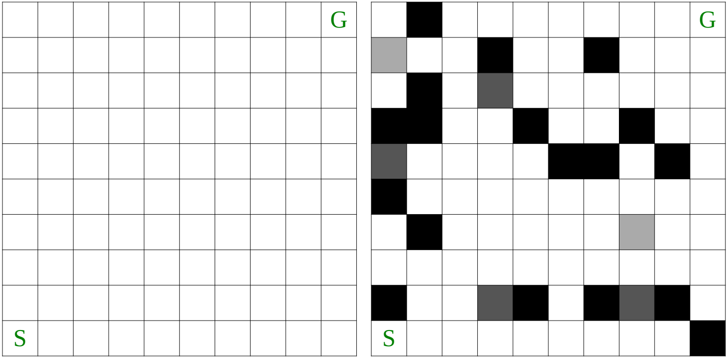
Algorytm Stentz'a
D^*

Przemysław Klęsk
pklesk@wi.zut.edu.pl

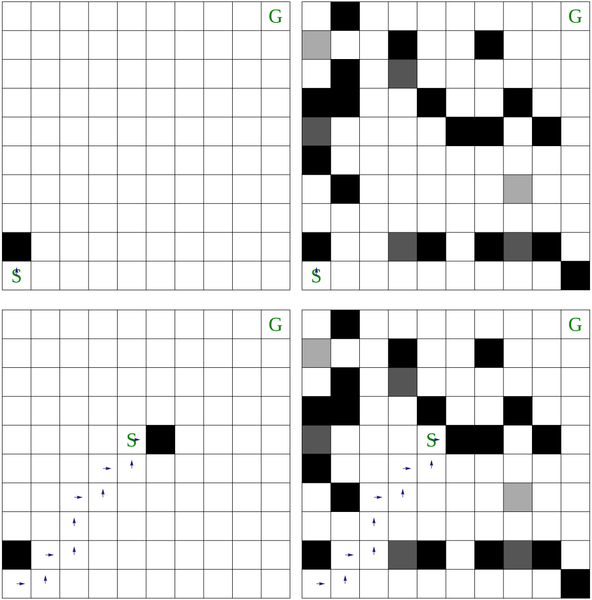
Katedra Metod Sztucznej Inteligencji i Matematyki Stosowanej

Zadanie

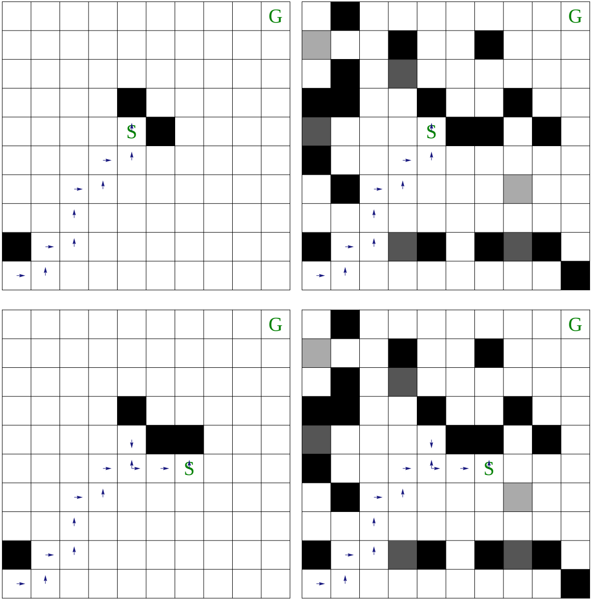
W nieznanym terenie (lub znanym tylko częściowo) należy dojść do celu o podanych współrzędnych. Koszty przejść i przeszkody poznawane są na bieżąco w trakcie przechodzenia ścieżki (przykłady: rzeczywiste roboty mobilne, postaci w grach komputerowych).



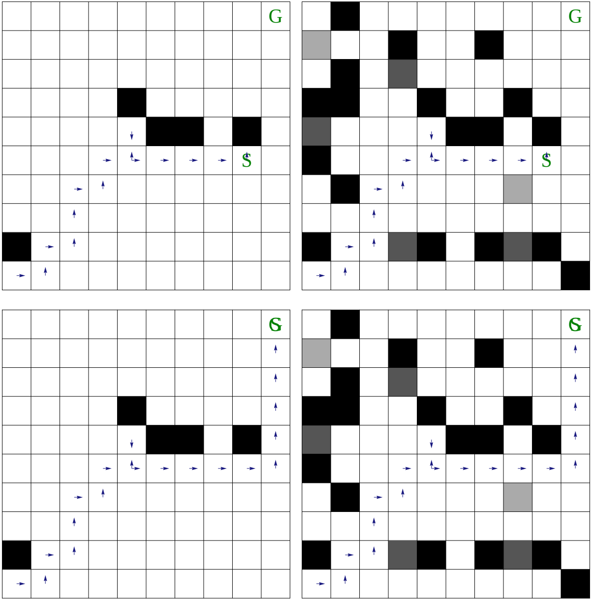
Ilustracja przykładu (1)



Ilustracja przykładu (2)



Ilustracja przykładu (3)



Algorytm Stentz'a (1994) — własności

- Znany również pod nazwą D^*, z intencją rozumienia nazwy jako *dynamic* A^*.
- Mimo nazwy, przeprowadzany w sposób bliższy algorytmowi Dijkstry.
- Algorytm realizuje **optymalne postępowanie** w świetle poznawanych na bieżąco informacji.

Algorytm Stentz'a — własności

- Algorytm działa iteracyjnie — ścieżka jest wyznaczana wielokrotnie.
- Pierwszy przebieg jest *wsteczną* wersją a. Dijkstry — budujemy kolejkę stanów idąc od celu do startu.
- Przy fizycznym przechodzeniu ścieżki i natrafieniu na niezgodność doświadczenia z dotychczasową wiedzą aktualizujemy ścieżkę.
- Stany w kolejce mogą wielokrotnie zmieniać swoje koszty i wielokrotnie trafiać do kolejki.
- Algorytm jest efektywniejszy niż wielokrotne uruchamianie algorytmu Dijkstry „od zera”.

Zbiór akcji i stanów

- Niech A oznacza **zbiór możliwych akcji**. W szczególności dla siatki kwadratów:
$$A = \{\uparrow, \rightarrow, \downarrow, \leftarrow\}.$$
- Niech X oznacza **zbiór stanów**. Dla siatki kwadratów:
$$X = \{x_{ij}\},$$
gdzie i numer wiersza, j numer kolumny.
- Przez $A(x)$ będzie oznaczany zbiór akcji możliwych dla stanu x.

Wykonywanie akcji

- Niech $t(x, a)$ (*transition*) oznacza **funkcję przenoszącą** pewien stan x z użyciem akcji a w nowy stan x', tzn.
$$t(x, a) = x'.$$
  Na przykład dla siatki kwadratów $t(x_{25}, \rightarrow) = x_{26}$.
- Formalnie t jest funkcją $t\colon X \times A \rightarrow X$.

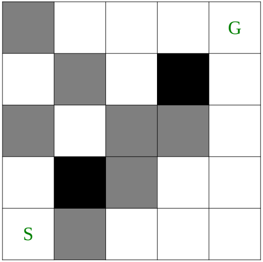
Koszty przejść

- Niech $c(x, a)$ (*costs of transition*) oznacza **funkcję ponoszonego kosztu**, który trzeba ponieść wykonując w x akcję a.
- Formalnie $c\colon X \times A \to \mathbb{R}_+ \cup \{\infty\}$.
- Jeżeli wykonanie w x akcji a jest niemożliwe (przeszkoda lub brzeg mapy), to $c(x, a) = \infty$.
- Taka postać funkcji c pozwala na ogólną reprezentację mapy, gdzie **przejścia w pewien stan, ale z różnych kierunków mogą mieć różne koszty**.
- Jeżeli dla siatki kwadratów chcemy utożsamić koszty przejścia do tego samego stanu x_{ij}, to:

$$c(x_{i-1,j}, \downarrow) = c(x_{i+1,j}, \uparrow) = c(x_{i,j-1}, \rightarrow) = c(x_{i,j+1}, \leftarrow) = \mathrm{map}(i, j).$$

dla wszystkich i, j.

Gdy koszty przejść utożsamione z mapą



$$\mathrm{map} = \begin{pmatrix} 2 & 1 & 1 & 1 & 1 \\ 1 & 2 & 1 & \infty & 1 \\ 2 & 1 & 2 & 2 & 1 \\ 1 & \infty & 2 & 1 & 1 \\ 1 & 2 & 1 & 1 & 1 \end{pmatrix}$$

$$c(\cdot,\uparrow) = \begin{pmatrix} \infty & \infty & \infty & \infty & \infty \\ 2 & 1 & 1 & 1 & 1 \\ 1 & 2 & 1 & \infty & 1 \\ 2 & 1 & 2 & 2 & 1 \\ 1 & \infty & 2 & 1 & 1 \end{pmatrix} \quad c(\cdot,\rightarrow) = \begin{pmatrix} 1 & 1 & 1 & 1 & \infty \\ 2 & 1 & \infty & 1 & \infty \\ 1 & 2 & 2 & 1 & \infty \\ \infty & 2 & 1 & 1 & \infty \\ 2 & 1 & 1 & 1 & \infty \end{pmatrix}$$

$$c(\cdot,\downarrow) = \begin{pmatrix} 1 & 2 & 1 & \infty & 1 \\ 2 & 1 & 2 & 2 & 1 \\ 1 & \infty & 2 & 1 & 1 \\ 1 & 2 & 1 & 1 & 1 \\ \infty & \infty & \infty & \infty & \infty \end{pmatrix} \quad c(\cdot,\leftarrow) = \begin{pmatrix} \infty & 2 & 1 & 1 & 1 \\ \infty & 1 & 2 & 1 & \infty \\ \infty & 2 & 1 & 2 & 2 \\ \infty & 1 & \infty & 2 & 1 \\ \infty & 1 & 2 & 1 & 1 \end{pmatrix}$$

Założenie początkowe

- W najbardziej pesymistycznym przypadku zakładamy pełną niewiedzę o mapie, z wyjątkiem informacji o współrzędnych startowych i docelowych.
- Jeżeli przyjmiemy, że nieutrudnione niczym przejście kosztuje 1 jednostkę, w przypadku pełnej niewiedzy, należy nastawić:

$$\forall x, a \qquad c(x, a) = 1.$$

  W szczególności przyjmujemy, że nie znamy również brzegów planszy.

Wielkości używane w algorytmie

- Niech $g_{\text{current}}(x)$ oznacza aktualnie znany koszt przejścia ze stanu x do stanu docelowego G.
- Niech $g_{\text{via}}(x, x')$ oznacza aktualnie zany koszt przejścia ze stanu x do stanu docelowego G, gdy *podróżujemy przez* stan x'.
- Funkcja g_{via} będzie tak naprawdę rozpatrywana tylko dla x, x' będących bezpośrednimi sąsiadami.
- Niech a będzie akcją, taką że $t(x, a) = x'$. Wówczas:

$$g_{\text{via}}(x, x') = c(x, a) + g_{\text{current}}(x'). \tag{1}$$

Wielkości używane w algorytmie

- Niech Q oznaczą **kolejkę stanów** przechowywaną w algorytmie (analogicznie do algorytmów Dijkstry i A^*).
- Niech V oznaczą **mapę odwiedzonych stanów**.
- Niech $g_{\text{best}}(x)$ oznacza najlepszą (najmniejszą) znaną wartość kosztu dla stanu x podczas jego historii życia w kolejce Q. Wiadomo, że:

$$g_{\text{best}}(x) \leqslant g_{\text{current}}(x). \tag{2}$$

- Kolejka Q jest posortowana według g_{best}.

Tymczasowy plan

- Niech $p(x)$ oznacza **planowaną akcję** przyporządkowaną obecnie do wykonania w stanie x.
- Algorytm będzie wielokrotnie wyznaczał **tymczasowy plan**, czyli pewien ciąg akcji
$$p_1, p_2, \ldots, p_k, p_{k+1}, \ldots,$$
które pozwalają przeprowadzić aktualny stan początkowy w stan docelowy wg obecnej znajomości kosztów przejść c. Tym samym, określona zostanie ścieżka stanów
$$x_1, x_2, \ldots, x_k, x_{k+1}, \ldots,$$
taka że
$$x_{k+1} = t(x_k, p(x_k)).$$
- Postać (agent, robot) będzie podążała zgodnie z planem, aż do doświadczenia niezgodności pomiędzy znanym (zakładanym) kosztem przejścia a faktycznym.

Clue algorytmu Stentz'a

- Dla pewnego stanu x załóżmy, że istnieje akcja a przeprowadzająca go w sąsiada x'. Jeżeli mamy że:

$$g_{\mathrm{via}}(x, x') < g_{\mathrm{current}}(x),$$

  to istnieje szansa, że koszt $g_{\mathrm{current}}(x)$ może być zredukowany.

- Jeżeli dodatkowo:

$$g_{\mathrm{current}}(x') \leqslant g_{\mathrm{best}}(x),$$

  to na pewno koszt $g_{\mathrm{current}}(x')$ jest **optymalny** w świetle znanych informacji.

- Gdy zachodzą oba te warunki to $g_{\mathrm{current}}(x)$ jest aktualizowane do $g_{\mathrm{via}}(x, x')$ i $p(x)$ jest aktualizowane do a.

Algorytm „zewnętrzny”

1. Zainicjalizuj wszystkie wartości $g_{\text{best}}, g_{\text{current}}, g_{\text{via}}$ zerami, oraz wszystkie p na puste.
2. Włóż stan docelowy G do kolejki Q.
3. Wykonuj w pętli **algorytm Stentz'a**, aż do momentu, kiedy jako jego wynik zostanie zwrócony stan początkowy S. *(w tym momencie algorytm Stentz'a zadziała równoważnie do wstecznego algorytmu Dijkstry)*
4. Główna pętla:
    1. Wykonuj obecny plan $p_1, p_2, \ldots$ odwiedzając ciąg stanów $x_{k+1} = t(x_k, p_k)$, gdzie x_1 oznacza aktualny stan S.
        1. Jeżeli dla pewnego x_k widać, że zastosowanie akcji p_k skutkowałoby kosztem większym niż znane $c(x_k, p_k)$, to uaktualnij $c(x_k, p_k)$ na prawdziwe, i przypisz:
           $$g_{\text{via}}(x_k, x_{k+1}) := c(x_k, p_k) + g_{\text{current}}(x_{k+1}),$$
           $$g_{\text{current}}(x_k) := g_{\text{via}}(x_k, x_{k+1}).$$
           Przerwij dalsze wykonywanie planu.
    2. Jeżeli $x_k = G$ to przerwij algorytm. *(punkt stopu)*
    3. Wstaw x_k do Q. Zapamiętaj $g_{\text{last}} := g_{\text{current}}(x_k)$. Ustaw $S := x_k$.
    4. Dopóki Q jest niepusta lub dopóki nie osiągnięto sytuacji, że $g_{\text{best}}(x) \geqslant g_{\text{last}}$ dla wszystkich x w Q:
        1. Wykonaj **algorytm Stentz'a**.

Algorytm Stentz'a (część 1)

1. Pobierz z Q stan x o najmniejszym g_{best}.
2. Jeżeli $g_{\text{best}}(x) < g_{\text{current}}(x)$
   (tzn. że x zwiększył swój koszt będąc w Q, i jeżeli można ten koszt poprawić podróżując przez jakiegoś sąsiada, dla którego optymalny koszt jest znany, to należy tak zrobić)
   1. Dla każdego $a \in A(x)$, takiego że $c(x,a) < \infty$, sprawdź dla $x' = t(x,a)$ czy: $g_{\text{via}}(x,x') < g_{\text{current}}(x)$ i $g_{\text{current}}(x') \leqslant g_{\text{best}}(x)$? Jeżeli tak, to:
      1. $g_{\text{current}}(x) := g_{\text{via}}(x,x')$.
      2. $p(x) := a$.
3. Dla wszystkich x', takich że istnieje $a' \in A(x')$ powodujące $t(x',a') = x$ oraz $c(x',a') < \infty$:
   1. $g_{\text{via}}(x',x) := c(x',a') + g_{\text{current}}(x)$.
   2. Jeżeli x' nie jest w V, to:
      1. $g_{\text{current}}(x') := g_{\text{via}}(x',x)$, $\quad g_{\text{best}}(x') := g_{\text{via}}(x',x)$.
      2. $p(x') := a'$.
      3. Włóż x' do Q.
   3. Jeżeli koszt dla x' wydaje się niepoprawny, bo $p(x') = a'$, ale $g_{\text{via}}(x',x) \neq g_{\text{current}}(x')$, to:
      1. $g_{\text{current}}(x') := g_{\text{via}}(x',x)$.
      2. Włóż x' do Q.

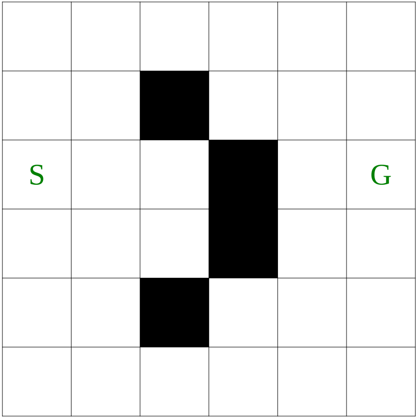
$\vdots$

Algorytm Stentz'a (część 2)

⋮

3. *(ciąg dalszy ciała pętli 3)*
   4. Jeżeli $p(x') \neq a'$ i $g_{\text{via}}(x', x) < g_{\text{current}}(x')$, to:
      (tzn. że lepiej iść z x' przez x niż użyć akcji $p(x')$)
      1. Jeżeli $g_{\text{current}}(x) = g_{\text{best}}(x)$, to: $p(x') := a'$ i włóż x' do Q, bo optymalny koszt dla x jest znany.
      2. W przeciwnym razie: $g_{\text{best}}(x) := g_{\text{current}}(x)$ (o ile $x \notin Q$), oraz włóż x do Q.
   5. *(unikanie cykli w p)*
      Jeżeli $x' \in V$ i $x' \notin Q$, oraz
      $p(x') \neq a'$ i $g_{\text{via}}(x, x') < g_{\text{current}}(x)$ i $g_{\text{current}}(x) > g_{\text{best}}(x)$,
      to włóż x' do Q ponownie.
4. Umieść x w V.

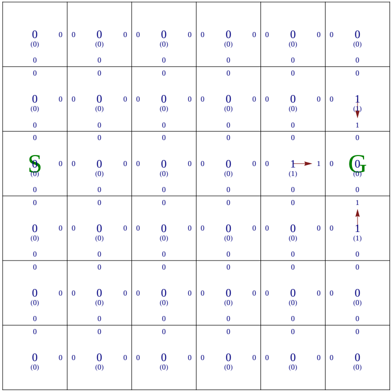
Przykład „ślepa uliczka” (1)



Przykład „ślepa uliczka” (2)

Na rysunkach g_{current} pisane w środku, g_{best} w nawiasie, g_{via} przy bokach. W opisie Q schemat: $x \to g_{\mathrm{best}}(x)\{g_{\mathrm{current}}(x)\}, \ldots$

(a)



$x = (3, 6)$ pobrane z Q

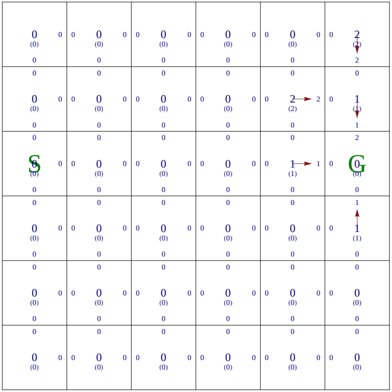
3.2 wykonany dla $x' = (2, 6)$

3.2 wykonany dla $x' = (4, 6)$

3.2 wykonany dla $x' = (3, 5)$

$Q : (2, 6) \to 1\{1\},\ (3, 5) \to 1\{1\},\ (4, 6) \to 1\{1\}$

(b)



$x = (2, 6)$ pobrane z Q

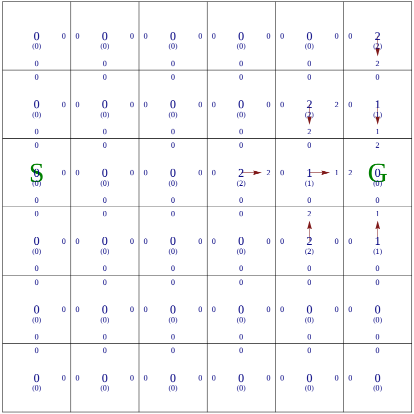
3.2 wykonany dla $x' = (1, 6)$

3.2 wykonany dla $x' = (2, 5)$

$Q : (3, 5) \to 1\{1\},\ (4, 6) \to 1\{1\},\ (1, 6) \to 2\{2\},\ (2, 5) \to 2\{2\}$

Przykład „ślepa uliczka” (3)

(c)



$x = (3,5)$ pobrane z Q

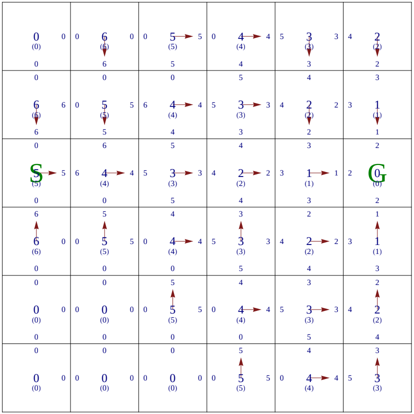
3.2 wykonany dla $x' = (2,5)$

3.2 wykonany dla $x' = (4,5)$

3.2 wykonany dla $x' = (3,4)$

$Q : (4,6) \to 1\{1\},\ (1,6) \to 2\{2\},\ (3,4) \to 2\{2\},\ (4,5) \to 2\{2\},$
$(2,5) \to 2\{2\}$

$\cdots$ (d) (**wsteczny a. Dijkstry zakończony**)

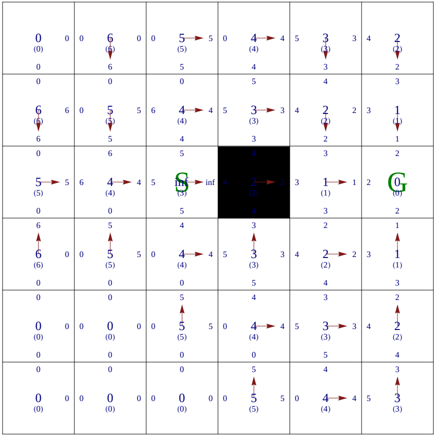


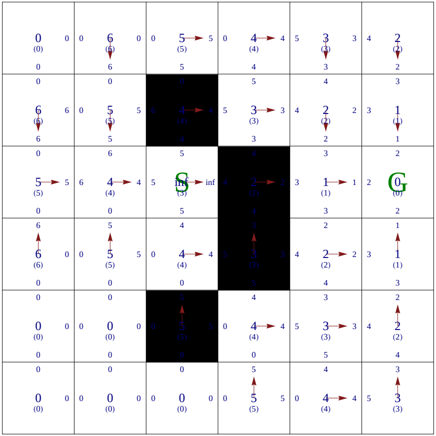
$\vdots$

$Q : (4,2) \to 5\{5\},\ (1,3) \to 5\{5\},\ (5,3) \to 5\{5\},$
$(6,4) \to 5\{5\},\ (1,2) \to 6\{6\},\ (4,1) \to 6\{6\},$
$(2,1) \to 6\{6\}$

Przykład „ślepa uliczka” (4)

Plan z aktualnego S**:** $(\rightarrow, \rightarrow, \rightarrow, \rightarrow, \rightarrow)$**. Osiągnięto stan** $(3,3)$**. Niezgodność wykryta dla** $c\,((3,3), \rightarrow)$.





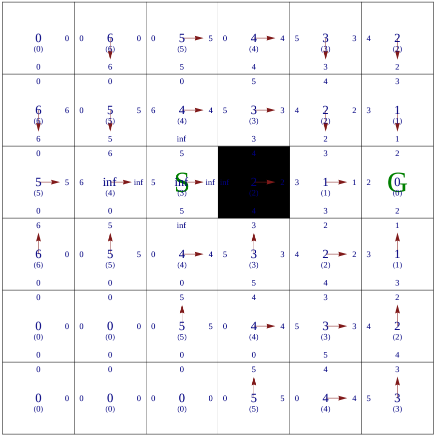
$(3,3)$ włożony do Q

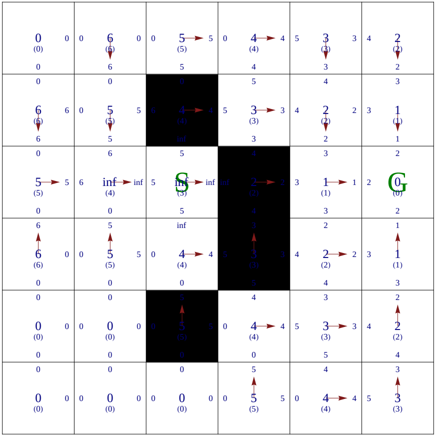
$g_{\text{last}} = \infty,$

$Q : (3,3) \rightarrow 3\{\infty\},\ (4,2) \rightarrow 5\{5\},\ (1,3) \rightarrow 5\{5\},\ (5,3) \rightarrow 5\{5\},\ (6,4) \rightarrow 5\{5\},\ (1,2) \rightarrow 6\{6\},$
$(4,1) \rightarrow 6\{6\},\ (2,1) \rightarrow 6\{6\}$

Przykład „ślepa uliczka” (5)

Po pierwszym przebiegu algorytmu Stentz’a.





$x = (3,3)$ pobrane z Q

3.5 wykonany dla $x' = (2,3)$

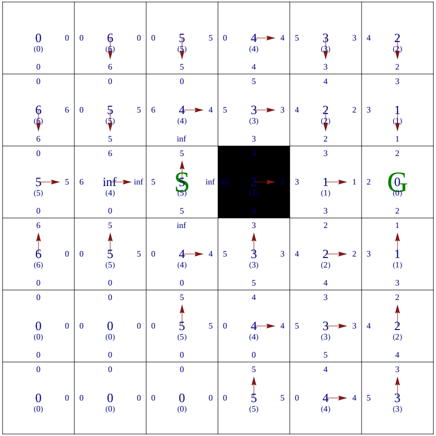
3.5 wykonany dla $x' = (4,3)$

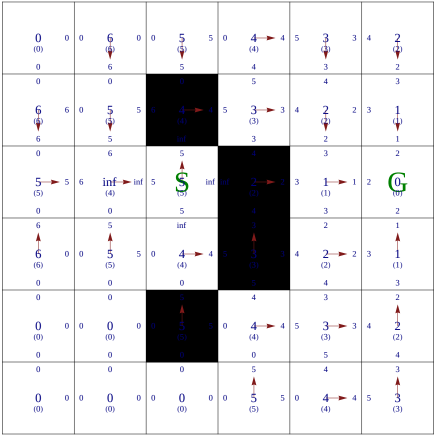
3.3 wykonany dla $x' = (3,2)$

$Q : (2,3) \to 4\{4\},\ (3,2) \to 4\{\infty\},\ (4,3) \to 4\{4\},\ (4,2) \to 5\{5\},\ (1,3) \to 5\{5\}$
$(5,3) \to 5\{5\},\ (6,4) \to 5\{5\},\ (1,2) \to 6\{6\},\ (4,1) \to 6\{6\},\ (2,1) \to 6\{6\}$

Przykład „ślepa uliczka” (6)

Po drugim przebiegu algorytmu Stentz’a.





$x = (2,3)$ pobrane z Q

3.2 wykonany dla $x' = (1,3)$ (neutralny)

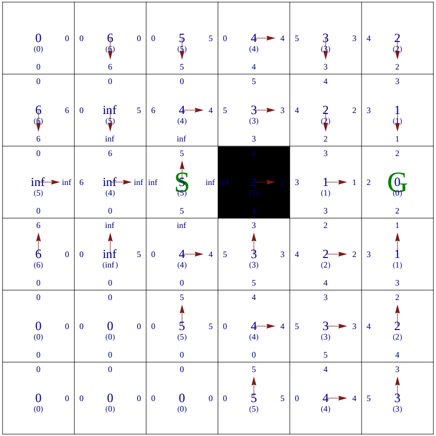
3.4.1 wykonany dla $x' = (3,3)$ — przekierowanie z x' na x

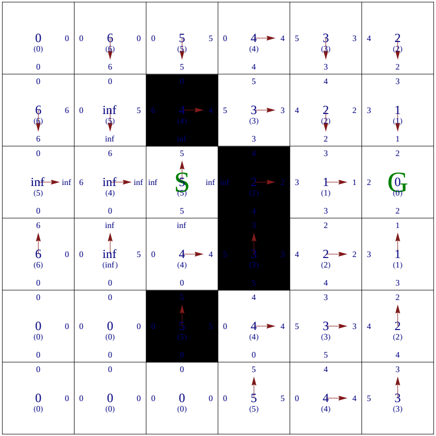
$Q : (3,2) \to 4\{\infty\},\ (4,3) \to 4\{4\},\ (4,2) \to 5\{5\}, (3,3) \to 5\{5\},\ (1,3) \to 5\{5\}$

$(5,3) \to 5\{5\},\ (6,4) \to 5\{5\},\ (1,2) \to 6\{6\},\ (4,1) \to 6\{6\},\ (2,1) \to 6\{6\}$

Przykład „ślepa uliczka” (7)

Po trzecim przebiegu algorytmu Stentz’a.





$x = (3,2)$ pobrane z Q

3.3 wykonany dla $x' = (2,2)$

3.2 wykonany dla $x' = (4,2)$

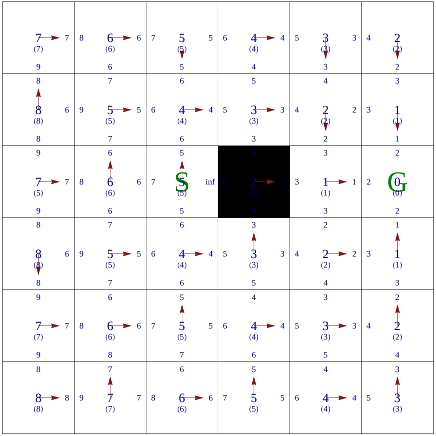
3.3 wykonany dla $x' = (3,1)$

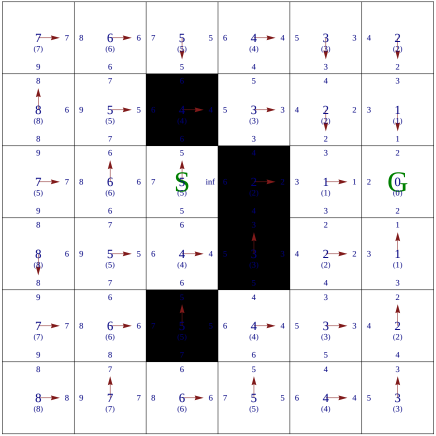
$Q : (4,3) \to 4\{4\},\ (2,2) \to 5\{\infty\},\ (3,1) \to 5\{\infty\},\ (3,3) \to 5\{5\},\ (1,3) \to 5\{5\}$

$(5,3) \to 5\{5\},\ (6,4) \to 5\{5\},\ (1,2) \to 6\{6\},\ (4,1) \to 6\{6\},\ (2,1) \to 6\{6\},\ (4,2) \to \infty\{\infty\}$

Przykład „ślepa uliczka” (8)

. . . Kolejka opróżniona po 22 przebiegach algorytmu Stentz’a. Osiągnięty stan algortymu:

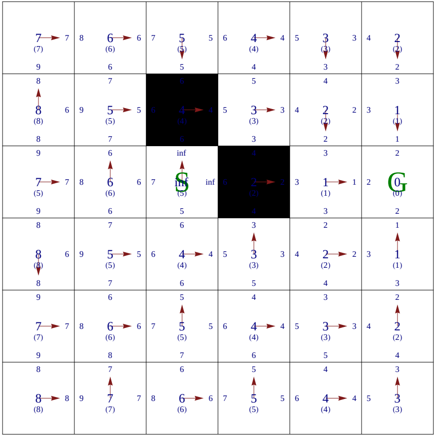


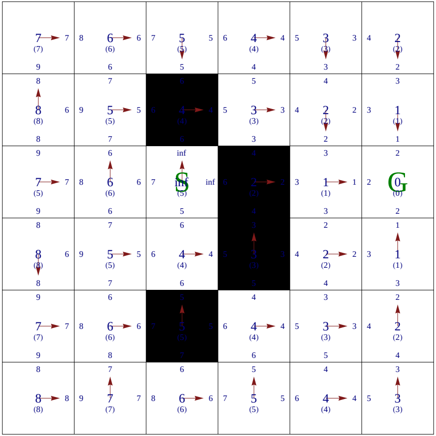


$Q : \emptyset$

Przykład „ślepa uliczka” (9)

Plan z aktualnego S**:** $(\uparrow, \rightarrow, \rightarrow, \downarrow, \rightarrow)$**. Osiągnięto stan** $(3,3)$**. Niezgodność wykryta dla** $c\,((3,3),\uparrow)$.





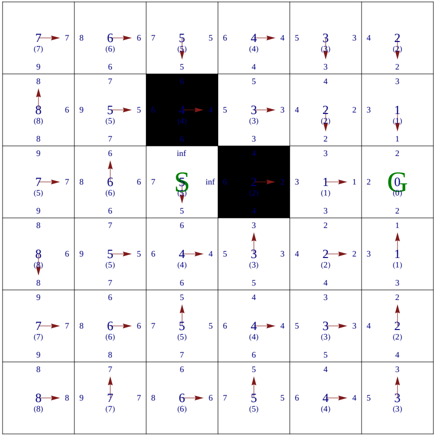
$(3,3)$ włożony do Q

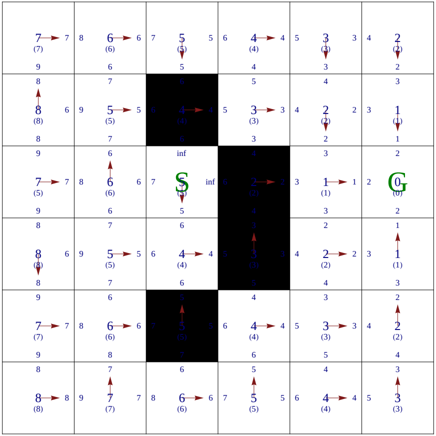
$g_{\text{last}} = \infty,$

$Q : (3,3) \rightarrow 5\{\infty\}$

Przykład „ślepa uliczka” (10)

Kolejka opróżniona po 1 przebiegu algorytmu Stentz’a.





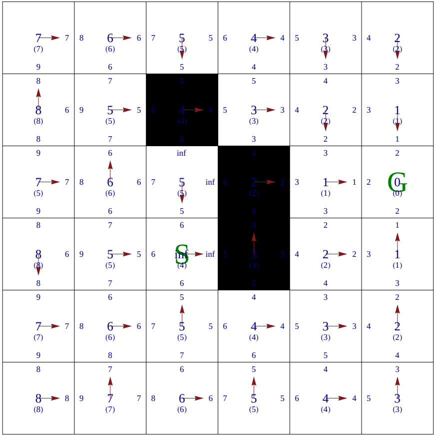
$x = (3, 3)$ pobrane z Q

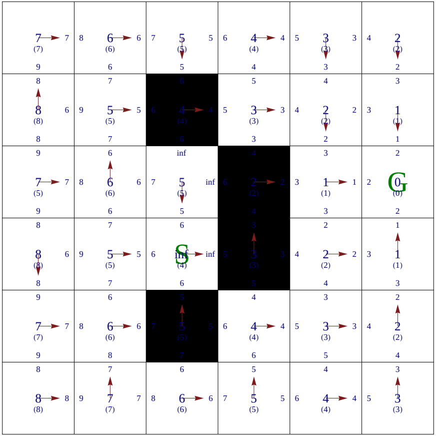
2.1 wykonany dla $x' = (4, 3)$ — przekierowanie z x na x'

$Q : \emptyset$

Przykład „ślepa uliczka” (11)

Plan z aktualnego S**:** $(\downarrow, \rightarrow, \uparrow, \rightarrow, \rightarrow)$**. Osiągnięto stan** $(4,3)$**. Niezgodność wykryta dla** $c\,((4,3), \rightarrow)$.





$(4,3)$ włożony do Q

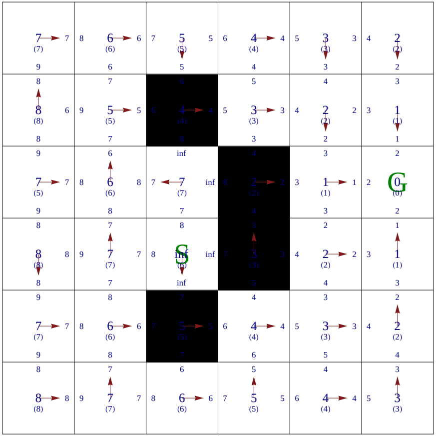
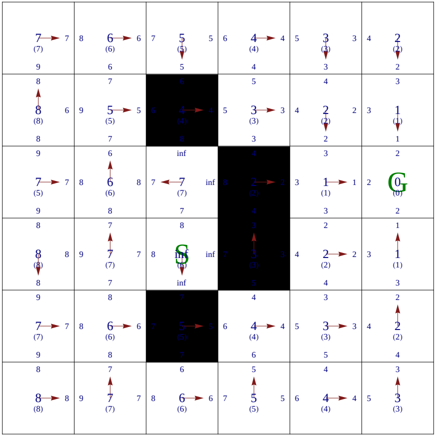
$g_{\text{last}} = \infty,$

$Q : (4,3) \rightarrow 4\{\infty\}$

Przykład „ślepa uliczka” (12)

Kolejny plan wyprowadzony po 10 przebiegach algorytmu Stentz’a.

Plan z aktualnego S**:** $(\downarrow, \rightarrow, \rightarrow, \rightarrow, \uparrow, \uparrow)$**. Osiągnięto stan** $(4,3)$**. Niezgodność wykryta dla** $c\,((4,3),\downarrow)$.

$(4,3)$ włożony do Q

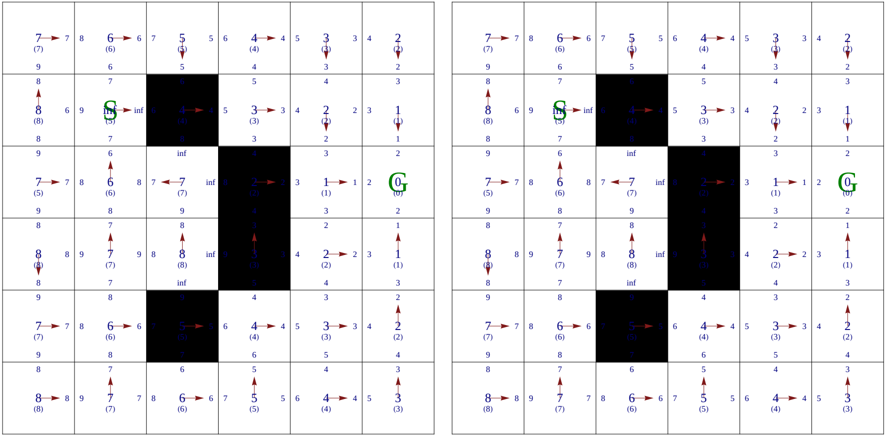
$g_{\text{last}} = \infty,$

$Q : (4,3) \rightarrow 6\{\infty\}$

Przykład „ślepa uliczka” (13)

Kolejny plan wyprowadzony po 4 przebiegach algorytmu Stentz'a.

Plan z aktualnego S**:** $(\uparrow, \leftarrow, \uparrow, \rightarrow, \rightarrow, \rightarrow\downarrow, \rightarrow)$**. Osiągnięto stan** $(2,2)$**. Niezgodność wykryta dla** $c\,((2,2), \rightarrow)$.



$(2,2)$ włożony do Q

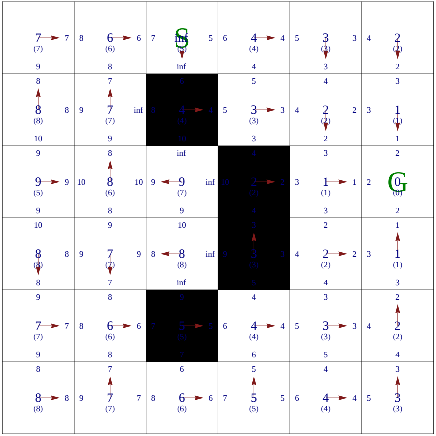
$g_{\text{last}} = \infty,$

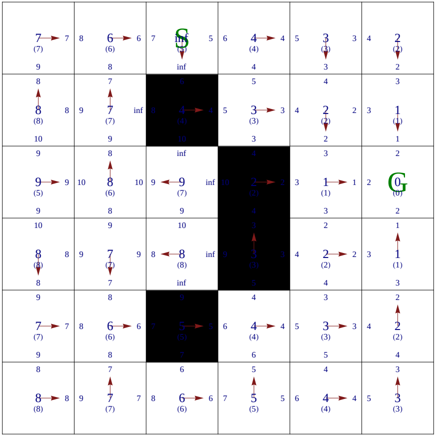
$Q : (2,2) \rightarrow 5\{\infty\}$

Przykład „ślepa uliczka” (14)

Kolejny plan wyprowadzony po 12 przebiegach algorytmu Stentz’a.

Plan z aktualnego S**:** $(\uparrow, \rightarrow, \downarrow, \rightarrow, \rightarrow, \downarrow, \rightarrow)$**. Osiągnięto stan** $(1, 3)$**. Niezgodność wykryta dla** $c\,((3,1), \downarrow)$.





$(1, 3)$ włożony do Q

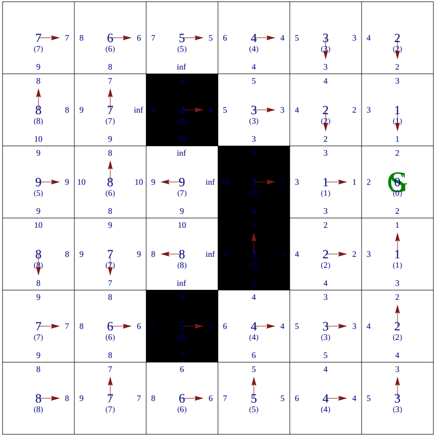
$g_{\mathrm{last}} = \infty,$

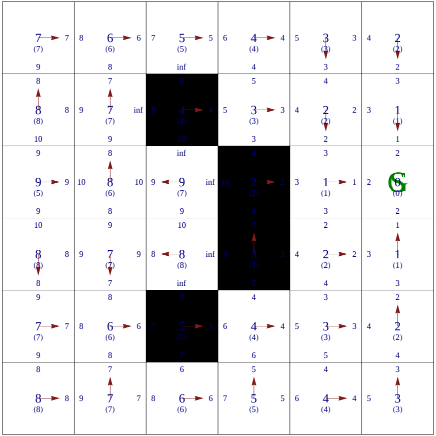
$Q : (1, 3) \rightarrow 5\{\infty\}$

Przykład „ślepa uliczka” (15)

Kolejny plan wyprowadzony po 1 przebiegu algorytmu Stentz’a.

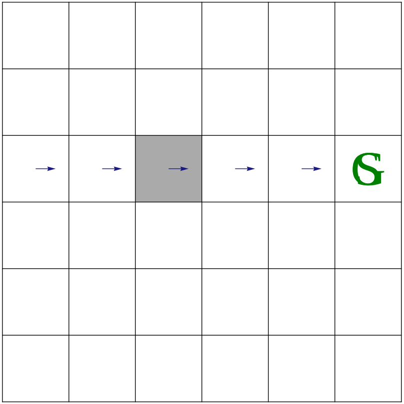
Plan z aktualnego S**:** $(\rightarrow, \rightarrow\downarrow, \downarrow, \rightarrow)$. **Osiągnięto stan** $G = (3, 6)$.



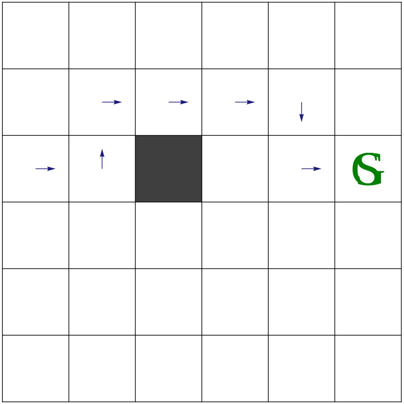


Przykład „przejść czy obejść?”

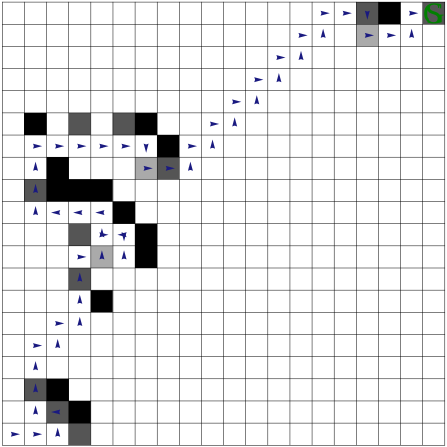
$c\left((3,2), \rightarrow\right) = 2$

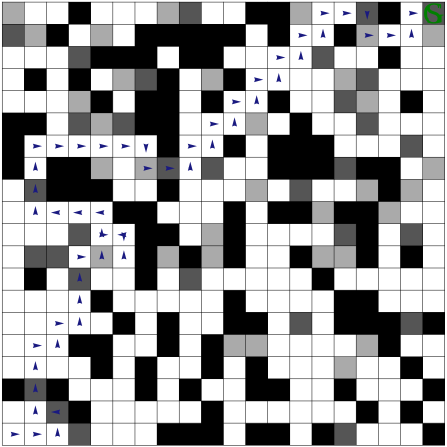


$c\left((3,2), \rightarrow\right) = 4$



Przykład dla siatki 20 × 20





Uwagi końcowe

- **Pole widzenia** Algorytm łatwo pozwala wprowadzić większe pole widzenia, jako otoczenie postaci o pewnym promieniu. Wystarczy przy podążaniu wg planu $p_1, p_2, \ldots$ uaktualnić wszelkie niezgodności funkcji c wykryte w otoczeniu. Zmiany wymaga tylko krok 4.1 algorytmu „zewnętrznego".
- Większe pole widzenia w przypadku średnim, powinno poprawiać trasę — zmniejszać tendencje do błądzenia.
- Algorytm bywa często wypowiadany w formie wprowadzającej **etykiety dla stanów**: *RAISE*, *LOWER*. Etykieta *RAISE* sygnalizuje, że koszt stanu jest większy niż ostatnio znany w kolejce Q (związane z krokiem 2.1 w podanym algorytmie Stentz'a). Etykieta *LOWER* sygnalizuje, że koszt stanu jest niższy niż ostatnio znany w kolejce Q (związane z krokiem 3.4 w podanym algorytmie Stentz'a).
- Praktyczne zastosowania militarne do w **eksploracyjnych pojazdach bezzałogowych** m.in.: *Automated Cross-Country Unmanned Vehicle (XUV)*.