Zadanie planowania ścieżki robota mobilnego

1. *Klasyczne* metody poszukiwania najkrótszej ścieżki
2. Inne metody

Omawiane metody planowania:

- Metoda pól potencjałowych
- Algorytm Dijkstry - poszukiwanie najkrótszej ścieżki w grafie
- Algorytm dyfuzyjny - poszukiwania najkrótszej ścieżki na mapie rastrowej
- Algorytm A*
- Planowanie lokalne

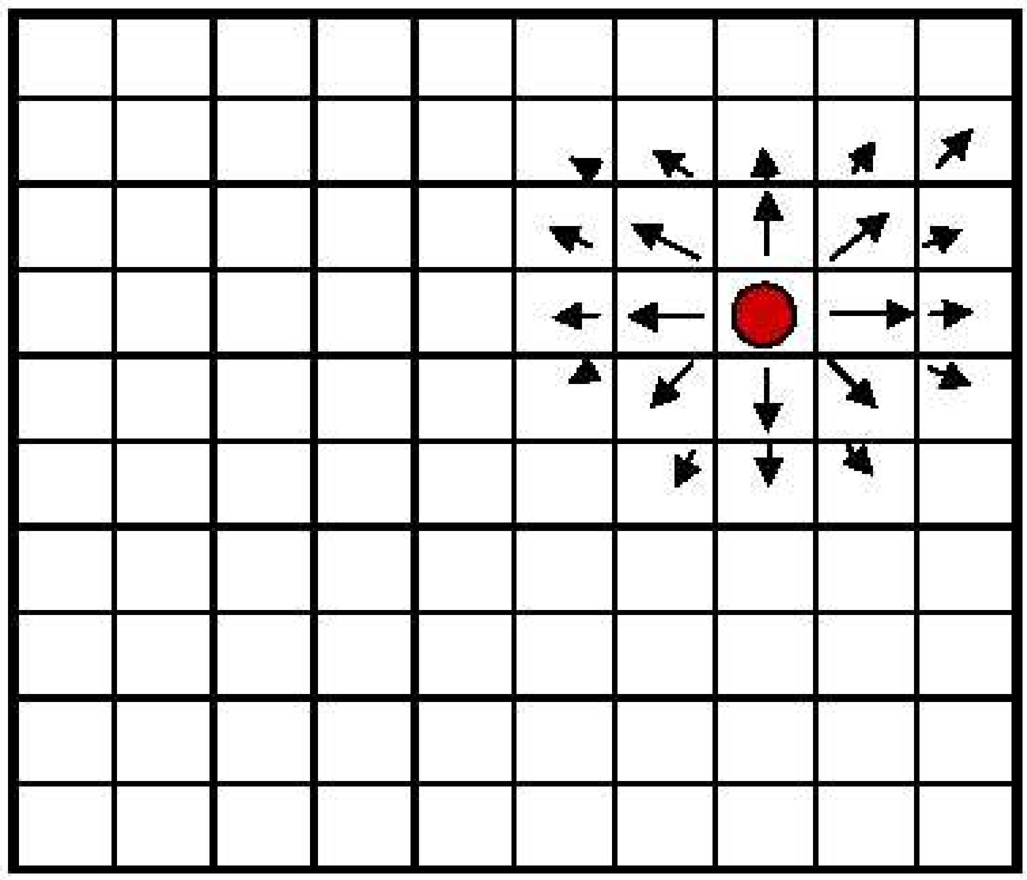
- Szkic algorytmów genetycznych
- Szkic algorytmów mrówkowych
- Logika rozmyta w robotyce

Metoda pól potencjałowych

- Pole potencjałowe: pole wektorowe
  - w reprezentacji komputerowej - tablica wektorów
- Wektor tego pola:
  - długość wektora reprezentuje moduł siły działającej na robota
  - kierunek i zwrot wektora reprezentuje kierunek i zwrot tej siły
- Przestrzeń pola wektorowego - przestrzeń dwuwymiarowa (mapa z lotu ptaka)

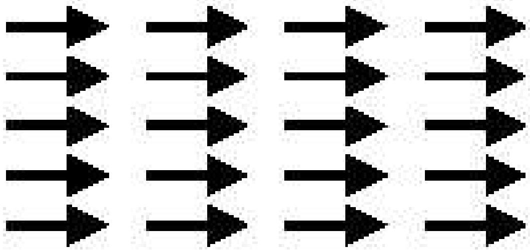
Metoda pól potencjałowych

- Mapa - komórkowa (rastrowa)
- Obiekty na mapie generują pole sił

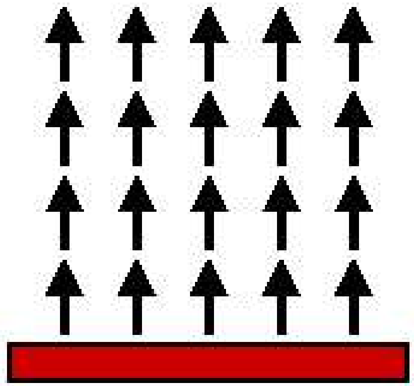


Podstawowe typy pól potencjałowych

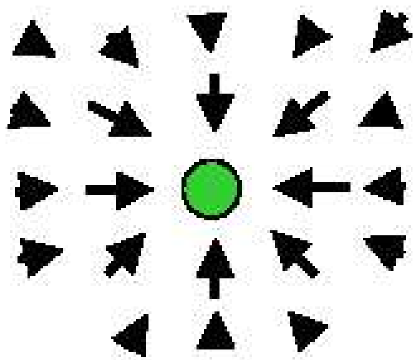
Jednorodne



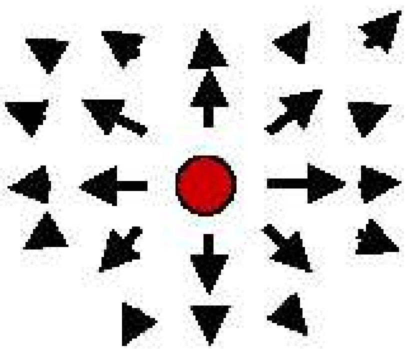
Prostopadłe



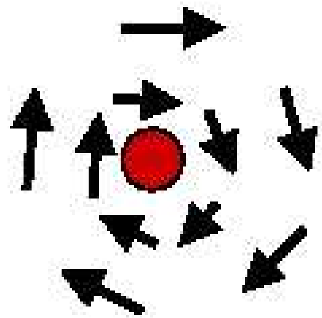
Przyciągające



Odpychające

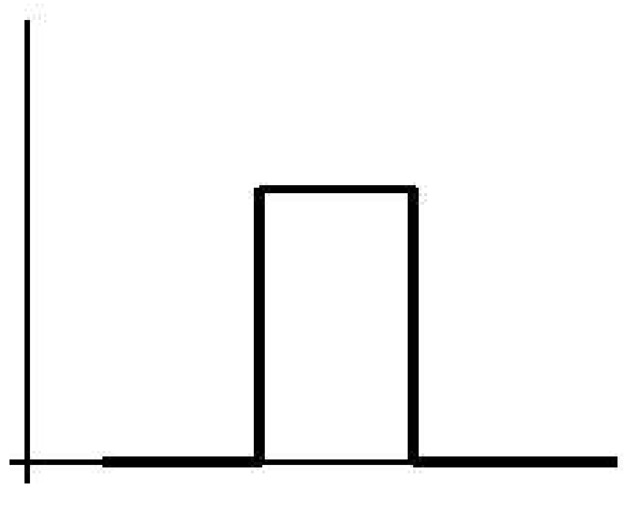


Styczne

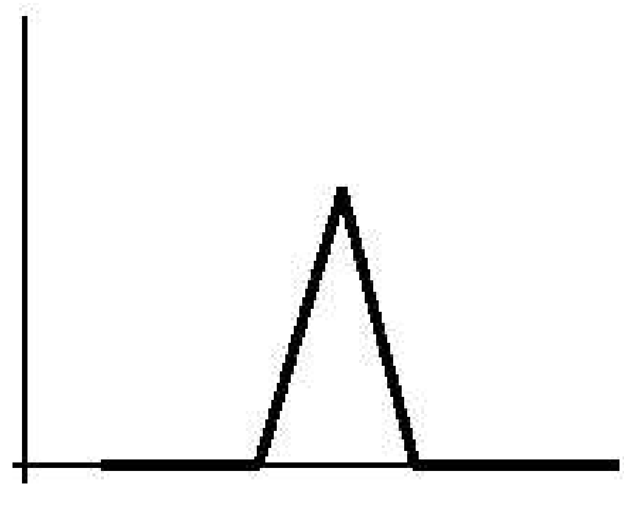


Zależność modułu siły od odległości

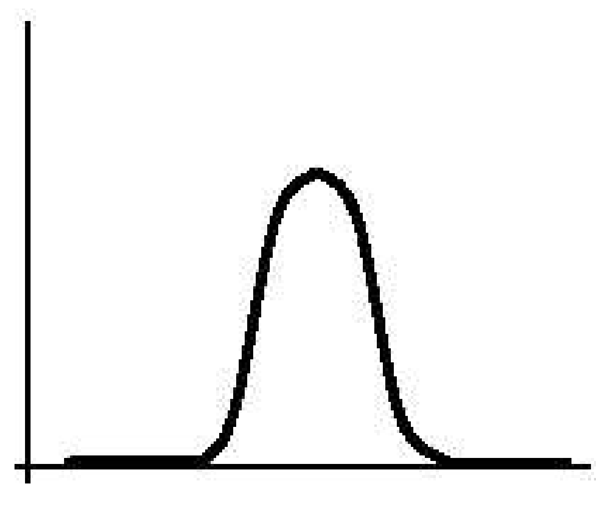
a zarazem zależność prędkości robota od odległości od obiektu generującego pole siłowe



stała



liniowa

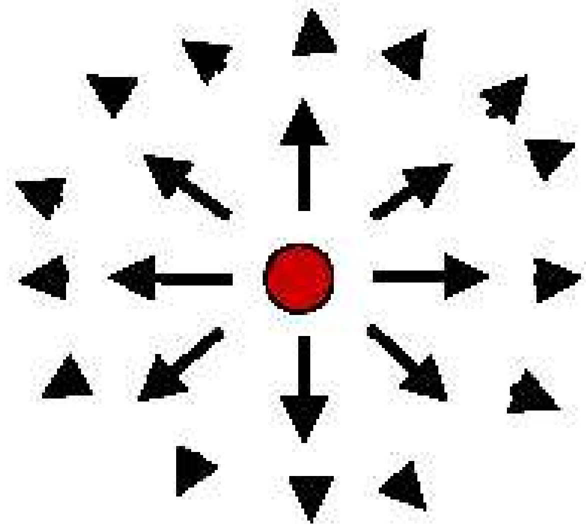


ekspotencjalna

Przykład opisu pola odpychającego

z liniową zależnością siły od odległości

$$V = \begin{cases} \dfrac{(D-d)}{D} & \text{dla } d \le D \\ 0 & \text{dla } d > D \end{cases}$$



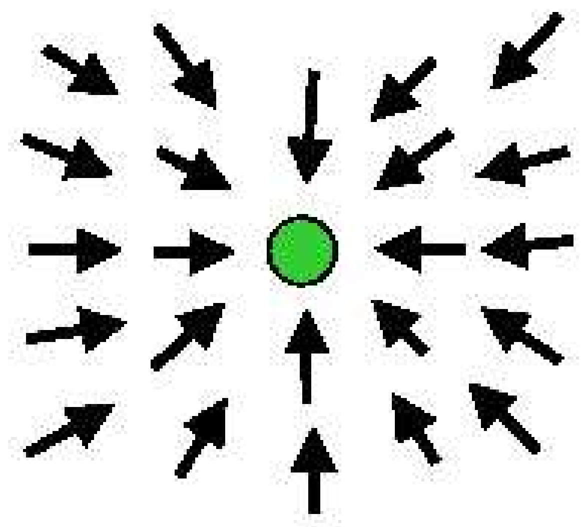
gdzie D jest maksymalną odległością, przy której pole oddziałuje na robota

Obliczanie pola potencjałowego

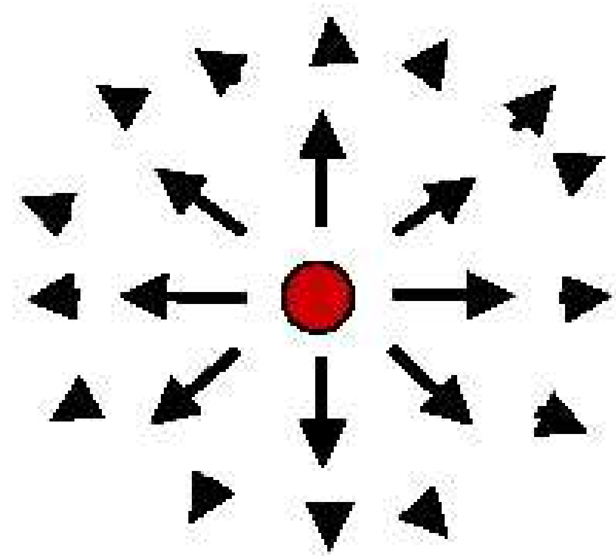
- Nie trzeba wyznaczać całości pola wektorowego, bo ograniczamy się w obliczeniach tylko do bezpośredniego otoczenia robota
- Robot na podstawie analizy działających na niego pól potencjałowych oblicza sumaryczny wektor sił

Przykładowe pola do różnych zadań

- Przyciąganie generowane przez cel robota
  - kierunek i zwrot - do celu
  - wartość - stała



- Odpychanie, aby uniknąć kolizji z przeszkodą (nieruchomą):
  - kierunek i zwrot od przeszkody
  - wartość - maleje eksponencjalnie wraz z odległością
  - wewnątrz przeszkody - wartość równa nieskończoności

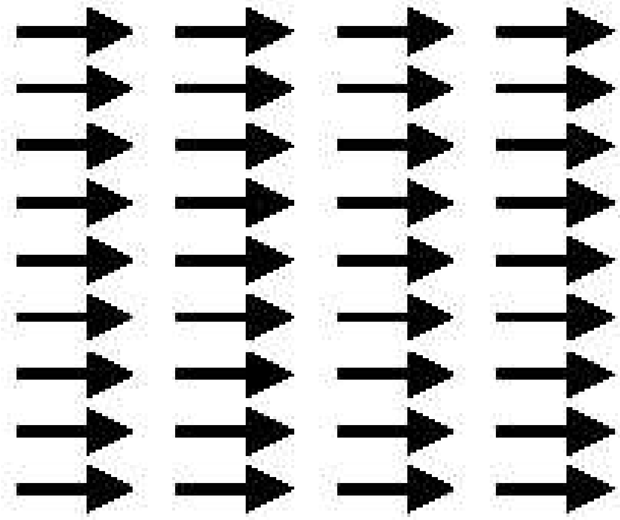


Następne przykładowe pola

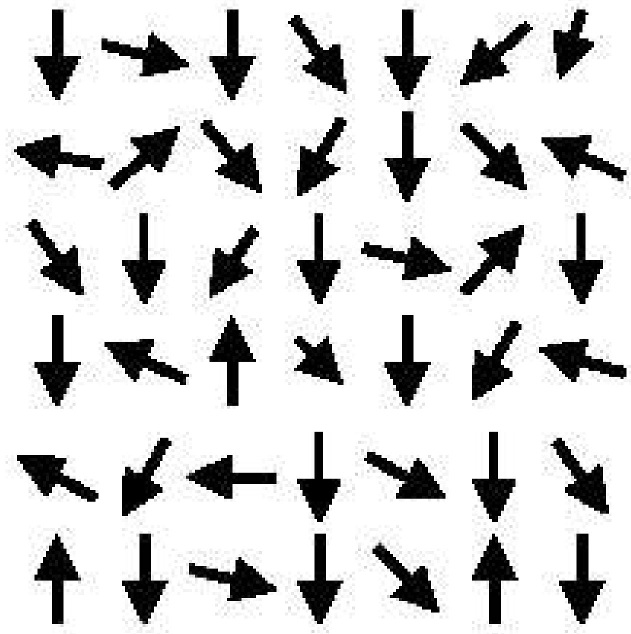
- Pozostań na środku ścieżki (korytarza)
  - kierunek prostopadły do kierunku ścieżki
  - zwrot do środka ścieżki
  - wartość:
    - duża i stała poza ścieżką
    - mniejsza na ścieżce
    - maleje liniowo w miarę zbliżania się do środka

Następne przykładowe pola

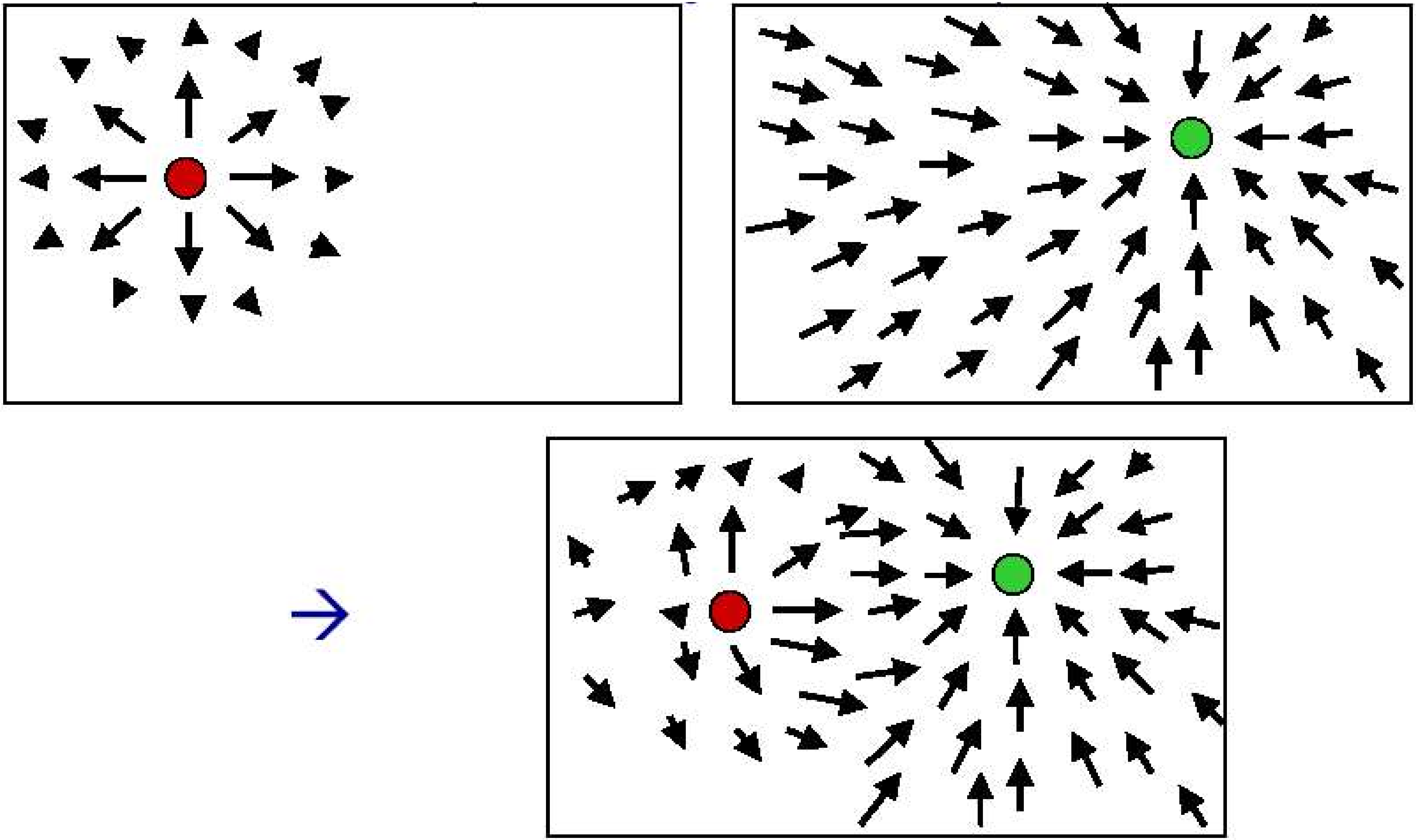
- Ruch na wprost w określonym kierunku:
  - kierunek i zwrot - stały
  - wartość - stała



- Ruch losowy:
  - kierunek i zwrot - przypadkowy
  - wartość - stała



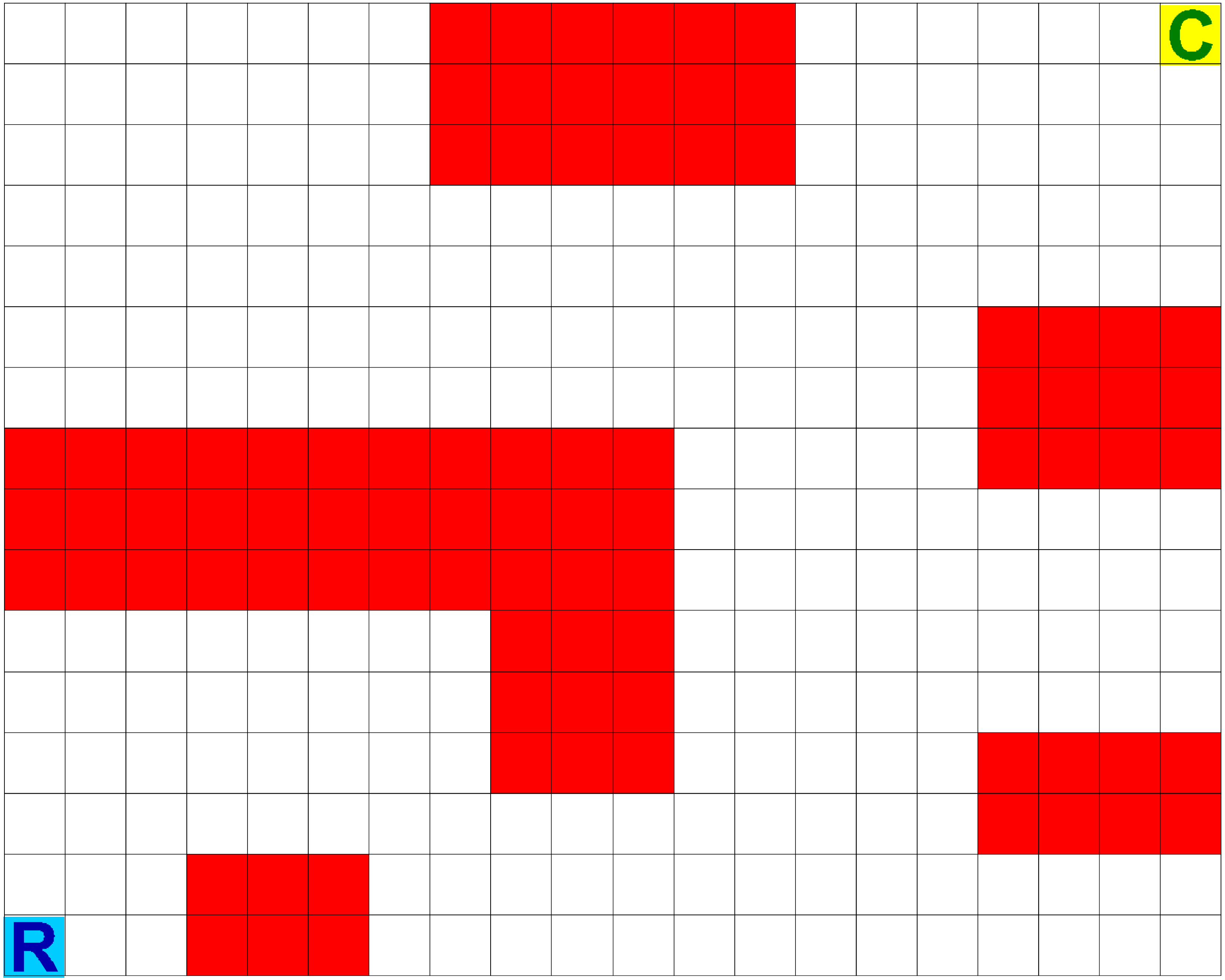
Sumaryczne pole potencjałowe



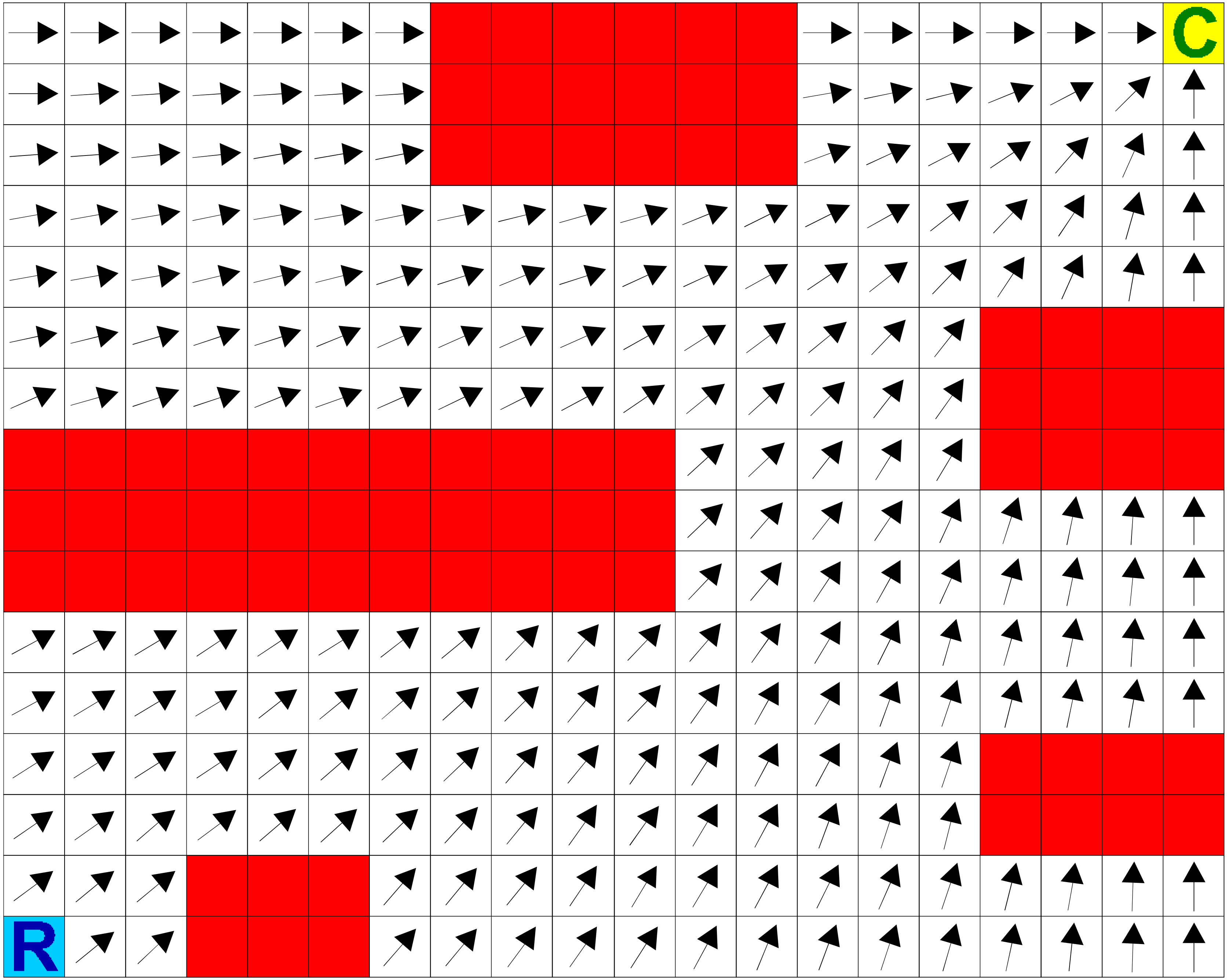
Planowanie ścieżki

- Położenie celu robota: generuje przyciągające pole centralne o charakterze stałym (niezależne od odległości)
- Przeszkody: pola odpychające o ograniczonym zasięgu działania wokół przeszkód, siła odpychająca maleje wraz ze wzrostem odległości od przeszkody
- Efekt - robot porusza się do celu po gradiencie sumarycznego pola potencjałowego i omija przeszkody

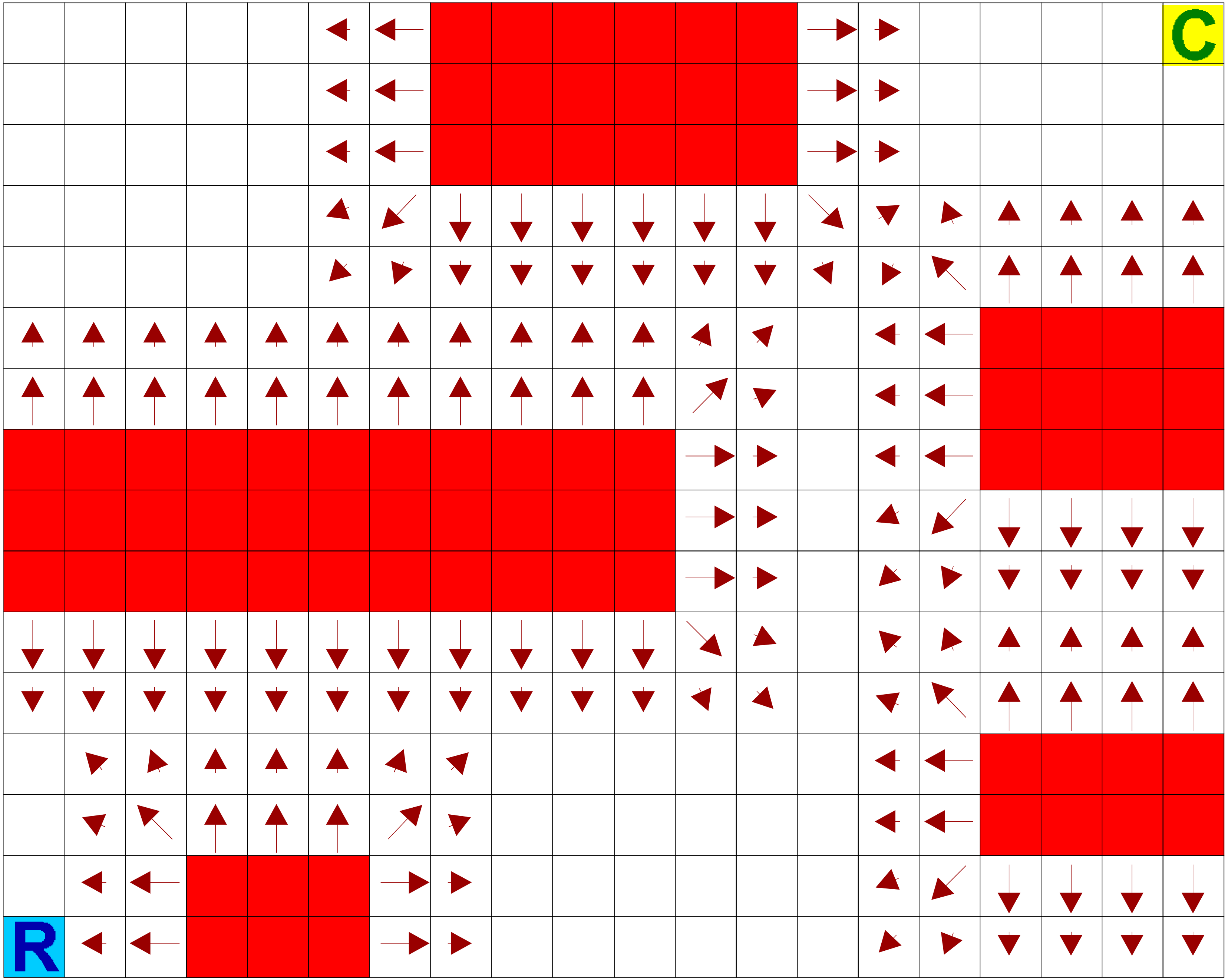
Planowanie ścieżki - mapa



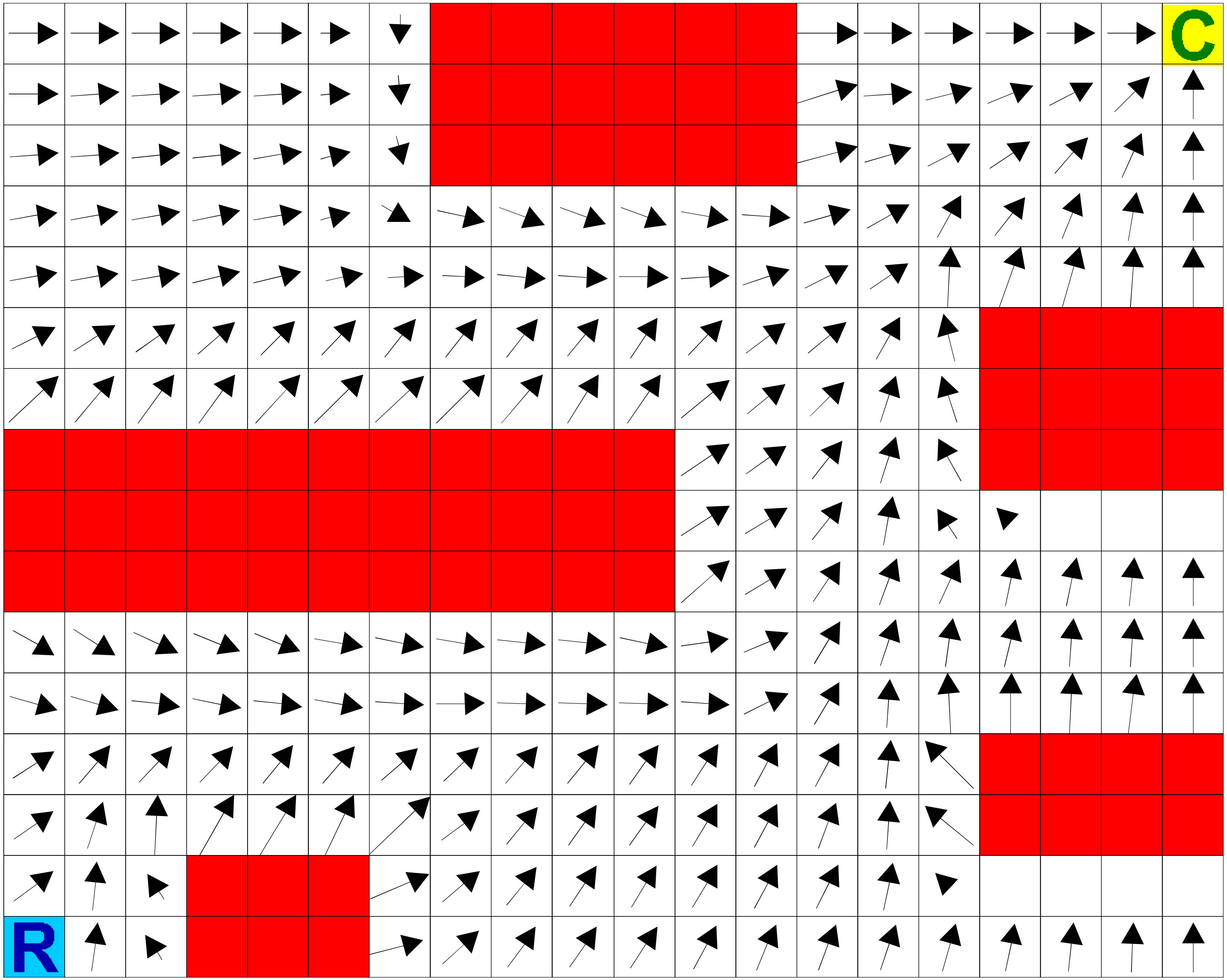
Planowanie - przyciąganie do celu (1)



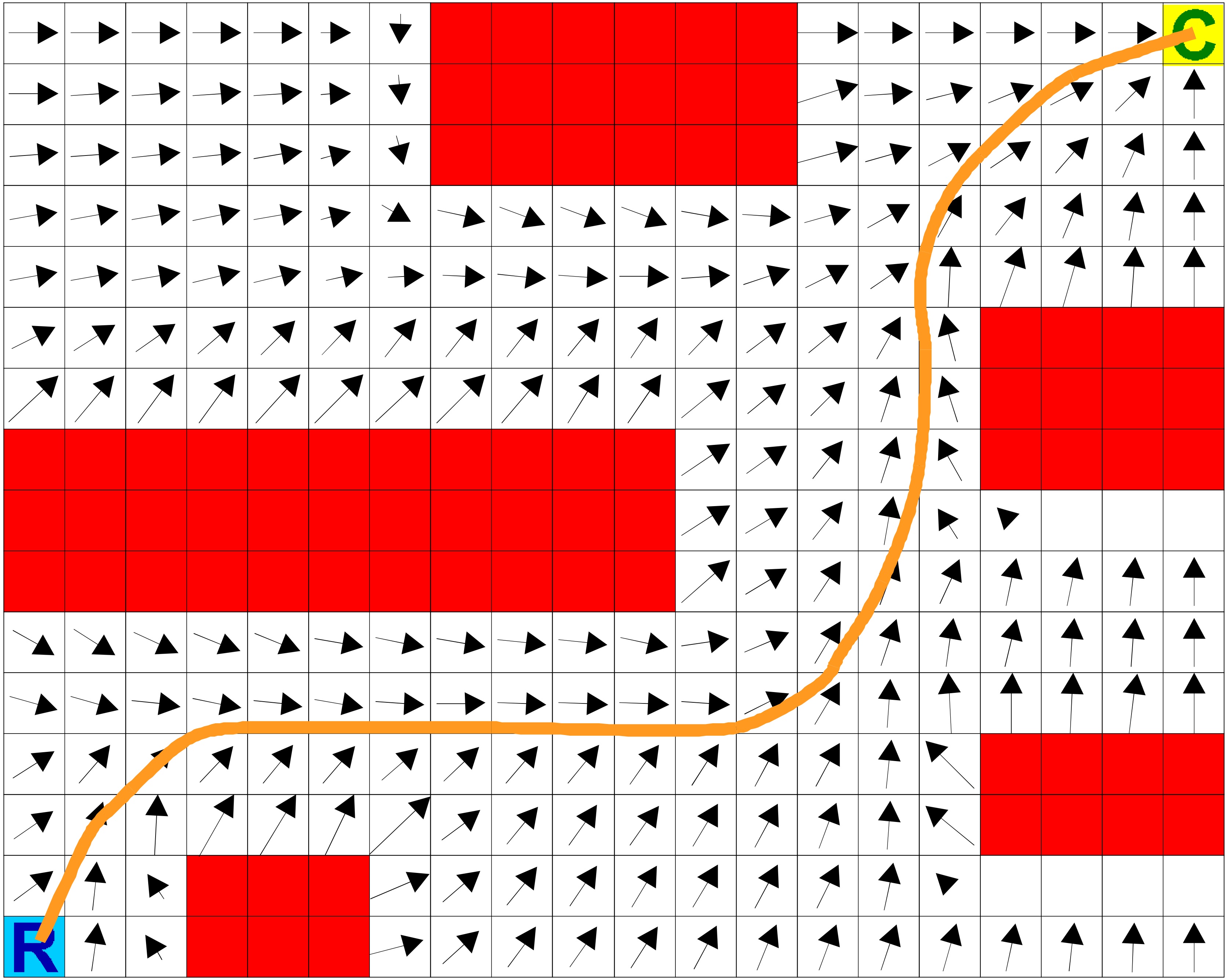
Planowanie - odpychanie od przeszkód (2)



Planowanie - pole sumaryczne (1+2)

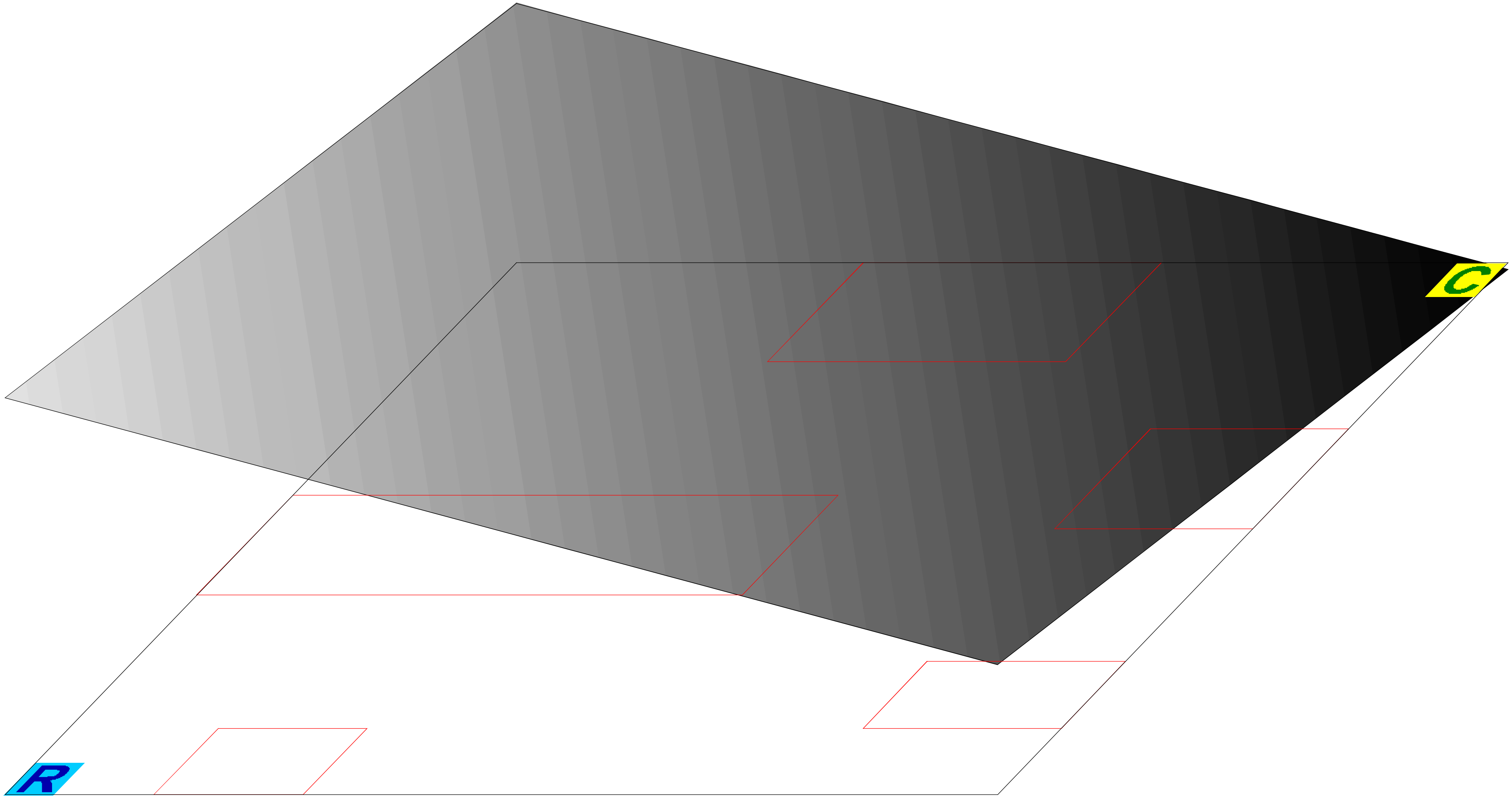


Zaplanowana ścieżka robota



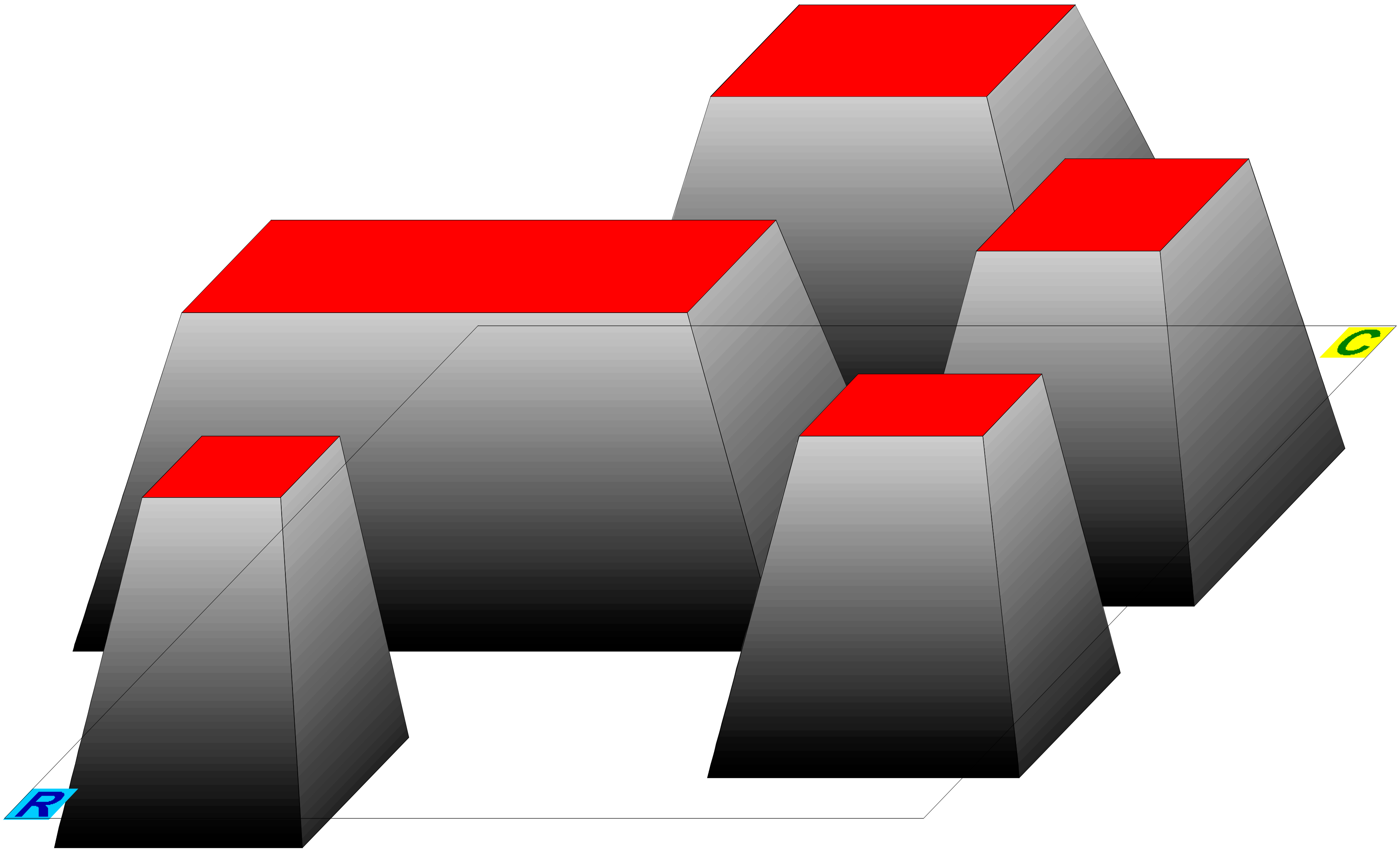
Inna graficzna reprezentacja metody

- Powierzchnia reprezentująca przyciąganie do celu

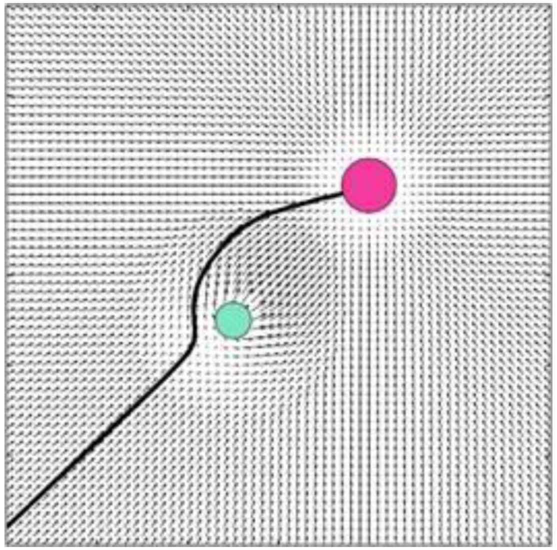


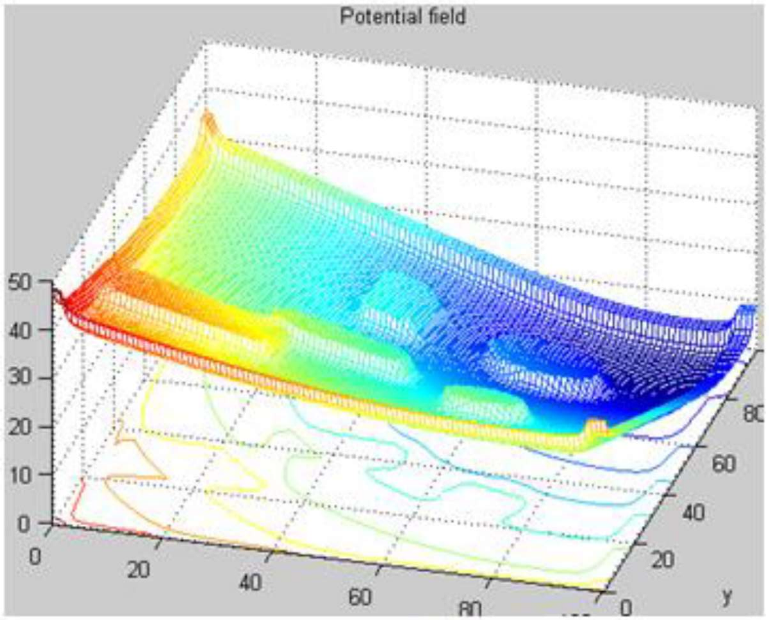
Powierzchnie

- Odpychanie przeszkód

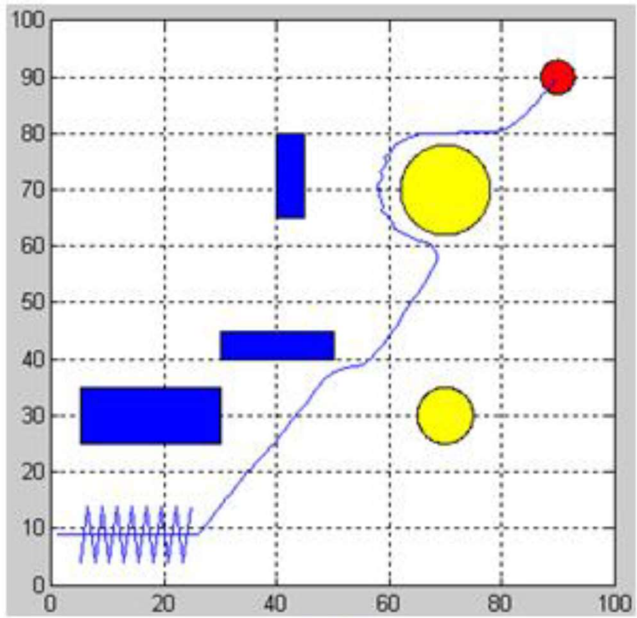


Powierzchnie

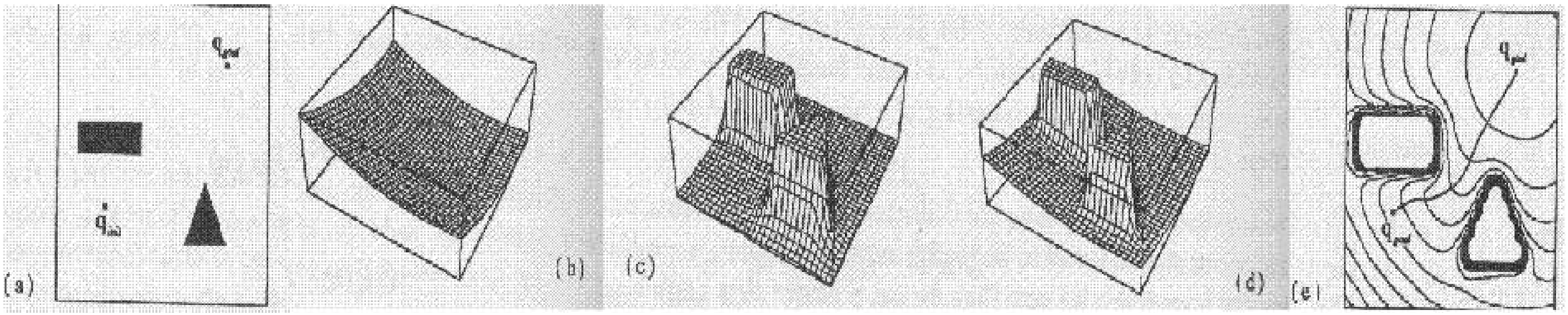




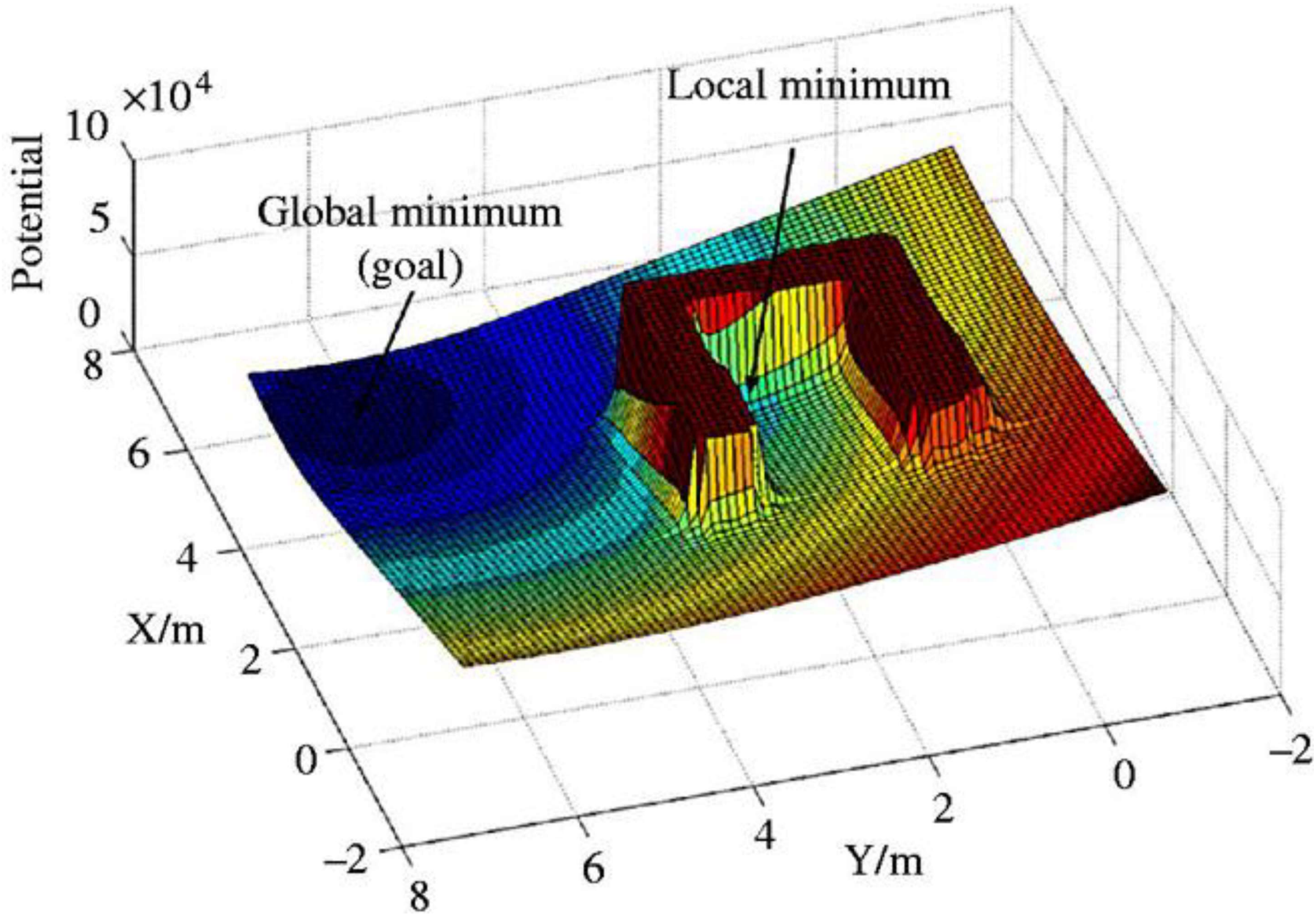
(a)



(b)



Powierzchnie



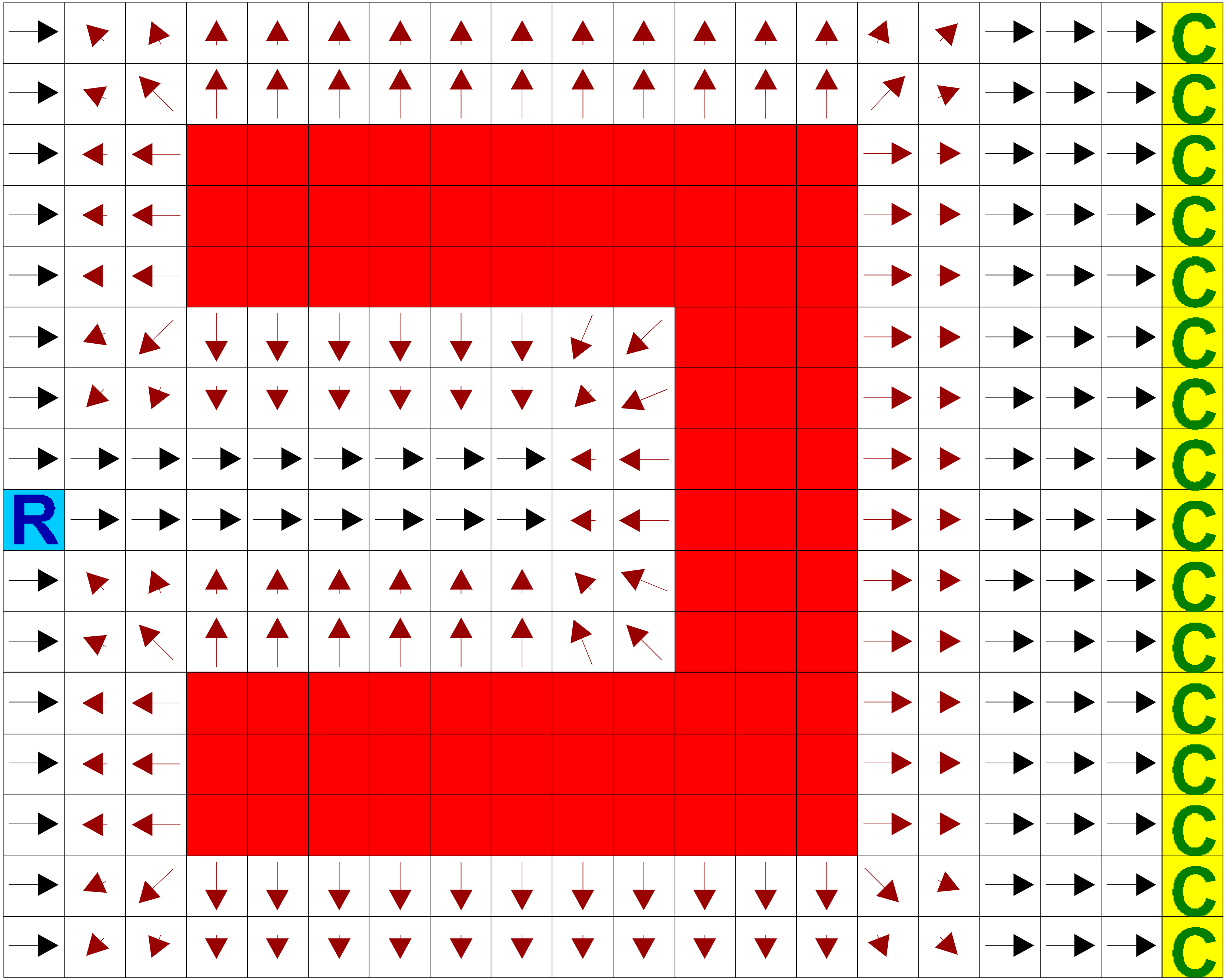
Powierzchnie

- Sumowanie oddziaływań
- Robot porusza się po powierzchni w polu grawitacyjnym
- Robot zachowuje się jak piłka puszczona na powierzchnię będącą sumą oddziaływań

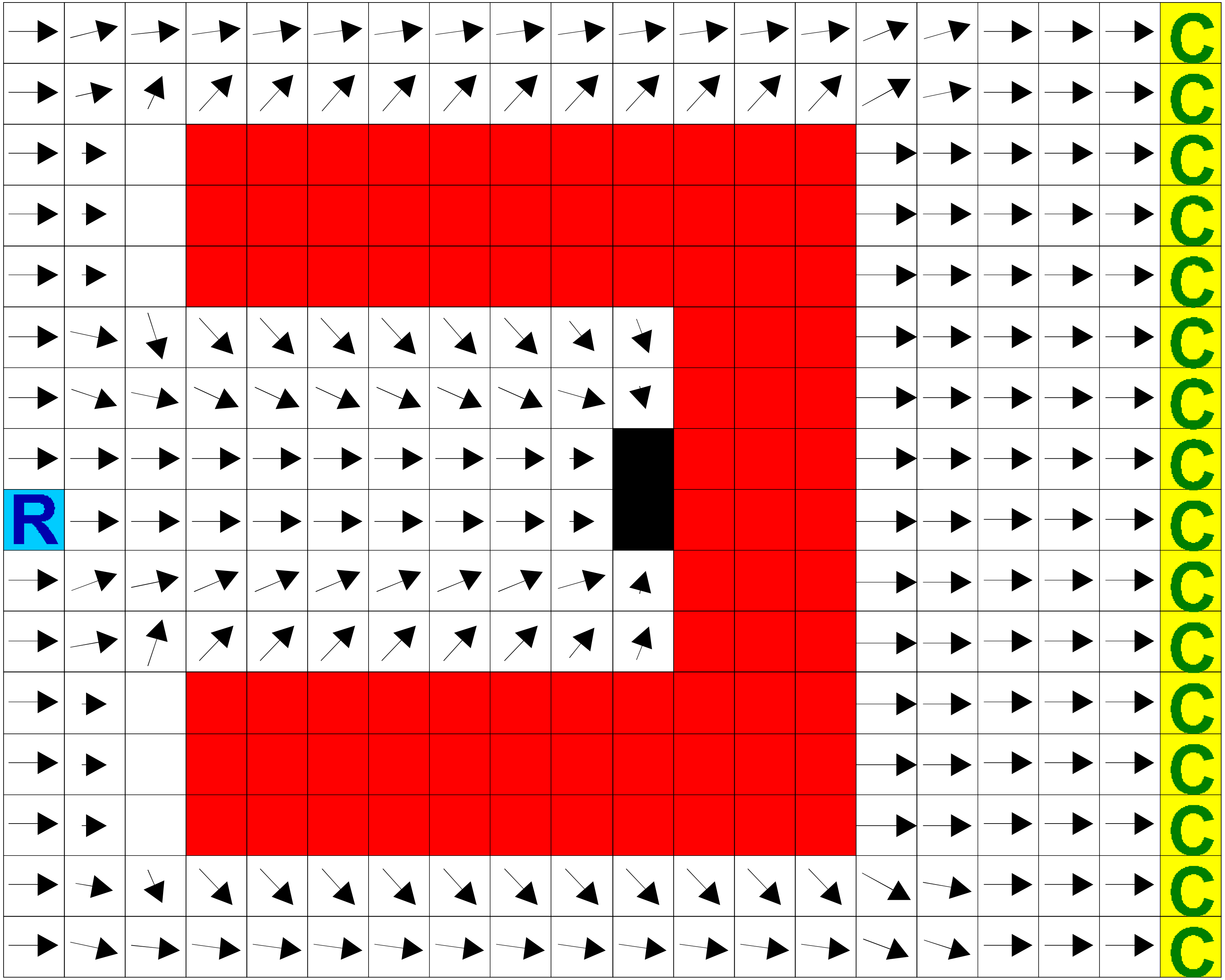
Metoda pól potencjałowych - problemy

- Częstotliwość aktualizacji zmian parametrów ruchu
  - mała częstotliwość - utrata gładkości ścieżek
- Lokalne minima
  - w pewnych warunkach robot może „ugrzęznąć”

Lokalne minima - pola od celu i przeszkód

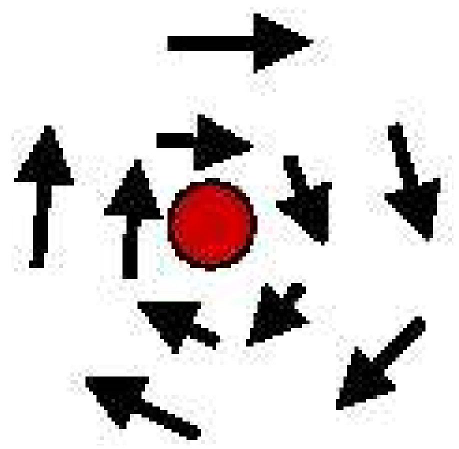


Lokalne minimum - *czarna dziura*



Jak radzimy sobie z minimami lokalnymi?

- Dodatkowy szum losowy - wyrzuca robota z minimum lokalnego
- Dodatkowy system unikania miejsc, które są zagrożone minimami
- Dodatkowe pole styczne otaczające przeszkody



- Modyfikacje matematycznych postaci funkcji opisujących pola sił (bardzo skomplikowane matematycznie)

Metody planowania ścieżki w grafie

Wybrane algorytmy grafowe:

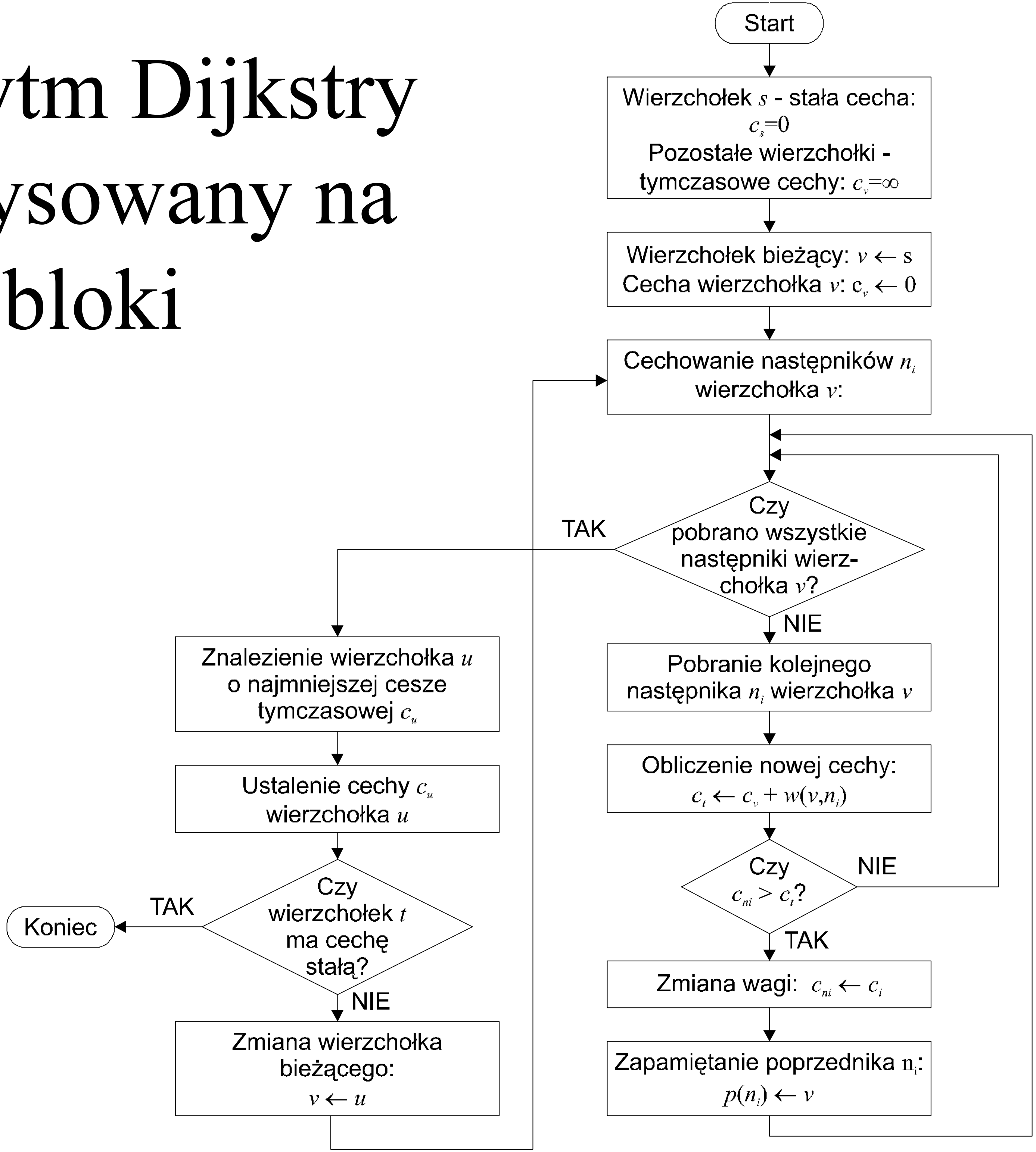
- Algorytm Dijkstry
- Algorytm dyfuzyjny
- Algorytm A*

Algorytm Dijkstry

Graf:

- wierzchołki - reprezentują punkty decyzyjne, wybrane innym algorytmem lub przez operatora
- łuki - reprezentują ścieżki między wierzchołkami
- wyróżniony wierzchołek s - startowy
- wyróżniony wierzchołek t - końcowy
- wszystkie łuki mają nieujemne wagi $w(i, j)$, gdzie i, j - dwa dowolne wierzchołki
- wagi - są proporcjonalne do odległości między punktami reprezentowanymi przez wierzchołki

Algorytm Dijkstry - rozrysowany na bloki



Algorytm Dijkstry

Początkowo:

- wierzchołkom są przypisywane *cechy* - określone wartości
- wierzchołek s ma cechę $c_s=0$
- wszystkie pozostałe wierzchołki mają cechy $c_v=\infty$
- cecha wierzchołka s jest *stała*
- pozostałe wierzchołki mają cechy *tymczasowe*
- wierzchołek s staje się wierzchołkiem *bieżącym (v)*

Algorytm Dijkstry (cechowanie)

1. Cechowane są wszystkie następniki wierzchołka v:
$c_t = c_v + w(v, n)$ (cecha tymcz. = suma cechy v i wagi łuku)

2. Jeżeli $c_t < c_n$ (nowa cecha jest mniejsza)
to:

– cecha c_t staje się nową cechą c_n

– zapamiętujemy poprzednika n: $p(n)=v$

3. Szukany jest wierzchołek u z najmniejszą cechą tymczasową c_u - cecha ta staje się cechą *stałą*

4. Jeżeli wierzchołek t ma cechę stałą - **koniec** cechowania

5. Zmiana bieżącego v na u i idziemy do p.1.

Algorytm Dijkstry (szukanie ścieżki)

- Poszukujemy (i zapamiętujemy) poprzednika wierzchołka t : $p(t)$
- Poszukujemy (i zapamiętujemy) poprzednika poprzednika itd.
  aż dojdziemy do wierzchołka s
- Zapamiętany w ten sposób ciąg wierzchołków odczytany od końca do początku reprezentuje *najkrótszą ścieżkę* od wierzchołka s do t

Przykład na tablicy

Algorytm Dijkstry - przykład

Algorytm dyfuzyjny

- Uproszczenie algorytmu Dijkstry
- Tylko na mapę rastrową
- Łuki - bezpośrednie sąsiedztwo komórek
- Wagi łuków: odległości między środkami komórek
- Jak wcześniej: wierzchołki *s* i *t*

Najważniejsza różnica względem alg. Dijkstry:

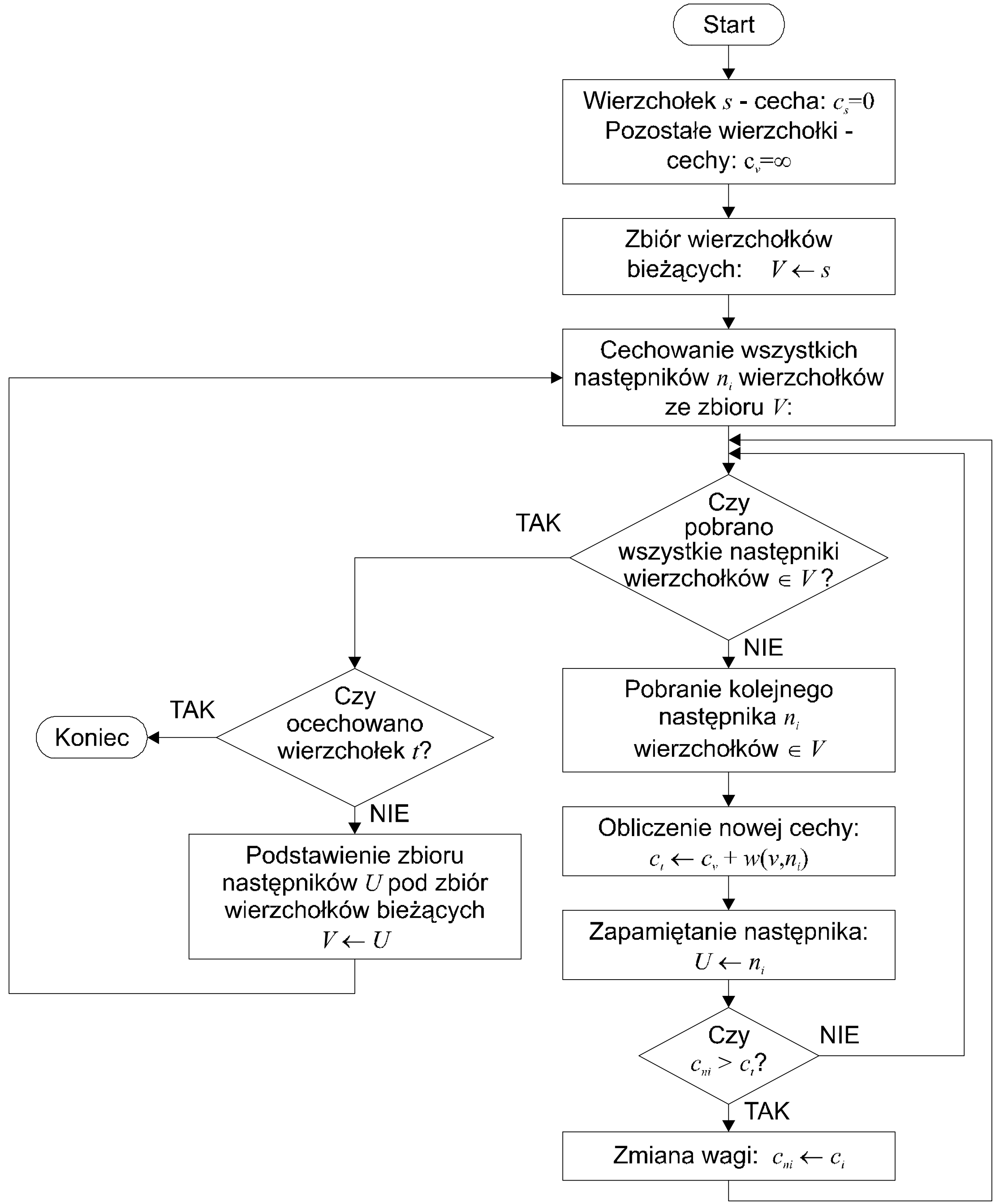
- zamiast wierzchołka bieżącego stosuje się *zbiór wierzchołków bieżących*

Reprezentacja przeszkód

Wierzchołki grafu reprezentujące *zajęte* klatki mapy rastrowej można przedstawić następująco:

1. usunąć je z grafu
2. nadać łukom przylegającym do tych wierzchołków wagi o bardzo dużych wartościach (można nadać wagi równe nieskończoności)
3. nadać wierzchołkom bardzo duże cechy, które są niezmienne

Algorytm dyfuzyjny - rozrysowany na bloki



Algorytm dyfuzyjny

Początkowo:

- wierzchołek *s* ma cechę $c_s=0$
- wszystkie pozostałe wierzchołki mają cechy $c_v=\infty$
- wierzchołek *s* wchodzi do zbioru wierzchołków *bieżących (V)*

Algorytm dyfuzyjny (cechowanie)

1. Cechowane są wszystkie następniki wierzchołków ze zbioru V cechami:

$c_t = c_v + w(v, n)$ (suma cechy v i wartości łuku)

Cechowany następnik zapamiętujemy w zbiorze U

2. Jeżeli $c_t < c_n$ (nowa cecha jest mniejsza)

to:

– cecha c_t staje się nową cechą c_n

– zapamiętujemy poprzednika n: $p(n) = v$

3. Jeżeli wierzchołek t ocechowany - **koniec** cechowania

4. Podstawienie zbioru U za V i skok do p.1.

Algorytm dyfuzyjny (szukanie ścieżki)

- Poszukujemy (i zapamiętujemy) poprzednika wierzchołka t : $p(t)$
- Poszukujemy (i zapamiętujemy) poprzednika poprzednika itd.
  aż dojdziemy do wierzchołka s
- Zapamiętany w ten sposób ciąg wierzchołków odczytany od końca do początku reprezentuje *najkrótszą ścieżkę* od wierzchołka s do t

(dokładnie tak samo, jak dla algorytmu Dijkstry)

Przykład na tablicy

Algorytm dyfuzyjny - nazwa

Nazwa algorytmu pochodzi od kształtu każdego ze zbiorów wierzchołków bieżących V. Zbiory te oddzielają całkowicie cel od klatek nieocechowanych (także od klatki zajętej przez robota) i w poszczególnych iteracjach algorytmu przypominają falę, rozprzestrzeniającą się wokół celu. Efekt ten można zaobserwować na kolejnym slajdzie.

Algorytm dyfuzyjny - przykład

26	25	26	27	28	30	32	34
24	23	24	25	27	29	31	33
22	21	22	24	26	28	30	32
20	19						
18	17		8	6	4	2	0
17	15		9	7	5	3	2
16	14	12	10	8	6	5	4
17	15	13	11	9	8	7	6

Cechy algorytmu dyfuzyjnego

- Brak minimów lokalnych
- Dużo szybszy od alg. Dijkstry
- Szybszy w obliczeniach od pól potencjałowych
- Obecność przeszkód **przyspiesza** działanie algorytmu
- Łatwy do zaprogramowania

- Tylko na mapie rastrowej o stałej wielkości komórki

Algorytm **A*** *(A z gwiazdką)*

- Zbliżony do alg. Dijkstry
- Dodatkowo sposób na redukcję liczby cechowanych wierzchołków w celu zwiększenia szybkości działania
- Ograniczenie cechowania wierzchołków do tych, przez które prowadzą potencjalnie najbardziej obiecujące drogi do celu
- Działa dobrze zarówno w zwykłych grafach, jak i na mapie rastrowej
- Aktualnie jeden z najpopularniejszych algorytmów

A*

- Cechy wierzchołków:
  - G - koszt ścieżki od startu do aktualnej pozycji (jak w alg. Dijkstry)
  - H - szacunkowa długość ścieżki do punktu docelowego
  - F = G + H
- Sprawdza się tylko wierzchołki ocechowane - sąsiadujące z bieżącym i jeszcze nie rozpatrywane
- Dzięki wpływowi funkcji H można zmniejszyć liczbę wierzchołków cechowanych przez ograniczenie przeszukiwań do min. wartości F - algorytm szybszy od a. Dijkstry

A* - funkcja heurystyczna

- H - odległość powinna być niedoszacowana - wtedy mamy gwarancję znalezienia rozwiązania optymalnego
- Wzrost jakości szacowania - szybkość alg. rośnie
- Może to być np. odległość euklidesowa
- Stosuje się też funkcję typu Manhattan (odległość obliczana jako suma odległości w pionie i w poziomie) - dosyć szybko się liczy

A* - poprawianie ścieżek

Często w wyniku działania algorytmu uzyskuje się ścieżki, w których robot często musi zmieniać orientację. Aby temu zapobiec:

- dodaje się pewną wartość do funkcji G w przypadku zmiany orientacji
- stosuje się poprawę ścieżki wykorzystując analizę widoczności między węzłami
- stosuje się wygładzanie na etapie realizacji między węzłami