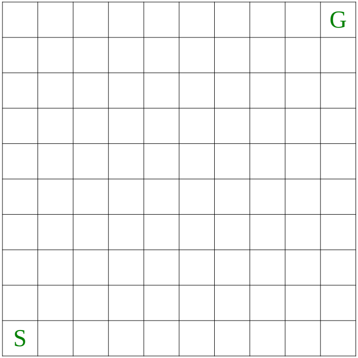
Stentz's algorithm
D^*

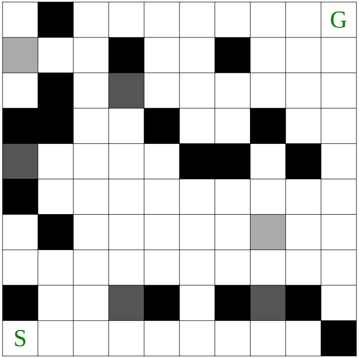
Przemysław Klęsk
pklesk@wi.zut.edu.pl

Department of Methods of Artificial Intelligence and Applied Mathematics

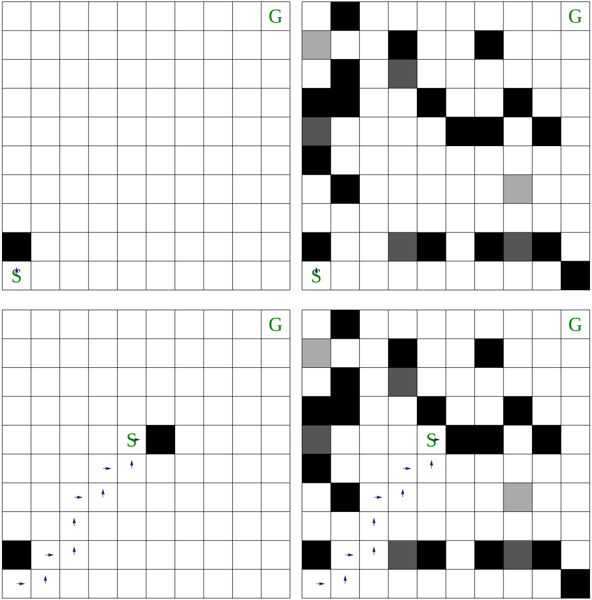
Problem

In an unknown area (or known only partially) the task is to reach a target position of given coordinates. Costs of transitions and obstacles are learned on-the-go during the path is traversed (examples: real mobile robots, artificial players in computer games).

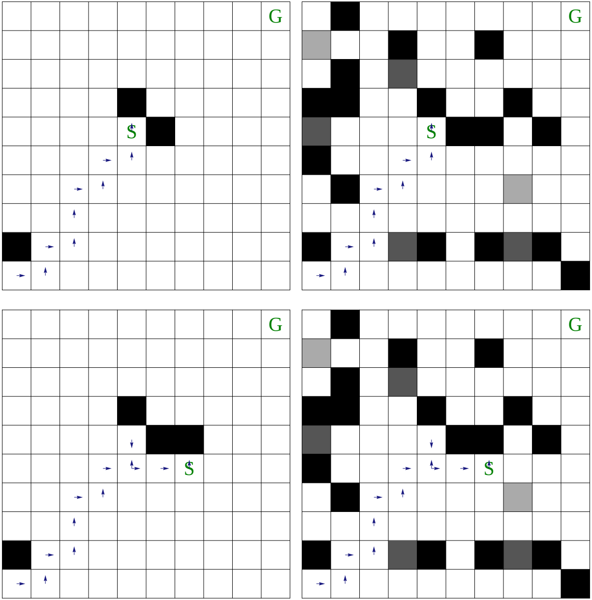




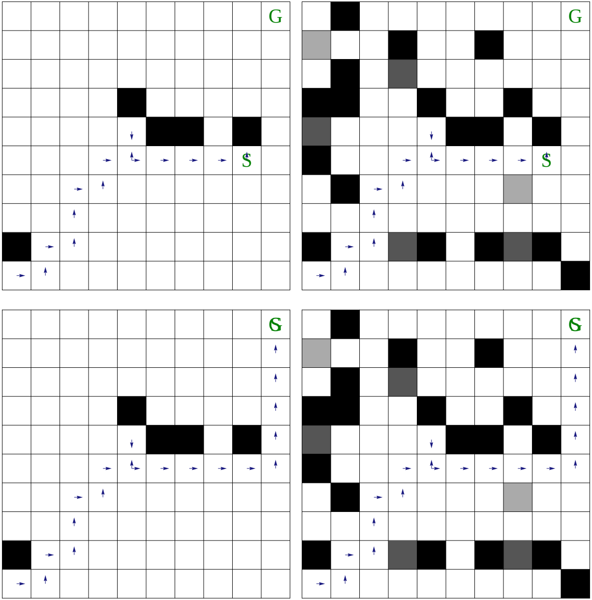
Example illustration (1)



Example illustration (2)



Example illustration (3)



Stentz's algorithm (1994) — properties

- Also known as D^*, with an intention the name is understood as *dynamic* A^*.
- Despite its name, carried out in a manner closer to Dijkstra's algorithm.
- Algorithm performs **optimal behavior** taking into account information learned so far.

Stentz's algorithm — properties

- Works iteratively — the planned path is derived multiple times.
- The first run of the algorithm is a *backward* versions of Dijkstra's algorithm — we built a queue of states by going from the goal towards the start point.
- During the actual traversal of the path, when a discrepancy between the knowledge so-far occurs, we update the path.
- States in the queue are allowed to change their costs multiple times and to enter the queue multiple times.
- The algorithm is more effective then multiple executions of Dijkstra's algorithm from scratch.

Sets of actions and states

- Let A denote **set of possible actions**. In particular, for regular grid of squares:
$$A = \{\uparrow, \rightarrow, \downarrow, \leftarrow\}.$$
- Let X denote **set of states**. For regular grid of squares:
$$X = \{x_{ij}\},$$
where i is row index, j is column index.
- By $A(x)$ set of actions possible for state x shall be denoted.

Executing actions

- Let $t(x, a)$ (*transition*) denote a **function transiting** given state x via action a into a new state x', i.e.

$$t(x, a) = x'.$$

  E.g. for grid of squares $t(x_{25}, \rightarrow) = x_{26}$.

- Formally, t is a function $t\colon X \times A \rightarrow X$.

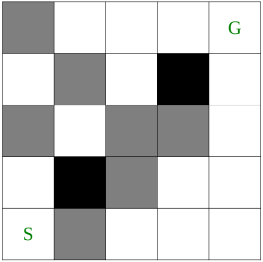
Costs of transitions

- Let $c(x,a)$ (*costs of transition*) denote a **function of taken cost**, which has to be taken when executing in x an action a.
- Formally, $c\colon X \times A \to \mathbb{R}_+ \cup \{\infty\}$.
- If in x execution of a is impossible (obstacle or map border), then $c(x,a) = \infty$.
- Such form of c allows for general map representation, where **transitions to certain state, but from different directions, can have different costs**.
- If, for grid of squares, we want to identify all transition costs to the same state x_{ij}, then:

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

for all i, j.

When costs of transitions identical with map



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

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

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

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

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

Initial assumptions

- In the most pessimistic case we assume complete lack of map knowledge, except for start and goal coordinates.
- If we assume that transitions untroubled by anything cost 1 unit, then in the case of complete unawareness, one shall impose:

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

  In particular, we assume that positions of map borders are not known as well.

Functions/quantities used in the algorithm

- Let $g_{\text{current}}(x)$ denote currently known cost of transition from x to goal G.
- Let $g_{\text{via}}(x, x')$ denote currently known cost of transition from x to goal G, when *traveling via* x'.
- Function g_{via} shall in fact be considered only for such x, x' which are direct neighbors.
- Let a be an action, s.t. $t(x, a) = x'$. Then:

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

Functions/quantities used in the algorithm

- Let Q denote the **queue of states** kept in algorithm (analogically to Dijkstra's or A^* algorithms).
- Let V denote the **map of visited states**.
- Let $g_{\text{best}}(x)$ denote the best (the lowest) known cost value for x during its lifetime in Q. It is known that:

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

- Q is ordered according to g_{best}.

Temporary plan

- Let $p(x)$ denote a **planned action** currently assigned to be executed in state x.
- The algorithm calculates (multiple times) a **temporary plan**, i.e. a certain sequence of actions

  $$p_1, p_2, \ldots, p_k, p_{k+1}, \ldots,$$

  which allow to travel from current start state to goal according to current knowledge of transition costs c. Therefore, a sequence of states is in the same time derived:

  $$x_1, x_2, \ldots, x_k, x_{k+1}, \ldots,$$

  such that

  $$x_{k+1} = t(x_k, p(x_k)).$$

- A person (agent, robot) shall travel on in accordance with the plan, until he (it) experiences a discrepancy between known (assumed) transition cost and a true one.

Core issue of Stentz's algorithm

- For certain x assume that there exists a which transits x into its neighbor x'. If we have that:
$$g_{\text{via}}(x, x') < g_{\text{current}}(x),$$
then there is a chance that cost $g_{\text{current}}(x)$ can be reduced.
- Additionally, if:
$$g_{\text{current}}(x') \leqslant g_{\text{best}}(x),$$
then the cost $g_{\text{current}}(x')$ is for certain **optimal** in the light of information at disposal.
- When both conditions are met then $g_{\text{current}}(x)$ is updated to $g_{\text{via}}(x, x')$ and $p(x)$ is updated to a.

„Outer" algorithm

1. Initialize all $g_{\text{best}}, g_{\text{current}}, g_{\text{via}}$ with zeros, and all p with voids.
2. Insert goal state G into queue Q.
3. In a loop, perform iteratively **Stentz's algorithm**, until as its result the start state S is returned. *(at that moment Stentz's algorithm works equivalently to a backward Dijkstra's algorithm)*
4. Main loop:
    1. Carry out current plan $p_1, p_2, \ldots$ visiting sequence of states $x_{k+1} = t(x_k, p_k)$, where x_1 denotes current S.
        1. If for certain x_k it can be observed that executing p_k would result in a cost greater than known $c(x_k, p_k)$, then update $c(x_k, p_k)$ to the true one, and assign:
           $$g_{\text{via}}(x_k, x_{k+1}) := c(x_k, p_k) + g_{\text{current}}(x_{k+1}),$$
           $$g_{\text{current}}(x_k) := g_{\text{via}}(x_k, x_{k+1}).$$
           Abort further execution of plan.
    2. If $x_k = G$ then stop the algorithm. *(stop condition)*
    3. Insert x_k into Q. Memorize $g_{\text{last}} := g_{\text{current}}(x_k)$. Set $S := x_k$.
    4. As long as Q is non-empty or until the condition $g_{\text{best}}(x) \geqslant g_{\text{last}}$ is not met for all x in Q:
        1. Perform **Stentz's algorithm**.

Stentz's algorithm (part 1)

1. Poll from Q the state x with the lowest g_{best}.
2. If $g_{\text{best}}(x) < g_{\text{current}}(x)$
   (it means x has increased its cost while being in Q, and if this cost could be reduced by traveling via some neighbor for which an optimal cost is known, then one should do so)
   1. For all $a \in A(x)$, s.t. $c(x,a) < \infty$, check for $x' = t(x,a)$ if: $g_{\text{via}}(x,x') < g_{\text{current}}(x)$ and $g_{\text{current}}(x') \leqslant g_{\text{best}}(x)$? If so, then:
      1. $g_{\text{current}}(x) := g_{\text{via}}(x,x')$.
      2. $p(x) := a$.
3. For all x', s.t. there exists $a' \in A(x')$ causing $t(x',a') = x$ and $c(x',a') < \infty$:
   1. $g_{\text{via}}(x',x) := c(x',a') + g_{\text{current}}(x)$.
   2. If x' is not in V then:
      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. Insert x' into Q.
   3. If cost for x' seems to be incorrect because $p(x') = a'$, but $g_{\text{via}}(x',x) \neq g_{\text{current}}(x')$, then:
      1. $g_{\text{current}}(x') := g_{\text{via}}(x',x)$.
      2. Insert x' into Q.

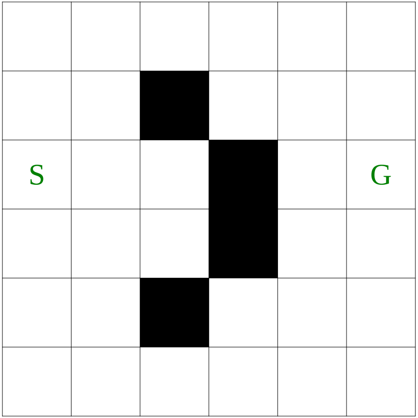
⋮

Stentz's algorithm (part 2)

⋮

3. *(continuation of loop's 3 body)*
    4. If $p(x') \neq a'$ and $g_{\text{via}}(x', x) < g_{\text{current}}(x')$ then:
       (it means that it is better to go from x' via x than to use action $p(x')$)
        1. If $g_{\text{current}}(x) = g_{\text{best}}(x)$ then: $p(x') := a'$ and insert x' into Q, because optimal cost for x is known.
        2. Otherwise: $g_{\text{best}}(x) := g_{\text{current}}(x)$ (if $x \notin Q$), and insert x into Q.
    5. *(avoiding cycles in p)*
       If $x' \in V$ and $x' \notin Q$, and
       $p(x') \neq a'$ and $g_{\text{via}}(x, x') < g_{\text{current}}(x)$ and $g_{\text{current}}(x) > g_{\text{best}}(x)$,
       then insert x' into Q again.
4. Put x into V.

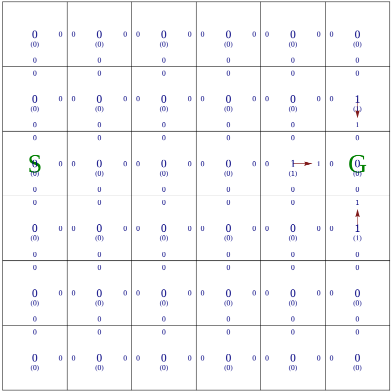
Example „dead end” (1)



Example „dead end” (2)

In figures g_{current} in the center, g_{best} in parenthesis, g_{via} at sides. In description of Q the scheme is: $x \rightarrow g_{\text{best}}(x)\{g_{\text{current}}(x)\}, \ldots$

(a)



$x = (3,6)$ polled from Q

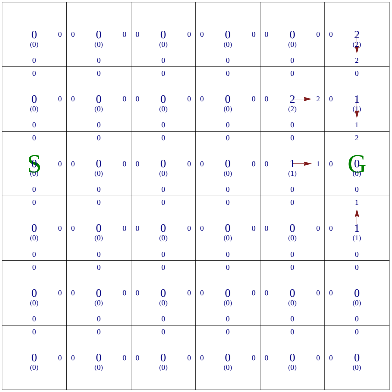
3.2 executed for $x' = (2,6)$

3.2 executed for $x' = (4,6)$

3.2 executed for $x' = (3,5)$

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

(b)



$x = (2,6)$ polled from Q

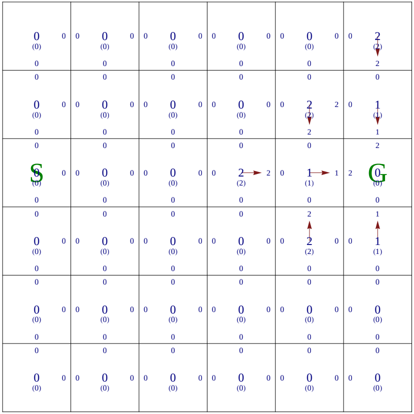
3.2 executed for $x' = (1,6)$

3.2 executed for $x' = (2,5)$

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

Example „dead end” (3)

(c)



$x = (3,5)$ polled from Q

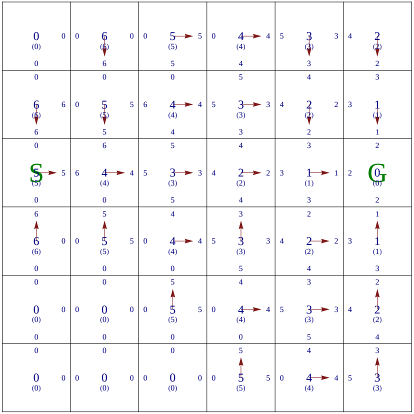
3.2 executed for $x' = (2,5)$

3.2 executed for $x' = (4,5)$

3.2 executed for $x' = (3,4)$

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

· · · (d) (**backward Dijkstra's done**)

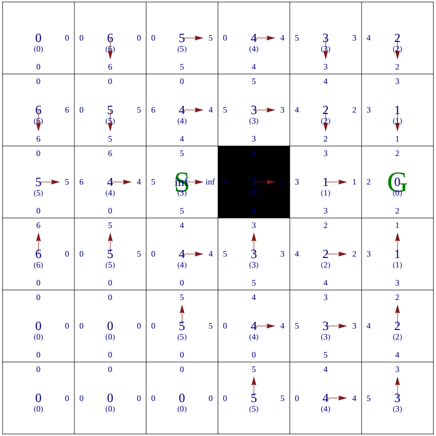


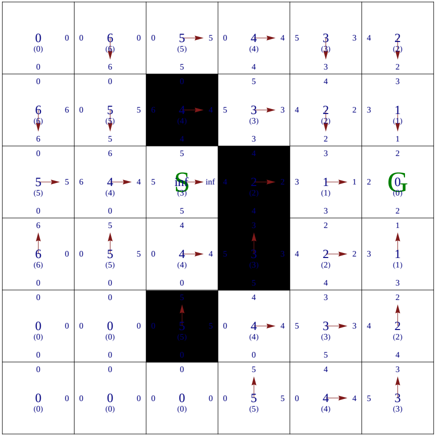
$\vdots$

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

Example „ślepa uliczka” (4)

Plan from current S**:** $(\rightarrow, \rightarrow, \rightarrow, \rightarrow, \rightarrow)$. **State** $(3,3)$ **reached. Discrepancy experienced for** $c\,((3,3), \rightarrow)$.





$(3,3)$ inserted into Q

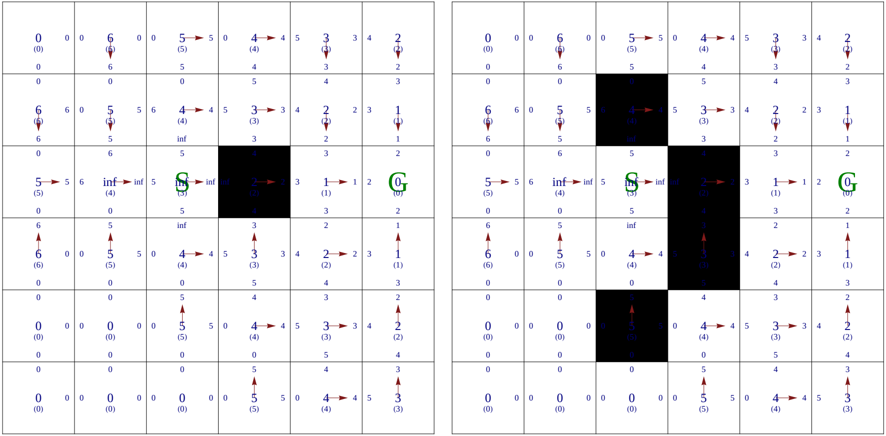
$g_{\text{last}} = \infty$,

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

$(4,1) \rightarrow 6\{6\},\ (2,1) \rightarrow 6\{6\}$

Example „dead end” (5)

After first run of Stentz’s algorithm.



$$x = (3,3) \text{ polled from } Q$$

$$3.5 \text{ executed for } x' = (2,3)$$

$$3.5 \text{ executed for } x' = (4,3)$$

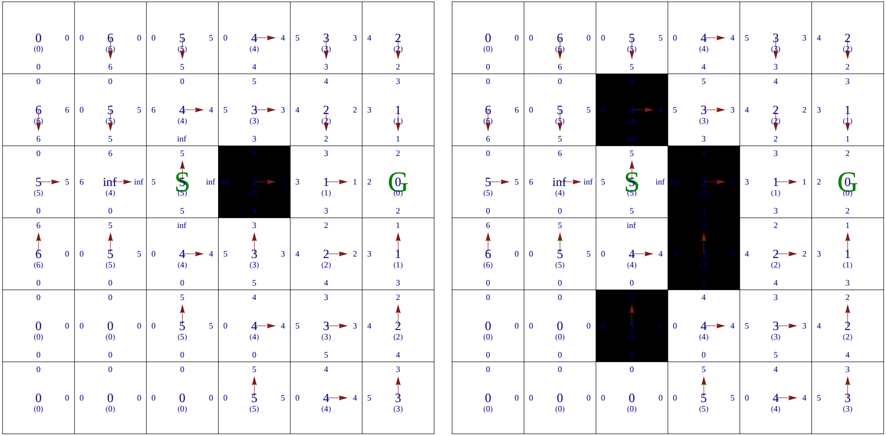
$$3.3 \text{ executed for } x' = (3,2)$$

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

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

Example „dead end" (6)

After second run of Stentz's algorithm.



$x = (2,3)$ polled from Q

3.2 executed for $x' = (1,3)$ (neutral)

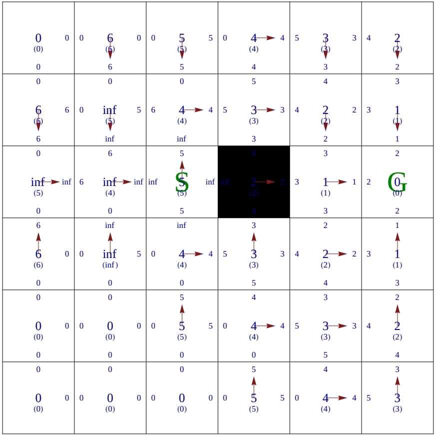
3.4.1 executed for $x' = (3,3)$ — redirection from x' onto x

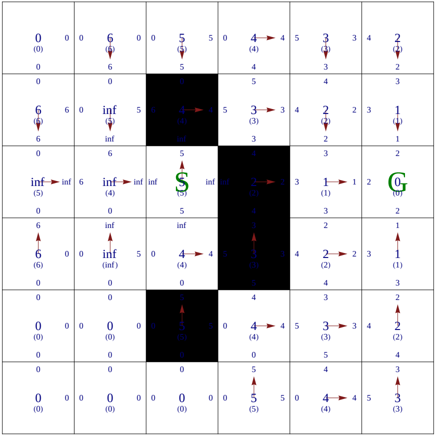
$Q : (3,2) \to 4\{\infty\},\ (4,3) \to 4\{4\},\ (4,2) \to 5\{5\},\ (3,3) \to 5\{5\},\ (1,3) \to 5\{5\}$

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

Example „dead end" (7)

After third run of Stentz's algorithm.





$x = (3, 2)$ polled from Q

3.3 executed for $x' = (2, 2)$

3.2 executed for $x' = (4, 2)$

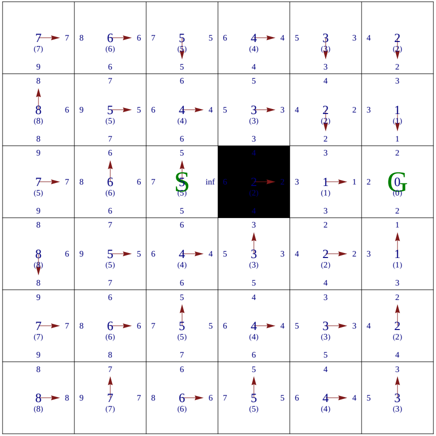
3.3 executed for $x' = (3, 1)$

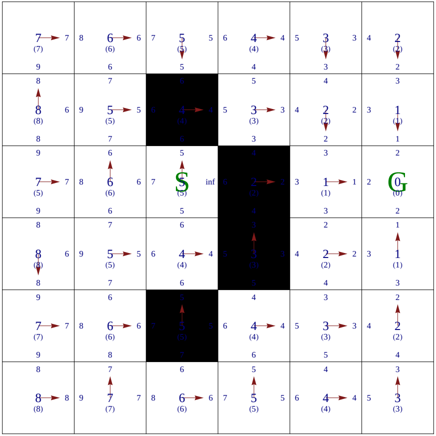
$Q : (4, 3) \to 4\{4\},\ (2, 2) \to 5\{\infty\},\ (3, 1) \to 5\{\infty\},\ (3, 3) \to 5\{5\},\ (1, 3) \to 5\{5\}$

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

Example „dead end” (8)

. . . Queue empty after 22 runs of Stentz’s algorithm. Reached state of algorithm:

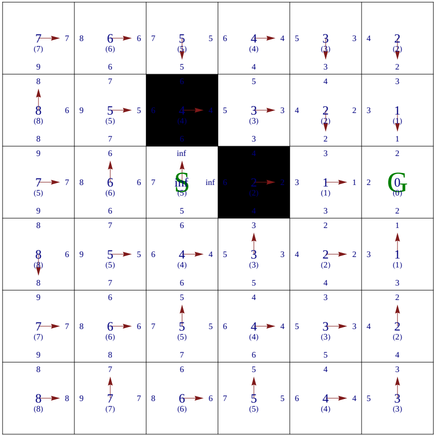


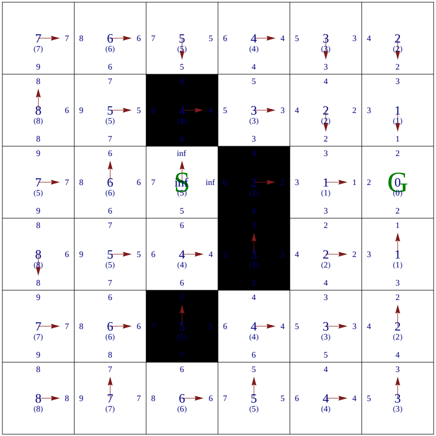


$Q : \emptyset$

Example „dead end” (9)

Plan from current S: $(\uparrow, \rightarrow, \rightarrow, \downarrow, \rightarrow)$. State $(3,3)$ reached. Discrepancy experienced for $c\,((3,3),\uparrow)$.





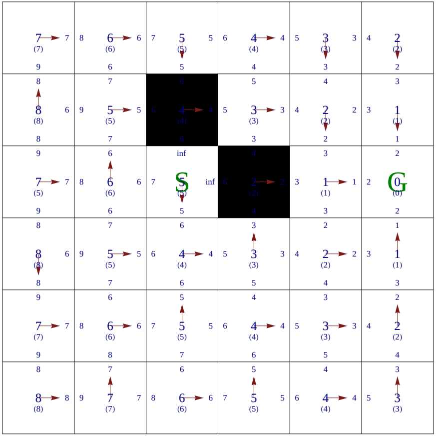
$(3,3)$ inserted into Q

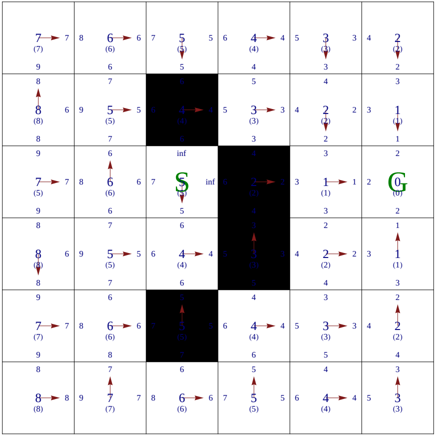
$g_{\text{last}} = \infty$,

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

Example „dead end" (10)

Queue empty after 1 run of Stentz's algorithm.





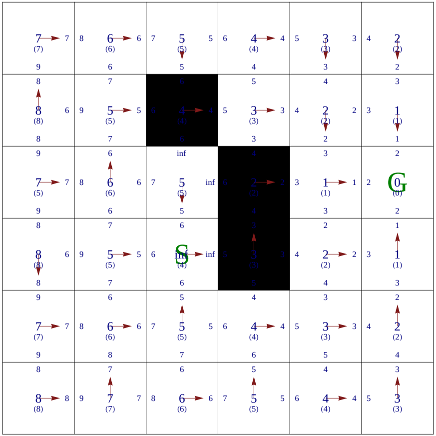
$x = (3,3)$ polled from Q

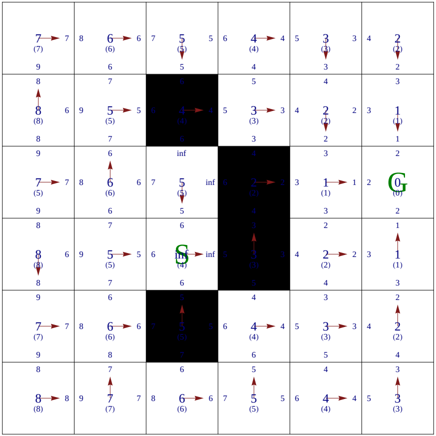
2.1 executed for $x' = (4,3)$ — redirection from x onto x'

$Q : \emptyset$

Example „dead end” (11)

Plan from current S: $(\downarrow, \rightarrow, \uparrow, \rightarrow, \rightarrow)$. State $(4,3)$ reached. Discrepancy experienced for $c\,((4,3), \rightarrow)$.





$(4,3)$ inserted into Q

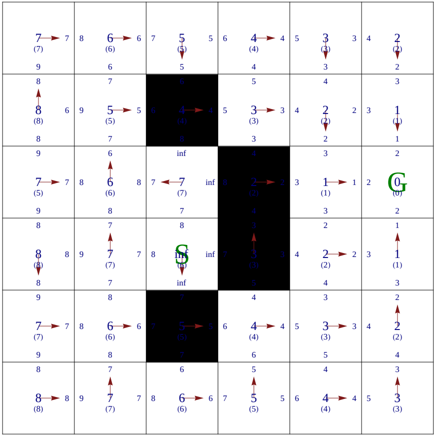
$g_{\text{last}} = \infty,$

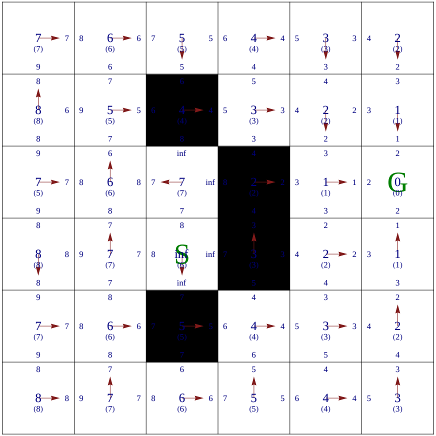
$Q : (4,3) \rightarrow 4\{\infty\}$

Example „dead end” (12)

New plan calculated after 10 runs of Stentz's algorithm.

Plan from current S**:** $(\downarrow, \rightarrow, \rightarrow, \rightarrow, \uparrow, \uparrow)$**. State** $(4,3)$ **reached. Discrepancy experienced for** $c\,((4,3),\downarrow)$.





$(4,3)$ włożony do Q

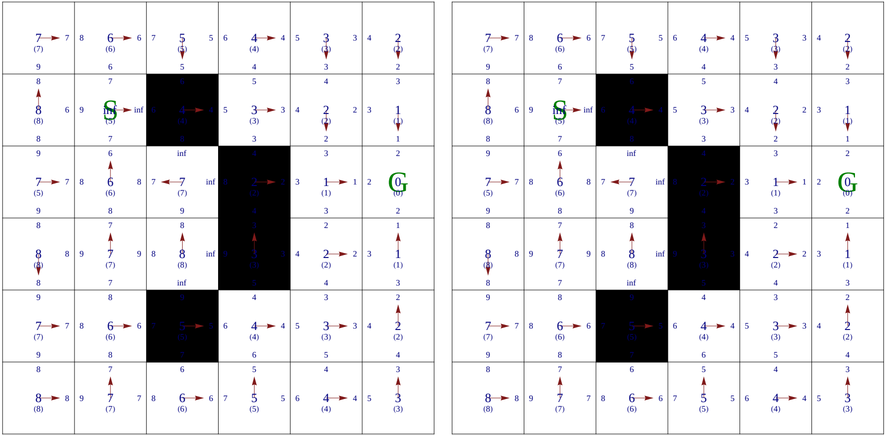
$g_{\text{last}} = \infty,$

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

Example „dead end” (13)

New plan calculated after 4 runs of Stentz's algorithm.

Plan from current S: $(\uparrow, \leftarrow, \uparrow, \rightarrow, \rightarrow, \rightarrow\downarrow, \rightarrow)$. State $(2,2)$ reached. Discrepancy experienced for $c((2,2), \rightarrow)$.



$(2,2)$ inserted into Q

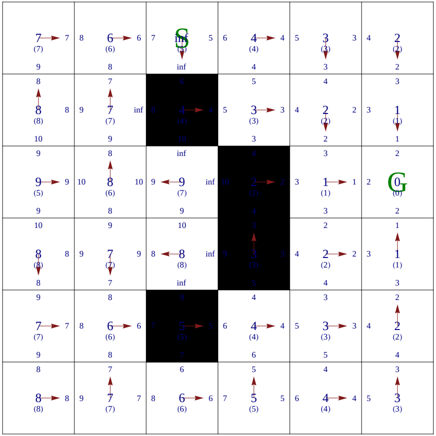
$g_{\text{last}} = \infty,$

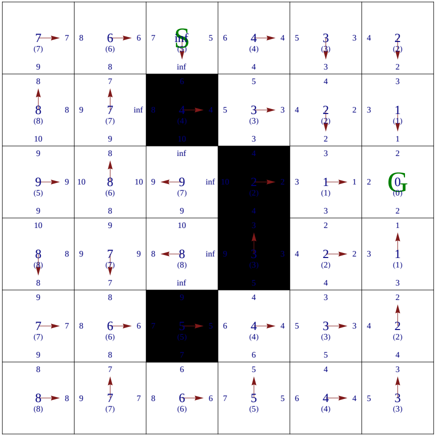
$Q : (2,2) \rightarrow 5\{\infty\}$

Example „dead end" (14)

New plan calculated after 12 runs of Stentz's algorithm.

Plan from current S**:** $(\uparrow, \rightarrow, \downarrow, \rightarrow, \rightarrow, \downarrow, \rightarrow)$. **State** $(1,3)$ **reached. Discrepancy experienced for** $c\,((3,1), \downarrow)$.





$(1,3)$ inserted into Q

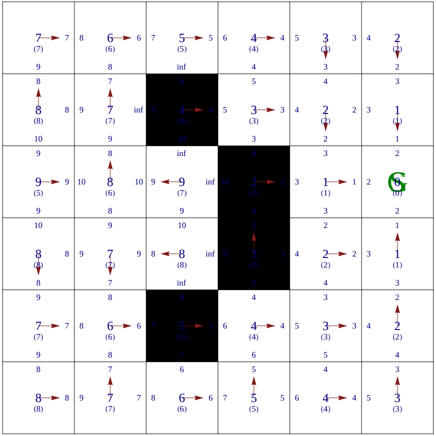
$g_{\text{last}} = \infty,$

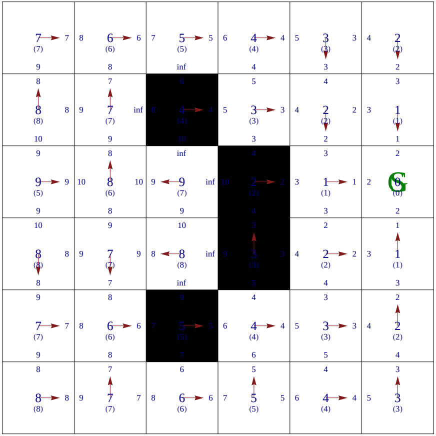
$Q : (1,3) \rightarrow 5\{\infty\}$

Example „dead end” (15)

New plan calculated after 1 run of Stentz’s algorithm.

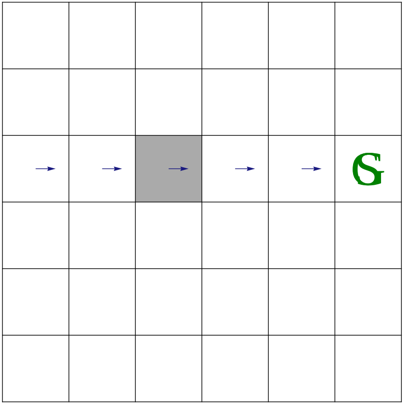
Plan from current S: $(\rightarrow, \rightarrow\downarrow, \downarrow, \rightarrow)$. **State $G = (3, 6)$ reached.**



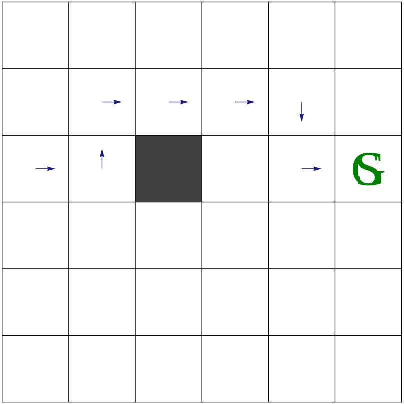


Example „to go through or to go arround?”

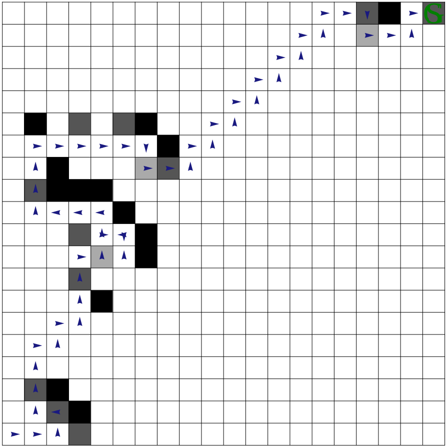
$c\left((3,2),\rightarrow\right) = 2$

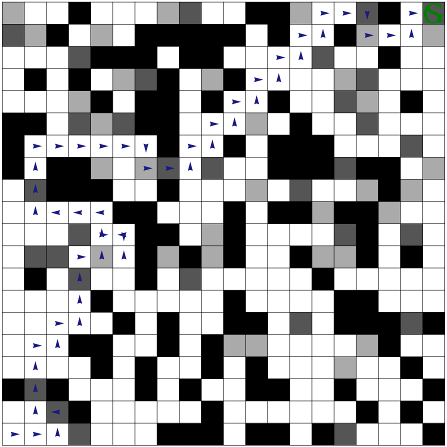


$c\left((3,2),\rightarrow\right) = 4$



Example for grid 20 × 20





Concluding remarks

- **Vision zone** The algorithm enables to introduce larger vision zone, as a neighborhood of the agent with certain radius. When following the plan $p_1, p_2, \ldots$ it suffices to update all discrepancies of function c discovered with such neighborhood. Only the step 4.1 int the „outer" algorithm requires a change.
- Larger vision zone should (in average case) improve the path i.e. decrease the tendency to roam unnecessarily.
- Algorithm is often presented by introducing **labels for states**: *RAISE*, *LOWER*. Label *RAISE* indicates a cost for a state greater than last known in Q (related to step 2.1 in given Stentz's algorithm). Label *LOWER* indicates a cost for a state lower than last known Q (related to step 3.4 in given Stentz's algorithm).
- Practical military applications in **exploratory unmanned vehicles** e.g.: *Automated Cross-Country Unmanned Vehicle (XUV)*.