadto, dla uproszczenia prezentacji wyników przyjmujemy, że przestrzeń, w której poszukujemy rozwiązań jest dwuwymiarowa. Dla tak przyjętych warunków prowadzone będą dalsze rozważania.



Rysunek 2. Odwzorowanie przeszkody B z przestrzeni W do C, przy założeniu, że robot jest wielokątem o ustalonej orientacji θ . $F(A)$ — układ odniesienia dla orientacji robota A.

3 Rozwiązania klasyczne

W tej sekcji pracy przedstawione zostaną dotychczas proponowane, klasyczne już podejścia do problemu planowania. Można je podzielić na kilka podstawowych grup [6]:

1. metody sieciowe,
2. podział przestrzeni na komórki, oraz
3. metody pola potencjału.

Metody te zasadniczo różnią się od siebie. Ponadto, w ramach każdej z metod istnieją różnorodne wariacje, koncentrujące się na różnych aspektach problemu. Prawie wszystkie przyjmują założenia dotyczące reprezentacji przestrzeni, oraz właściwości robota, tak jak to zostało przedstawione w rozdziale 2.

3.1 Metody sieciowe

Metody te opierają się na pewnej ogólnej idei: przetwórzmy przestrzeń poszukiwań tak, aby otrzymała formę sieci, z węzłami i połączeniami między nimi, i aby mieściła się wyłącznie w wolnej przestrzeni konfiguracji C_{free}. W ten sposób dostajemy gotowy zbiór scieżek poprawnych — sieć R, z których należy teraz wybrać najoptymalniejszą. Różnice w podejściach polegają tu głównie na sposobie wygenerowania odpowiedniej sieci. Dalej planowanie ścieżki sprowadza się do określenia w sieci punktu wyjścia oraz punktu docelowego i przeszukania zbioru rozwiązań w celu wybrania najlepszego. Jako przykłady metod sieciowych można wymienić [1][2][6]:

1. graf widzialności,
2. diagramy Voronoi,
3. metodę swobodnej drogi, oraz
4. metodę sylwetki.

Wszystkie wymienione powyżej metody, opierają się na podstawowym założeniu stworzenia sieci R w przestrzeni C_{free}. Różnią się za to w sposobach uzyskania tej sieci.

3.2 Podział przestrzeni na komórki

Podstawową zasadą w tej metodzie jest podział przestrzeni C_{free} na zbiór obszarów — komórek, które nie mają ze sobą części wspólnych. Dwa z tych obszarów zawierają punkt wyjścia i punkt dojścia. Następnie, po dokonaniu podziału, generowany jest graf połączeń pomiędzy obszarami. W tym grafie określana jest sekwencja, łącząca dwa wybrane obszary (nazywana kanałem), i stanowi ona podstawę do wygenerowania ścieżki [12].

Przy podziale przestrzeni na obszary istotne jest zachowanie dwóch następujących założeń:

1. geometryczny kształt każdego z obszarów powinien być na tyle prosty, aby łatwo było wyznaczyć ścieżkę, biegnącą przez ten obszar, oraz
2. nie powinno być trudne sprawdzenie istnienia połączenia między dwoma dowolnymi obszarami i — w przypadku sąsiedztwa — wyznaczenie ścieżki biegnącej przez oba te obszary.

3.3 Metody pola potencjału

W metodach pola potencjału robot traktowany jest jako punkt, który przemieszcza się pod wpływem działania sił. Ruch robota jest procesem iteracyjnym. W każdym kolejnym kroku iteracji wektor siły, wynikający z kierunku i natężenia pola w danym punkcie przestrzeni, jest uważany za wskazanie najlepszego kierunku dalszego poruszania się. Wtedy robot przemieszcza się o pewien niewielki odcinek i w swym nowym położeniu znów analizuje oddziałujące na niego pole. Takie podejście okazało się bardzo skuteczne dla przypadków, gdy położenie i kształt przeszkód w przestrzeni nie jest znany, ale rozpoznawany na bieżąco w czasie ruchu.

Metoda pola potencjału charakteryzuje się wysoką efektywnością poszukiwań w czasie rzeczywistym i to niezależnie od kształtu figur, jednak jest dosyć nieodporna na lokalne minima pola. Z tego powodu łączy się ją z innymi technikami, np. grafowymi. Mimo to jest ona chętnie implementowana w praktycznie działających planerach, właśnie ze względu na swą szybkość i niezależność od kształtu przeszkód [4][5][6][10].

Funkcja potencjału U jest sumą dwóch elementarnych funkcji:

$$U(q) = U_{att}(q) + U_{rep}(q)$$

gdzie:

U_{att} — przyciągająca — jest funkcją wynikającą z konfiguracji q_{init} i q_{goal},

U_{rep} — odpychająca — wynika z kształtu i położenia przeszkód.

Przy stosowaniu tej metody w praktyce najwięcej wysiłku włożono w metody takiego tworzenia pola potencjału aby nie miało ono minimów lokalnych. Okazało się przy tym, że jest to bardzo trudny problem i możliwe są rozwiązania jedynie dla niezbyt złożnych kształtów figur oraz tylko 2- lub 3-wymiarowych przestrzeni.

4 Rozwiązanie wykorzystujące algorytmy ewolucyjne

4.1 Omówienie działania algorytmu ewolucyjnego

Algorytmy ewolucyjne opierają sie w swym działaniu na wykorzystaniu metod probabilistycznych do stworzenia *populacji* rozwiązań i znalezienia rozwiązania najlepszego dla tej populacji. Mamy tu więc populację osobników, które dzielą się na osobniki poprawne — tj. należące do zbioru rozwiązań — i niepoprawne — tj. nie spełniające ograniczeń, nałożonych na zbiór rozwiązań problemu. Każdemu osobnikowi przypisana jest jego *funkcja dopasowania*, czyli jego wartość dla danego problemu. Taka ocena rozwiązań w populacji pozwala nam rozpoznawać i odróżniać osobniki lepsze od gorszych. Korzystając z tej funkcji dopasowania tworzona jest nowa populacja osobników. Przedtem jednak pewna część populacji ulega przekształceniom, tworzone są nowe osobniki, być może lepsze od swoich rodziców. W każdym algorytmie istnieje zbiór operatorów ewolucyjnych, których zadaniem jest właśnie przekształcać i tworzyć te nowe osobniki. I dopiero z takiego zbioru: osobników starych i nowych, zgodnie z ustalonymi wcześniej regułami wybierana jest nowa populacja. Taki iteracyjny proces przekształcania i tworzenia nowych osobników powtarza się wielokrotnie, aż do spełnienia warunku końcowego, którym może być np. przekroczona maksymalna ilość iteracji lub jakość uzyskanego najlepszego osobnika, etc. [8] Ogólna struktura algorytmu ewolucyjnego jest przedstawiona na rysunku 3.

4.2 Wykorzystanie algorytmu ewolucyjnego do planowania ścieżki

Problem planowania ścieżki przy przyjętych założeniach dotyczących ograniczeń z rozdziału omawiającego metody klasyczne, można opisać w taki sposób, aby możliwe było zastosowanie algorytmu ewolucyjnego [7]. Taki opis musi spełniać warunek konieczności utworzenia pięciu niezbędnych składników algorytmu:

1. reprezentacja potencjalnych rozwiązań,
2. sposób stworzenia populacji początkowej,
3. funkcja ewaluacji osobników,
4. operatory genetyczne, oraz
5. wartości do sparametryzowania algorytmu (rozmiar populacji, prawdopodobieństwa uruchamiania poszczególnych operatorów, etc).

```
Algorytm ewolucyjny
begin
    t ← 0
    inicjalizuj P(t)
    ewaluuj P(t)
    while (not warunek kończący) do
    begin
        t ← t + 1
        wybierz P' from P(t − 1)
        przekształć P'
        ewaluuj P'
        wybierz P(t) from P(t − 1) ∪ P'
    end
end

t — numer iteracji
P(t) — populacja rozwiązań w iteracji t
```

Rysunek 3. Struktura algorytmu ewolucyjnego

Reprezentacja potencjalnych rozwiązań. Przyjęto, że pojedynczy osobnik reprezentuje jedną całą ścieżkę. Taki osobnik w algorytmie ewolucyjnym traktowany jest jak chromosom. Składa się z uporządkowanej listy genów i podlega genetycznym operacjom — krzyżowania, mutacji, etc. Tu pojedynczy osobnik jest uporządkowaną listą punktów z przestrzeni C. Ta lista jest listą wierzchołków łamanej, tworzącej ścieżkę. Kolejne punkty tej listy są nazywane węzłami. Drogę od węzła do następnego węzła robot przebywa po linii prostej. Takie zdefiniowanie osobnika umożliwia istnienie ścieżek poprawnych i niepoprawnych. Ścieżkami niepoprawnymi są te, dla których odcinek drogi, poprowadzony pomiędzy dowolną parą dwóch kolejnych węzłów, przecina przynajmniej jedną przeszkodę, lub też, jeżeli którykolwiek z węzłów ścieżki jest położny wewnątrz przeszkody. Pojedynczy gen — punkt w przestrzeni i jednocześnie węzeł ścieżki — składa się z kilku liczb. Oprócz współrzędnych (x, y) zawiera informację o tym, czy następny odcinek ścieżki jest poprawny, oraz czy sam punkt jest poprawny (nie leży wewnątrz przeszkody). Pierwszym punktem każdego chromosomu jest punkt startowy ścieżki robota, natomiast ostatnim — cel, do którego ma on dotrzeć.

Sposób tworzenia populacji początkowej. Podstawowym sposobem tworzenia populacji początkowej jest losowe wygenerowanie zbioru uporządkowanych list punktów jako populacji początkowej. Jest to oczywiście podejście najprostsze. Można również w jakiś sposób wstępnie przygotować zbiór ścieżek, tak aby algorytm nie tracił czasu na znajdowanie ścieżek poprawnych, ale od razu przeszedł do ich optymalizacji. Tu od razu narzucają się jako metody inicjalizacyjne, metody sieciowe. Po utworzeniu sieci — np. grafu widzialności lub diagramu Voronoi — następuje generacja zbioru ścieżek, będących różnymi frag-

mentami grafu, zawsze łączącymi punkt początkowy z docelowym. W ten sposób dostajemy od razu na wstępie populację różnych, dość dobrych ścieżek poprawnych. Inną metodą inicjalizacji populacji, tak aby otrzymać zbiór osobników poprawnych, może być reperacja wszystkich osobników niepoprawnych z całej losowo wybranej populacji odpowiednim operatorem opisanym w dalszej częśsci artykułu.

Funkcja ewaluacji osobników. Dobre zdefiniwanie tej funkcji sprawia zawsze najwięcej trudności. Niewątpliwie powinna silnie zależeć od długości drogi, ale również od gładkości i odległości od przeszkód [9]. Dla reprezentacji chromosomu jako p:

$$p =< m_1, m_2, .., m_n >$$

gdzie m_i jest pojedynczym i-tym węzłem ścieżki o długości n węzłów, istnieje następująca funkcja ewaluacji:

$$PathCost(p) = w_d * \sum_{i=1}^{n-1} dist(m_i) + w_s * \sum_{i=2}^{n-1} smooth(m_i) + w_c * \max_{i=2}^{n-1} clear(m_i)$$

gdzie:
$dist(m_i)$ — jest odległością odcinka między i-tym a i+1-szym węzłem ścieżki, jeżeli jest on poprawnym, lub pewną stałą C_1 w przeciwnym przypadku;
$smooth(m_i)$ — jest gładkością danego węzła, będącą funkcją kąta θ_i między odcinkami (m_{i-1}, m_i) a (m_i, m_{i+1});
$clear(m_i)$ — określa jakość węzła pod względem odległości od przeszkód, i ma wartość C_2 jeżeli m_i leży bliżej którejkolwiek przeszkody niż pewna ustalona odległość θ, lub — w przeciwnym przypadku — różnicą między najbliżej znajdującą się przeskodą a wartością θ.

Stałe w_d, w_s i w_c to wagi, jakie zostają przypisane do trzech odpowiednich składników sumy z wzoru na funkcję ewaluacji.

Operatory genetyczne. Do zbioru operatorów należą wszystkie klasyczne operatory genetyczne. Zostały one jednak nieco zmodyfikowane w celu lepszego dopasowania do struktur, na których działają. Dodatkowo zostało dodanych kilka nowych operatorów, specyficznych dla rozwiązywanego problemu [11]:

- Krzyżowanie,
- Mutacja,
- Dodanie genu,
- Usunięcie genu,
- Zamiana pozycji genów,
- Reperacja choromosomu, oraz
- Miękkie krzyżowanie.

Krzyżowanie. Jest to operacja identyczna, jak w algorytmach genetycznych. W dwóch chromosomach wybierane są punkty cięcia. Ze względu na zmienną długość chromosomu nie muszą to być te same punkty względem początku, ale zupełnie dowolne. Następnie chromosomy wymieniają się odciętymi fragmentami list genów. Punkt cięcia nie musi być wybierany zawsze losowo. Można tu przyjmować dowolne kryteria wyboru.

Mutacja. W interpretacji ścieżki operator ten losowo przesuwa węzeł o pewien odcinek w dowolnym kierunku. Można tu zaproponować kilka sposobów działania takiego operatora. Może to być zmiana współrzędnych (x, y) punktu o $\delta x, \delta y$, przy czym o znaku wartości δx i δy decyduje losowana wartość binarna. W zależności od wartości $|\delta x|$ i $|\delta y|$ względem rozmiarów przestrzeni C i przeszkód można tu mówić o lokalnej mutacji (gdy wartość jest mała), mającej na celu np. wygładzenie ścieżki, lub globalnej, mającej na celu wprowadzenie radykalnej zmiany w ścieżce. Innym sposobem mutowania operatora jest losowanie długości promienia r z przedziału $(0, R\rangle$ i kąta θ z przedziału $(0, 2\pi)$ liczonego względem układu współrzędnych przestrzeni C dla określenia nowego położenia mutowanego punktu. Zastosowanie takiego podejścia — cylindrycznego układu współrzędnych — dla określenia nowego położenia punktu, daje większą możliwość modyfikacji położenia punktu. Ponadto zmienna w czasie (np. malejąca) maksymalna długość promienia — R (mechanizm podobny do stosowanego w symulowanym wyżarzaniu) daje możliwość poprzez mutację szukania zupełnie nowych ścieżek na początku procesu i optymalizacji tychże ścieżek na jego końcu (zmiany spowodowane mutacją z czasem maleją i z globalnych stają się lokalne).

Dodawanie węzła. Operacja dodaje do ścieżki losowo wygenerowany węzeł. Miejsce w ścieżce, w którym ma być dodany nowy węzeł jest również wybierane losowo, aczkolwiek można tu zmieniać reguły, np. zwiększając prawdopodobieństwo wybrania niektórych odcinków ścieżki (np. tych, które są niepoprawne).

Usuwanie węzła. Operacja usuwa dowolny węzeł ze zbioru węzłów ścieżki. Tutaj również można rozpatrywać metody dające większe szanse usunięcia węzłów, do których dochodzą, lub od których odchodzą niepoprawne odcinki ścieżki.

Zamiana węzłów. Operacja zamiany kolejności dwóch dowolnych węzłów w ścieżce.

Reperacja ścieżki. Operator ten obejmuje odcinki ścieżki, które przecinają conajmniej jedną przeszkodę. W takim przypadku po ustaleniu pierwszego punktu przecięcia odcinka z przeszkodą w tym punkcie dodawany jest nowy węzeł ścieżki. Aby węzeł nie został umieszczony na samej krawędzi przeszkody, jest on odsunięty od ściany przeszkody o pewną ustaloną odległość θ. Po utworzeniu nowego węzła zaczynają być dodawane kolejne odcinki ścieżki (i odpowiednio kolejne węzły) biegnące wzdłuż ścian przeszkody i okrążające ją. Kolejne odcinki są dodawane tak długo, póki nie napotkają na swej drodze i przetną ścieżkę naprawianą (lub nie przekroczony zostanie limit ilości węzłów dla pojedynczej ścieżki). Po napotkaniu ścieżki nowy odcinek łączy się z nią (jest dodawany jeszcze jeden nowy węzeł) , a fragment ścieżki znajdujący się wewnątrz przeszkody jest usuwany. Tym sposobem ścieżka omija przeszkodę, a w konsekwencji zmienia swój status z niepoprawnej na poprawną i wchodzi do zbioru

potencjalnych rozwiązań problemu.

Miękkie krzyżowanie. Operator ten przypomina nieco klasyczne krzyżowanie. Jeżeli dwie ścieżki przecinają się w choć jednym punkcie, to w tym punkcie przecięcia dodawany jest do obu ścieżek nowy węzeł, a następnie pozostałe fragmenty ścieżek są zamieniane. Jeżeli punktów przecięć jest kilka, miejsce nowego węzła jest wybierane losowo spośród nich. Jeżeli punktów przecięć brak, operacja nie może zostać zrealizowana.

Zbiór parametrów. Do zbioru parametrów należą: rozmiar populacji, warunek końca iteracji, oraz wszystkie prawdopodobieństwa i wagi dla operatorów ewolucyjnych. Ponadto rozmiar chromosomu — ścieżki — choć nie jest stały, lecz dowolny, jednak nie jest dłuższy niż pewna ustalona wartość maksymalna [11].

Inne możliwości wzbogacenia algorytmu. Dodatkowo opisany powyżej algorytm ewolucyjny może zostać wyposażony w mechanizm pamięci genetycznej. W tym zastosowaniu polegałby on na umieszczeniu w osobniku z populacji dodatkowej informacji pochodzącej od jego rodziców. Mogą to być np. kopie ich chromosomów. Ta informacja może być uwzględniana przez operatory genetyczne. Mogą one w swym działaniu wykorzystywać nie tylko chromosom danego osobnika, ale również chromosomy, które otrzymał od rodziców, jeżeli są one oceniane jako bardziej wartościowe. Przechowywany materiał genetyczny może być uwzględniony również przez funkcję ewaluacji osobników i wpływać na wyliczoną wartość danego osobnika.

5 Rozszerzenia problemu i ocena prezentowanych metod

Dotychczas prezentowane wszystkie rozwiązania problemu planowania ścieżki mobilnego robota były efektywne i dawały pożądane rezultaty dla określonego wcześniej zbioru założeń — ograniczeń, jakie przedstawiono w rozdziale 2. Spróbujmy teraz poszerzyć nasz zbiór oczekiwań wobec przedstawionych metod i porównajmy w nowym świetle ich możliwości. Do zbioru rozszerzeń zaliczymy [6]:

1. istnienie w przestrzeni C dodatkowo nieznanych przeszkód, które są rozpoznawane w miarę zbliżania się do nich,
2. możliwość samodzielnego przemieszczania się przeszkód w czasie, przy czym zmiana położenia może być znana lub nie,
3. planowanie aktywne, tj. wykonywane w trakcie zmierzania robota do celu,
4. sytuacja, kiedy postawiono przed robotem cel wielokryterialny, tj. gdy istotne jest nie tylko dotarcie do określonego punktu w przestrzeni, ale również spełnienie innych warunków dodatkowych.

W kolejnych podsekcjach omówione zostaną powyższe rozszerzenia z perspektywy przedstawionych wcześniej metod klasycznych i ewolucyjnych.

5.1 Metody klasyczne

Dodatkowe nieznane przeszkody. Przy założeniu istnienia w przestrzeni nierozpoznanych przeszkód, zarówno metody sieciowe, jak i podziału na komórki są bezradne. Pojawienie się nowej przeszkody wymaga rozpoczęcia budowy sieci i planowania ścieżki od początku. Jedynie metoda pola potencjału radzi sobie w takim przypadku dość dobrze.

Przeszkody ruchome w czasie. Przeszkody ruchome w czasie wymagają wprowadzenia do opisu przestrzeni C dodatkowego wymiaru: czasu. Generowanie sieci z takim dodatkowym wymiarem przy założeniu, że znamy sposób zmiany położenia każdej z ruchomych przeszkód powoduje jedynie bardzo duży wzrost złożoności obliczeniowej algorytmu generowania sieci i poszukiwania najkrótszej ścieżki. Mamy tu bowiem do czynienia ze ścieżką zmienną w czasie, a więc w trakcie planowania wystąpi konieczność określenia czasu potrzebnego do pokonania kolejnych etapów ścieżki. Dla przypadku kiedy przyszła zmiana położenia ścieżki jest nieznana, algorytm po zaobserwowaniu każdej kolejnej zmiany musi rozpoczynać planowanie od początku. To samo dotyczy również grafów budowanych z podziału przestrzeni na komórki. Znów jedynie metoda pola potencjału radzi sobie dobrze z takim rozszerzeniem.

Planowanie aktywne. Planowanie w trakcie poruszania się — nawigacja robota — w przypadku metod sieciowych jak i grafowych niesie ze sobą ryzyko, że trzeba będzie wycofać się z raz podjętej drogi, jeżeli algorytm pracujący w trakcie ruchu robota oceni, że przyjęta droga jest nieoptymalna. W tych metodach robot zachowuje się trochę jak tramwaj. Jeżeli ruszy jedną trasą, nie może nagle skręcić w lewo lub w prawo, żeby sobie skrócić drogę. Będzie musiał dojechać do najbliższego węzła, lub wrócić do poprzedniego. Przy tym rozszerzeniu znów swoją wyższość okazują metody pola potencjału, które mogą na początku wskazać kierunek ruchu robota i potem, w miarę jego przemieszczania się, lokalnie precyzować kierunek jego ruchu, mając znany kierunek globalny.

Wielokryterialność. Metody sieciowe i grafowe są w zasadzie metodami jednokryterialnymi. Poszukują jedynie w miarę krótkiej poprawnej ścieżki do osiągnięcia punktu docelowego. Wymagania dodatkowe, takie jak np. aby na danym obszarze przestrzeni robot przebywał jak najkrócej, albo dokonywał jak najmniej skrętów nie są w ogóle możliwe do wyrażenia. Również metoda pola potencjału w tym przypadku ma słabe możliwości. Aczkolwiek tu jeszcze istnieje możliwość, żeby wartością pola stymulować robota, tak aby — jeżeli to możliwe — np. omijał niektóre miejsca mimo że są drożne. Jednak podnosi to silnie złożoność obliczeniową całego algorytmu.

5.2 Metoda wykorzystująca algorytmy ewolucyjne

W tej metodzie trzy pierwsze rozszerzenia nie wprowadzają żadnej zmiany w samej strukturze algorytmu. Na bieżąco aktualizowana mapa przestrzeni odno-

towuje fakt przemieszczenia się przeszkód i w tym momencie niektóre ze ścieżek tracą na swojej wartości (np. stają się niepoprawne) a inne zyskują. Poruszający się robot na bieżąco reaguje na zmiany w przestrzeni wybierając tą ścieżkę, która w danej chwili jest najlepsza. Jednocześnie po wykonaniu każdego drobnego kroku punkty startu wszystkich ścieżek są aktualizowane, tak aby zawierać aktualne położenie robota. Posiadana populacja nie jest usuwana ani zmieniana, następuję jedynie potrzeba powtórnej ewaluacji wszystkich ścieżek, dla określenia ich nowych wartości w kontekście zmian przestrzeni. Do czwartego rozszerzenia — wielokryterialności — metoda też jest doskonale przygotowana. Ewentualne dodatkowe warunki, które musi uwzględnić robot, "zaszywane" są w funkcji ewaluacji i wpływają na ocenę ścieżek. W tym momencie o wartości ścieżki świadczy już nie tylko jej skuteczność w dotarciu do celu, ale również inne kryteria mają na nią możliwość wpływu.

5.3 Podsumowanie porównań

Po przyjęciu dodatkowych rozszerzeń, które niewątpliwie przybliżają proces planowania do warunków rzeczywistych, widać możliwości jakie daje nam metoda wykorzystująca algorytmy ewolucyjne. Jedyna konkurująca z nią metoda pola potencjału ma jednak pewne słabe strony. Przede wszystkim cały czas jest słabo odporna na minima lokalne, dla tego też musi być wspierana dodatkowo innymi metodami, a ponadto jej złożoność obliczeniowa po wprowadzeniu rozszerzeń silnie wzrasta. Dla metody ewolucyjnej nowe wymagania w sposób niejako naturalny dają się wpisać w jej mechanizm i nie wymagają żadnych dodatkowych modyfikacji.

6 Wnioski

Opisana powyżej metoda planowania ścieżki wykorzystująca algorytmy ewolucyjne, została zaimplementowana. Choć projekt badawczy, którego tematem jest planowanie ścieżki robota, jest dopiero w swojej początkowej fazie, jednak istnieje już działający program — Evolutionary Planner/Navigator, w którym uruchomione zostały wszystkie opisane powyżej operatory ewolucyjne. Jest to program napisany w języku C++ dla środowiska UNIX. W trakcie realizacji projektu planowane jest wprowadzenie do programu mechanizmów pamięci genetycznej i określenie ich znaczenia dla efektywności algorytmu przy uwzględnieniu nawigacji robota, tj. planowania w trakcie poruszania się po ścieżce, oraz istnienia przeszkód nieznanych i rozpoznawanych dopiero przy zbliżeniu się do nich na pewien dystans — zasięg pola widzenia robota.

Podziękowania

Powyższy artykuł powstał w ramach projektu badawczego nr 8T11C 010 10 pt."Zastosowania algorytmów ewolucyjnych do problemów planowania drogi i nawigacji" finansowanego przez Komitet Badań Naukowych.

Bibliografia

1. Alexopoulos, Ch., Griffin, P., M., Correspondence — Path Planning for a Mobile Robot, *IEEE Transactions on Sytems, Man, and Cybernetics*, 22(2), March-Apr. 1992
2. Brooks, R., A., Solving the Find-Path Problem by Good Representation of Free Space, *IEEE Transactions on Sytems, Man, and Cybernetics*, SMC-13(3), March-Apr. 1983
3. Foux, G., Heymann, M., Bruckstein, A., Two-Dimensional Robot Navigation Among Unknown Stationary Polygonal Obstacles, *IEEE Transactions on Robotics and Automation*, 9(1), Feb. 1993
4. Hwang, Y., K., Ahuja, N., A Potential Field Approach to Path Planning, *IEEE Transactions on Robotics and Automation*, 8(1), Feb. 1992
5. Kim, J. O., Khosla, P., K., Real-Time Obstacle Avoidance Using Harmonic Potential Functions, *IEEE Transactions on Robotics and Automation*, 8(3), June 1992
6. Latombe, J., C., *Robot Motion Planning*, Kluwer Academic Publishers, 1991.
7. Lin, H.,-S., Xiao, J., Michalewicz, Z., Evolutionary Navigator for a Mobile Robot, Proc. IEEE Int. Conf. Robotics and Automation, San Diego, May 1994, pp. 2199-2204
8. Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, 3rd edition, 1996.
9. Michalewicz, Z., Xiao, J., *Evaluation of Paths in Evolutionnary Planner/Navigator*, Proceedings of of the 1995 International Workshop on Biologically Inspired Evolutionary Systems, Tokyo, Japan, May 30-31, 1995, pp.45-52
10. Rimon, E., Koditschek, D., E., Exact Robot Navigation Using Artificial Potential Functions, *IEEE Transactions on Robotics and Automation*, 8(5), Oct. 1992
11. Xiao, J., Michalewicz, Z., Zhang, L., *Operator Performance of Evolutionary Planner/Navigator*, Proceedings of of the 3rd IEEE ICEC, Nagoya, May 20-22, 1996,
12. Zielinsky, A., A Mobile Robot Exploration Algorithm, *IEEE Transactions on Robotics and Automation*, 8(6), Dec. 1992

This article was processed using the LaTeX macro package with LLNCS style