Algorytm wykorzystuje heurystykę do sprawdzenia najbardziej obiecujących, jeszcze nie odkrytych wierzchołków.

Minimalizowana jest funkcja:

F(w) = h(w) + g(w)

Gdzie:

w - aktualny węzeł

F(w) - estymata całkowitego kosztu drogi przechodzącej przez wierzchołek w

h(w) - koszt drogi już przebytej

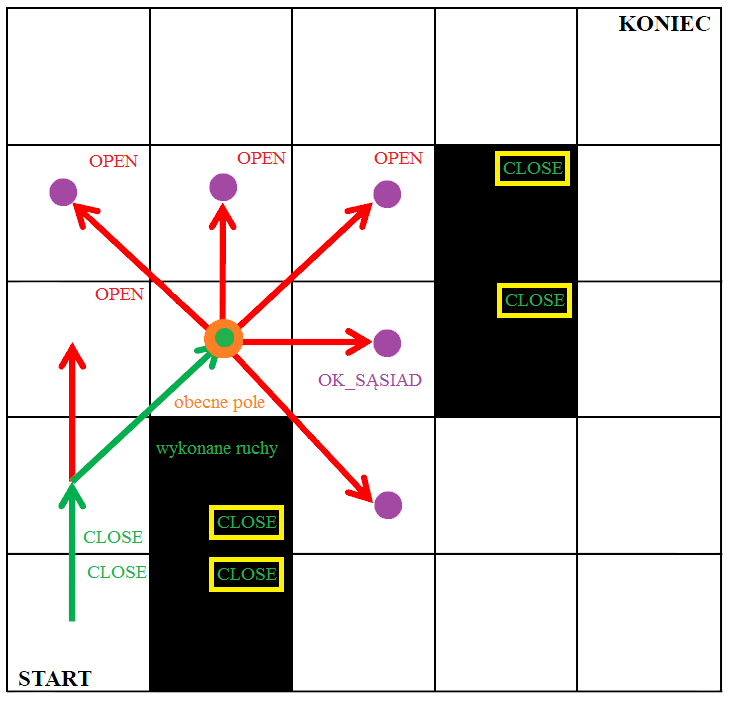
g(w) - estymata kosztu drogi łączącej wierzchołek w z końcem obliczony za pomocą heurystyki

Algorytm:

Idea programu składa się z tablic:

OPEN - tablica informująca, do których pól jest możliwość aby robot dotarł

CLOSE – tablica informująca, do których pól robot nie ma możliwości dotarcia lub odwiedzone te pola zostały już odwiedzone.



Algorytm:

Struktura OPEN :

OPEN(1,1)= 1;

OPEN(1,2)=currentX; % current X value

OPEN(1,3)=currentY; % current Y value

OPEN(1,4)=rodzic\_X; % rodzic X

OPEN(1,5)=rodzic\_Y; % rodzic Y

OPEN(1,6)=Hn; % Hn

OPEN(1,7)=Gn; % Gn

OPEN(1,8)=Fn; % G+H;

Struktura Sasiad

%% parametry wspolrzednych:

Sasiad(:,1)=sasiad\_x; % sasiad\_x

Sasiad(:,2)=sasiad\_y; % sasiad\_y %% parametry jakosci:

Sasiad(:,3)=Hn+dystans(Obecny punkt-> OK\_Sasiad); Hn

Sasiad(:,4)= dystans(Koncowy punkt-> OK\_Sasiad);); Gn

Sasiad(:,5)= Sasiad(:,3)+ Sasiad(:,4); Hn + Gn

Dodaj START -> OPEN

Dodaj przeszkody -> CLOSE

Pole=Start;

--------------POLACZ START Z KONCEM----------------

while(Pole<>Koniec || KoniecMozliwychTras)

dodaj OK\_SASIAD(obecny punkt) -> Sasiad

for n=1:nSasiad % dla kazdego n-sasiada

if (SASIAD == OPEN) sasiad istnieje w OPEN

if SASIAD.FN < OPEN.FN

OPEN.rodzic =Pole

OPEN.Gn = Sasiad.Gn

OPEN.Hn = Sasiad.Hn

End

End

if Sasiad != OPEN % nie jest jeszcze w OPEN

Dodaj Sasiad(n)-> OPEN

end

end

ZnajdzIndex = min(OPEN.Fn%-> STOP(-1) / PunktNowy

if (ZnajdzIndex ~=-1)

Pole = Pole z OPEN.Fn(ZnajdzIndex); %wykonaj ruch

Sciezka = Hn z OPEN.Fn(ZnajdzIndex);

Dodaj Pole -> CLOSE;

else

KoniecMozliwychTras =1;

end

end

----------------------POWROT-------------------------

Sciezka=[];

%ostatnio CLOSE

PolePowrot = CLOSE(end);

% poczatek powrotu

PolePowrot -> Sciezka

if (PolePowrot == Koniec)

rodzicPolePowrot=OPEN.rodzic(PolePowrot)

% dopisz rodzica do sciezki

rodzicPola -> Sciezka;

while(rodzicPolePowrot <> START)

% dopisz rodzica do sciezki

rodzicPolePowrot -> Sciezka

%szukaj rodzicow – rodzicow

rodzicPolePowrot=OPEN.rodzic(rodzicPolePowrot)

end;

end

Wyniki zaimplementowanego A\*