Dynamiczne planowanie drogi z użyciem impulsowych sieci neuronowych

Luźne tłumaczenie pracy *Dynamic Path Planning with Spiking Neural Networks* autorów Ulrich Roth, Marc Walker, Arne Hilmann, Heinrich Klar na potrzeby seminarium sieci neuronowe i statystyka odbywającego się w semestrze letnim 2004/2005 w Instytucie Informatyki Uniwersytetu Wrocławskiego:

Łukasz Gorgol (nr indeksu 129887)

1. Wprowadzenie

Sieci przetwarzające impulsy (ang. *spike-processing networks*) znajdują zastosowanie głównie w rozdzielaniu obiektów od tła i klasyfikowaniu obiektów. Cechują się dynamicznym zachowaniem i komunikacją za pomocą impulsów, oraz równoległym działaniem sieci.

Problem planowania drogi jest istotny we wszystkich zastosowaniach, w których robot powinien niezależnie odnajdywać drogę. Celem algorytmu planującego drogę jest przemieszczenie się robota z pewnego punktu do miejsca docelowego omijając przeszkody i minimalizując długość drogi. Większość prac podchodzących do tego problemu zakłada statyczne, znane środowisko; metody te opierają się na przeszukiwaniu grafu.

Planowanie drogi w nieznanym, dynamicznym środowisku jest daleko bardziej złożonym problemem. Metody globalne wyznaczają całą drogę od nowa po zmianie w środowisku, co ze względu na czas działania jest niedopuszczalne do zastosowania w systemach czasu rzeczywistego. Sporo podejść do dynamicznego wyznaczania drogi opiera się na wygenerowaniu początkowej drogi na podstawie znanych informacji i próbie lokalnego omijania przeszkód w miarę ich rozpoznawania – zmieniając prędkość, bądź podążając w kierunku celu po obwodzie przeszkody. Podejścia te nie gwarantują jednak znalezienia najkrótszej drogi, ani nawet jakiejkolwiek drogi.

Niniejsze opracowanie prezentuje nowy algorytm wyznaczania drogi, zwany algorytmem radarowym, oraz jego implementację z wykorzystaniem neuronów impulsowych. Algorytm działa dynamicznie i posiada zdolność ciągłej adaptacji drogi w zmieniającym się środowisku.

2. Radarowy algorytm wyznaczający drogę

a. Definicja problemu

Niech Ψ będzie n-wymiarową przestrzenią roboczą. N wymiarów odpowiada n stopniom swobody robota. Przestrzeń robocza opisana jest n-wymiarową sześcienną kratownicą. Każdy sześcian reprezentuje punkt P_i punkt w przestrzeni roboczej:

$$n = \prod_i n_i \qquad \Psi = \bigcup_{i=1}^{n} p_i$$

Zbiór Θ punktów-przeszkód zawiera wszystkie te punkty przestrzeni Ψ, których robot nie może osiągnąć. Zbiór Θ tworzony jest poprzez zmapowanie tak wszystkich rzeczywistych przeszkód, jak i przeszkód konfiguracyjnych (pojawiających się wskutek aktualnych ograniczeń robota) na Ψ. Dotyczy to również takich konfiguracji, które wymagają, aby część robota dotknęła lub przeniknęła przez jakiś obiekt.

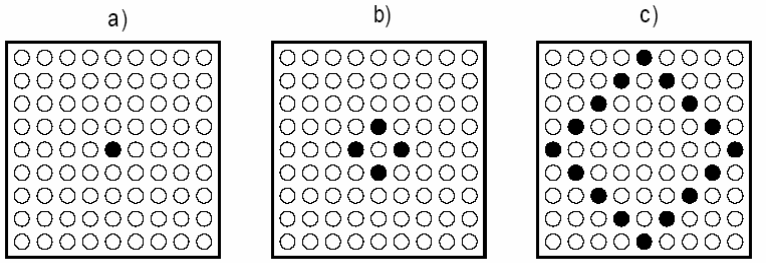
Mając dane punkt startowy S i punkt docelowy T zadaniem algorytmu jest stworzenie posortowanej listy Ω przyległych pól wiodących z S do T, spełniających warunek $\Omega \cap \Theta = \varnothing$.

b. Algorytm wyznaczania drogi

Idea algorytmu radarowego jest następująca: w przestrzeni Ψ dane są punkty startowy S i docelowy T. W regularnych odstępach czasu punkt T wysyła falę przebiegającą ze stałą prędkością przestrzeń Ψ. Czoło fali nie przedostaje się przez przeszkody. Sąsiednie polu S pole P osiągane jako pierwsze przez falę jest częścią optymalnej drogi do T. Dzieje się tak, ponieważ fala rozchodzi się we wszystkich kierunkach z jednostajną prędkością, tak więc jeżeli pole P jest osiągnięte najpierw, to musi leżeć na krótszej drodze do źródła fali niż wszystkie pozostałe pola, do których fala dociera później. P jest więc dodawane do drogi Ω i staje się nowym punktem startowym przy szukaniu dalszej drogi. W ten sposób droga zwiększana jest o jeden punkt z każdym wypuszczeniem fali aż do momentu osiągnięcia punktu T, kiedy powstaje cała optymalna droga łącząca punkty S i T.

i. Zachowanie statyczne

Rysunek pierwszy przedstawia rozchodzącą się falę w przestrzeni pozbawionej przeszkód.



Rysunek 1 Rozchodzenie się czoła fali w przestrzeni bez przeszkód

Rysunek 1a przestawia przestrzeń Ψ w czasie *t* gdy cel jest aktywowany do stworzenia fali. Rysunek 1b przedstawia początkowe czoło fali w czasie *t+1*. Rysunek 1c przedstawia czoło fali w czasie *t+4*.

Czoło fali jest zbiorem punktów w przestrzeni Ψ. Aktywność każdego punktu $P \in \Psi$ w czasie *t* wyznaczana jest z *reguły rozchodzenia*:

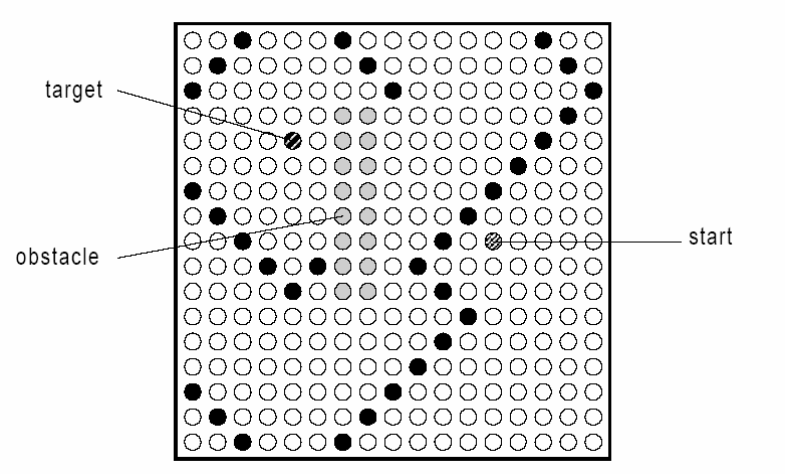
$$active(P,t) := \neg obstacle(P) \wedge nhv\_active(P,t-1) \wedge \neg active\_in(P,t,n)$$

(reguła rozchodzenia)

gdzie:

$active(P,t)$	prawda, jeśli P jest aktywne w czasie t
$obstacle(P)$	prawda, jeśli $P \in \Theta$
$nhv\_active(P,t-1)$	prawda, jeśli bezpośredni pionowy lub poziomy sąsiad P jest aktywny w czasie t
$active\_in(P,t,n)$	prawda, jeśli P było aktywne co najmniej raz w czasie *t-1, ..., t-n*

Poziomymi lub pionowymi sąsiadami punktu P w znaczeniu *nhv_active* są dwa bezpośrednio sąsiednie pola na każdej osi współrzędnych. Tak więc, w *N*-wymiarowej przestrzeni P ma $2N$ poziomych lub pionowych sąsiadów.



Rysunek 2 Dwa czoła fali w przestrzeni z przeszkodą

Rysunek 2 przedstawia dwa czoła fali w przestrzeni z przeszkodą. Czoło osiągnęło pole sąsiednie do pola startowego S. Punkt P dodawany jest do drogi Ω, jeśli spełniona jest *reguła drogi*:

$$path(P) := active(P) \wedge cursp\_neighbor(P) \wedge (P \neq S)$$

(reguła drogi)

gdzie:

$active(P,t)$	prawda, jeśli P jest aktywne
$cursp\_neighbor(P)$	prawda, jeśli bezpośredni sąsiad P oznaczony jest jako *current_starting_point*

Ostatni warunek $(P \neq S)$ zapobiega budowaniu drogi poza polem T. Kiedy pole P zostaje dodane do drogi, oznaczenie *current_starting_point* przenoszone jest na P. W danej chwili tylko jeden punkt przestrzeni może być tak oznaczony.

Do odcięcia wyznaczonej drogi Ω w punkcie P służy *akcja odcięcia*:

$CUT(P), P \in \Omega$:	usuń P i wszystkie następne punkty z Ω
$CUT(P), P \notin \Omega$:	nic nie rób

(akcja odcięcia)

Druga część akcji, dla $P \notin \Omega$, musi zostać zdefiniowana do implementacji *szumu odcinającego*.

ii. Zachowanie dynamiczne

W celu zdefiniowania dynamicznego zachowania algorytmu należy rozpatrzyć trzy przypadki:

- ruch punktu startowego S,
- ruch punktu docelowego T i
- ruch przeszkody.

Ruch punktu startowego

Jeśli punkt startowy S podąża wyznaczoną drogą Ω, przesuwając się do punktu $S' \in Q$ należy usunąć z Ω wszystkie punkty poprzedzające S'. Jeśli S nie podąża drogą Ω, np. jeśli $S' \notin Q$, należy w całości usunąć Ω, odcinając drogę w punkcie S:

$$S \to S', S' \in Q : \text{usunąć z } \Omega \text{ wszystkie punkty poprzedzające S'}$$
$$S \to S', S' \notin Q : cut(S)$$

Ruch punktu docelowego

Reguła rozchodzenia i *reguła drogi* gwarantują, że droga podąża za ruchem punktu T. Jednak, podążając za celem, długość drogi może stać się pod-optymalna (ang. *sub-optimal*). Istnieją dwie metody na poradzenie sobie z tym problemem:

Ograniczenie długości drogi

W przykładzie zastosowania, w którym robot podąża wyznaczoną drogą do celu, robot nie musi znać całej drogi, jeśli odległość do celu jest znaczna. Musi znać jedynie l kolejnych punktów drogi, aby zaplanować ruch bez szarpnięć (ang. *jerk-free motion*). W takim przypadku maksymalna długość drogi może zostać ograniczona do narzucone wartości l, następująco zmieniając *regułę drogi*:

$$path(P) \coloneqq active(P) \wedge cursp\_neighbor(P) \wedge (P \neq S) \wedge (|Q| < l)$$

Szum odcinający

Droga może stać się pod-optymalnie długą zarówno z powodu podążania za ruchomym celem, jak i ze względu na ruch przeszkody. Zamiast skomplikowanych algorytmów rozpoznających taką drogę i naprawiających ją, można użyć prostego rozwiązania: szumu. Poprzez zastosowanie *szumu odcinającego* losowym punktom $P \in \Psi$ aplikowana jest *reguła*

odcięcia. Punkt P wybierany jest z bardzo niskim prawdopodobieństwem α, np. α=0.5%. Dzięki temu droga odcinana jest w losowych odstępach czasu i z punktu odcięcia może optymalnie urosnąć, uwzględniając aktualne pozycje celu i przeszkód.

Ruch przeszkody

Ruch przeszkody może spowodować dwa problemy:

- Przeszkoda przesunęła się na istniejącą drogę, powodując $\Omega \cap \Theta \neq \varnothing$. Rozwiązanie tego problemu następuje za pomocą odcięcia drogi we wszystkich punktach $P \in (\Omega \cap \Theta)$

$$\forall P \in (\Omega \cap \Theta) : cut(P)$$

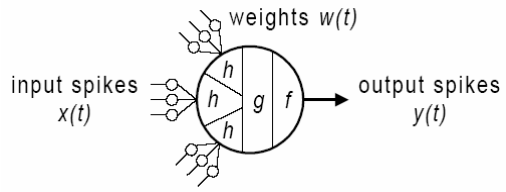
- Długość drogi stała się pod-optymalna. Rozwiązanie za pomocą *ograniczenia długości drogi* lub *szumu odcinającego*.

3. Implementacja neuronowa

Implementację neuronową ułatwia kilka cech zaprezentowanego algorytmu. Przestrzeń robocza jest dyskretna i każdy jej punkt może być reprezentowany poprzez neuron. Równoległość sieci neuronowych może zostać wykorzystana ze względu na lokalny charakter obliczeń i komunikacji. Stan punktu w przestrzeni lub neuronu zależy jedynie od swojego stanu przeszłego i od stanów bezpośrednich sąsiadów. Neuron wymaga wymieniania jedynie dyskretnej informacji: aktywny, nieaktywny. Jednakże, neuron powinien znać pojęcie czasu, aby zapamiętać czy został aktywowany jako część drogi czy jako składowa fali. Z tego względu do implementacji wybrane zostały przejawiające dynamiczne zachowanie neurony impulsowe.

a. Neurony impulsowe

Neurony impulsowe to skomplikowane neurony typu zintegruj-i-wystrzel (ang. *integrate-and-fire*). Ogólny model neuronu impulsowego przedstawia rysunek 3:



Rysunek 3 Ogólny model neuronu impulsowego

Neurony komunikują się wyłącznie za pomocą impulsów. Przychodzące impulsy $x(t) \in \{0,1\}$ są odpowiednio ważone i wywołują różny w czasie potencjał w synapsie, który zmienia się pod wpływem impulsowej funkcji odpowiedzi (ang. *impulse response function*) *h(t)* w skali czasu znacznie

dłuższej od pojedynczego impulsu. Neuron impulsowy może posiadać kilka synaps z różnymi impulsowymi funkcjami odpowiedzi. Bieg czasu funkcji *h(t)* modeluje potencjał postsynaptyczny i wyposaża neuron w pamięć krótkotrwałą. Funkcja ta może charakteryzować się ostrym wzrostem wartości, po którym następuje gwałtowne zmalenie, lub może to być wyłącznie malejąca funkcja. Funkcja łącząca *g* przechowuje różne potencjały za pomocą dodawania, odejmowania i/lub mnożenia, przez co uzyskuje się potencjał membrany. Funkcja wyjściowa *f* porównuje potencjał membrany z progiem odpalenia (ang. *firing treshold*), aby ustalić, czy wysłać impuls $y(t) \in \{0,1\}$ do połączonych neuronów, czy nie.

Wagi połączenia mogą zostać zmienione stosownie do odpowiednich reguł uczących. Na potrzeby tej implementacji wybrana została reguła Hebba, w której wzmacniane są wagi pomiędzy aktywnymi neuronami. Neuron zwany jest aktywnym-do-nauki-celu (ang. *target-learning-active*), gdy potencjał membrany jest większy niż wartość progowa uczenia. Neuron zwany jest aktywnym-do-nauki-źródła (ang. *source-learning-active*), gdy emituje impuls. Wagi są zmniejszane, gdy tylko cel, lub gdy tylko źródło jest aktywne-do-nauki. Dzięki temu można nauczyć bezpośredniego połączenia z neuronu źródłowego do docelowego, gdy oba neurony symultanicznie przechodzą w stan aktywny-do-nauki.

Do symulacji użyty został symulator sieci impulsowych z graficznym interfejsem użytkownika, SimSPiNN, który został stworzony, ponieważ dostępne neuro-symulatory nie mają dobrego wsparcia do symulacji impulsowych sieci neuronowych. Do przeprowadzania symulacji bardzo dużych sieci (N>100 000) w czasie rzeczywistym powstaje obecnie symulator sprzętowy, zwany NESPINN.

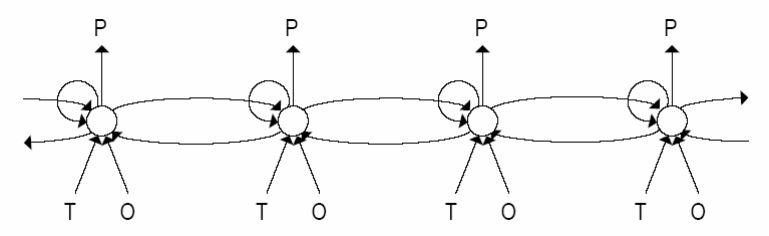
b. Implementacja neuronowa algorytmu radarowego

Do implementacji algorytmu radarowego z użyciem neuronów impulsowych potrzebne są jedynie dwie przetwarzające warstwy:

- warstwa rozsyłająca, implementujące regułę rozsyłania i
- warstwa drogi implementująca regułę drogi.

Każdemu punktowi przestrzeni odpowiada jeden neuron w warstwie rozsyłającej i jeden w warstwie drogi.

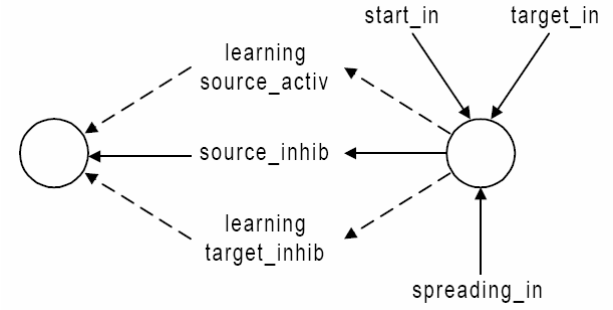
Rysunek 4 przedstawia poziome odcięcie poprzez warstwę rozsyłającą:



Rysunek 4 Poziome odcięcie poprzez warstwę rozsyłającą

Każdy neuron jest połączony ze swoimi poziomymi i pionowymi sąsiadami poprzez połączenie wzmacniające i ze samym sobą poprzez połączenie hamujące. Wszystkie połączenia charakteryzują się opóźnieniem – 1 kroku w czasie. Kiedy neuron jest aktywny, np. gdy wysyła impuls, przesyła ten impuls do swoich poziomych i pionowych sąsiadów. Tak więc, w następnym kroku czasowym, każdy sąsiedni neuron stanie się aktywny, jeśli obecnie nie jest hamowany przez samego siebie (ang. *self-inhibited*). W ten sposób czoło fali aktywnych neuronów rozprzestrzenia się po warstwie rozsyłającej. Samohamowanie neuronów jest wymagane do zapobieżenia cofaniu się fali.

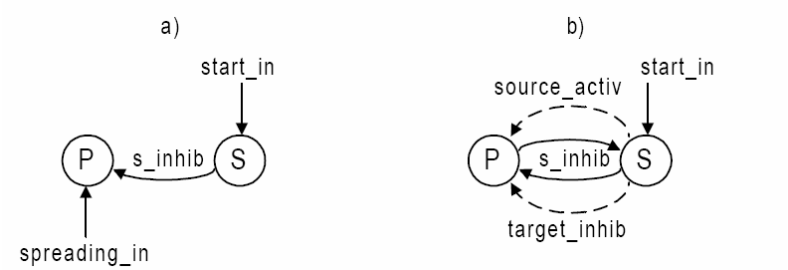
Każdy neuron rozsyłający posiada hamujący punkt-wejścia-przeszkody (ang. *obstacle-point-input*) (O), przez który jest hamowany, gdy neuron jest częścią przeszkody, oraz wzmacniający punkt-wejścia-celu (ang. *target-point-input*) (T), przez który może zostać okresowy aktywowany do stworzenia fali. Każdy neuron warstwy rozsyłającej ma odpowiadający mu neuron w warstwie drogi, do którego jest podłączony połączeniem wzmacniającym (P).



Rysunek 5 Wejścia i połączenia do lewego sąsiada neuronu w warstwie drogi

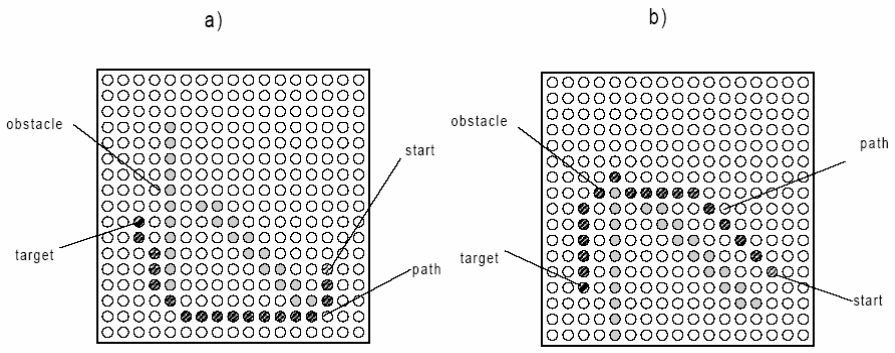
Każdy neuron w warstwie drogi jest połączony ze wszystkimi swoimi ośmioma sąsiadami za pomocą trzech połączeń – jednym stałym i dwoma do uczenia. Ponad to każdy neuron posiada trzy wejścia. Rysunek pokazuje wejścia i połączenia z lewymi sąsiadami neuronu warstwy drogi.

Rysunek 6 wyjaśnia działanie warstwy drogi:



Rysunek 6 Droga z neuronu S do P **a)** przed, **b)** po wyuczeniu połączeń

Ukierunkowane połączenie ma zostać wyuczone od punktu startowego S do pierwszego punktu P drogi. Aby nauczyć się tego połączenia, odpowiedni neuron S warstwy drogi musi być w stanie aktywnym-do-nauki-źródła, a równoległy mu neuron P musi być w stanie aktywnym-do-nauki-celu. Rysunek a przedstawia sytuację początkową, gdy połączenie nie zostało jeszcze wyuczone. Neuron S staje się aktywne-do-nauki-źródła poprzez ciągły sygnał podawany na jego wejście start_in. Aktywacja źródła prowadzi do aktywacji impulsu, więc neuron stale wytwarza impulsy, przekazując hamowanie nauki-źródła przez stałe połączenie s_inhibit do neuronu P. Neuron P nie wytwarza impulsu, więc połączenie s_inhibit z neuronu P do neuronu S na razie nie wywołuje efektu. Czoło fali w warstwie rozsyłającej osiągnęło neuron P, tak więc neuron P dostaje impuls od odpowiadającego mu neuronowi warstwy rozsyłającej poprzez wejście spreading_in. Impuls ten powoduje ustawienie neuronu P w stan aktywny-do-nauki-celu. Z tego względu z neuronu S do neuronu P tworzone są połączenia uczące source_activ i target_inhib. Neuron P poprzez połączenie source_activ otrzymuje od bezustannie wysyłającego impulsy neuronu S aktywację do-nauki-źródła. Poprzez target_inhib zapobiegana jest dalsza aktywacja do-nauki-celu neuronu P. Poprzez połączenie stałe s_inhib z neuronu P do neuronu S usuwany jest stan aktywny-do-nauki-źródła neuronu S. Tak więc teraz neuron P jest aktywny-do-nauki-źródła i regularnie wytwarza impulsy. Jest to taka sama sytuacja, w jakiej znajdował się neuron S. Jeśli kolejne czoło fali w warstwie rozsyłającej osiągnie sąsiadujący z neuronem P neuron Q, budowana jest droga z P do Q. Wejście target_in używane jest do wstrzymania budowy drogi w punkcie odpowiadającym docelowemu.



Rysunek 7 Przykład dwóch kroków symulacji z ruchomym celem i ruchomą przeszkodą

Rysunek 7 przedstawia przykład dwóch kolejnych kroków symulacji, gdzie pokazane zostały optymalne drogi od źródła do celu. Gdy punkt źródłowy, docelowy, lub przeszkoda poruszają się, dynamicznie budowana jest nowa optymalna droga do celu. Droga jest odcinana i odbudowywana, gdy przeszkoda pojawia się na drodze do celu. Gdy droga staje się pod-optymalna poprzez przesunięcie celu lub przeszkody, nowa optymalna droga jest odszukiwana wykorzystując ograniczenie długości drogi i szum odcinający.

4. Podsumowanie

Przedstawiony został nowy algorytm planowania drogi – „radar path planner", zdolny do wyznaczania drogi w nieznanym, częściowo znanym i zmiennym środowisku. Droga powstaje w następujący sposób: w stałych odstępach czasu cel wysyła falę przebiegającą przestrzeń ze stałą prędkością. To pole sąsiadujące z polem startowym, do którego najpierw dotrze fala, jest częścią optymalnej ścieżki. Wyznaczone pole sąsiednie staje się polem startowym i procedura powtarza się aż do osiągnięcia celu.

Gdy porusza się punkt startowy, cel, lub przeszkoda – nowa optymalna droga jest dynamicznie budowana. W razie, gdy na drodze pojawia się przeszkoda, droga jest obcinana i odbudowywana. Gdy droga, pod wpływem przesunięcia się celu lub przeszkody, okazuje się pod-optymalna, nowa optymalna droga odnajdywana jest poprzez skrócenie długości wyznaczonej drogi i usunięcie szumu.

Ponadto przedstawiona została efektywna implementacja algorytmu z użyciem neuronów impulsowych. Właściwości algorytmu ułatwiają implementację wykorzystującą neurony. Ze względu na dynamiczne zachowanie wybrano neurony impulsowe.