1. SEMINARSKA NALOGA

Nika Čelan Aleks Stepančič

Fakulteta za računalništvo in informatiko
Univerza v Ljubljani

20. november 2022

1. pristop

Najina prva ideja je bila, zgenerirati binarne osebke. Vsak osebek predstavlja zaporedje ukazov, kam naj se premakne v labirintu. Po dva znaka skupaj sta predstavljala en korak:

$$00 \rightarrow \textit{desno} \rightarrow$$
$$01 \rightarrow \textit{gor} \uparrow$$
$$10 \rightarrow \textit{levo} \leftarrow$$
$$11 \rightarrow \textit{dol} \downarrow$$

1. pristop

Fitness funkcija

1. Vzamemo ustrezna dva bita in ju pretvorimo v smer
2. Zapomnimo si polje v katerem smo pristali
3. V primeru da je trenutno polje zid, osebek kaznujemo

$$Vrni : Rezultat + |število| - K_1$$

4. Če smo v polju, v katerem smo že bili, osebek kaznujemo

$$Rezultat = |število| - K_2$$

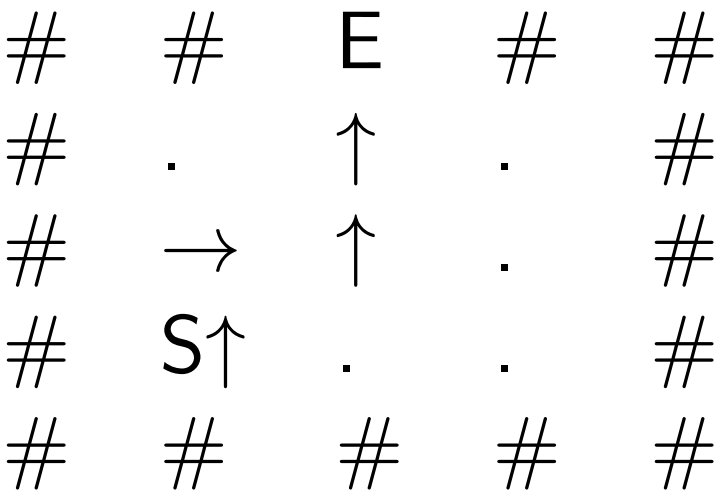
5. Če smo na cilju, osebek nagradimo ampak odštejemo število korakov

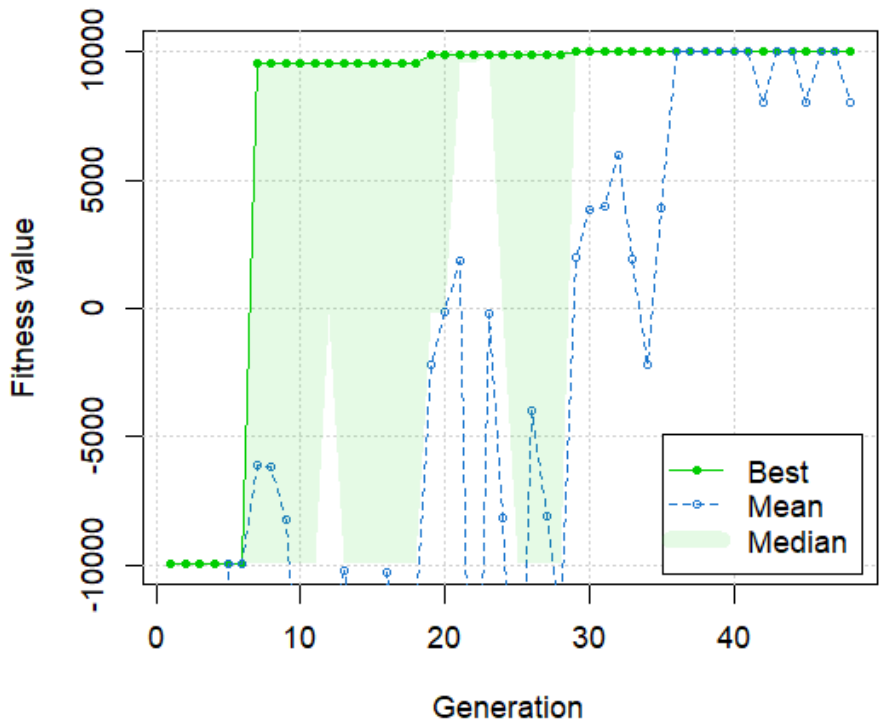
$$Vrni : Rezultat - |število| + K_1$$

1. pristop

Primer rešitve majhnega labirinta

Tak pristop se je izkazal za dobrega pri labirintih majhnih dimenzij: pri populaciji velikosti 10, že pri 7. iteraciji najde dobro rešitev do 30. iteracije pa optimalno



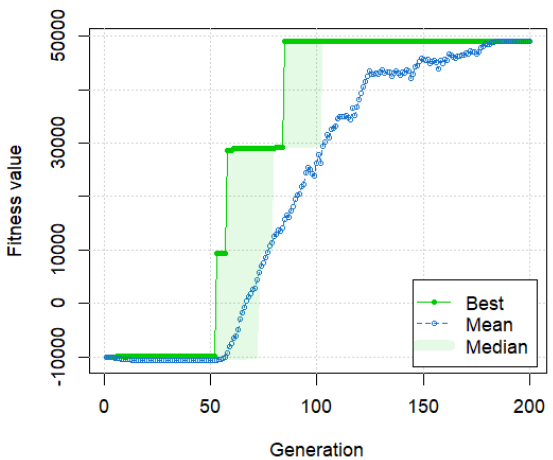


1. pristop

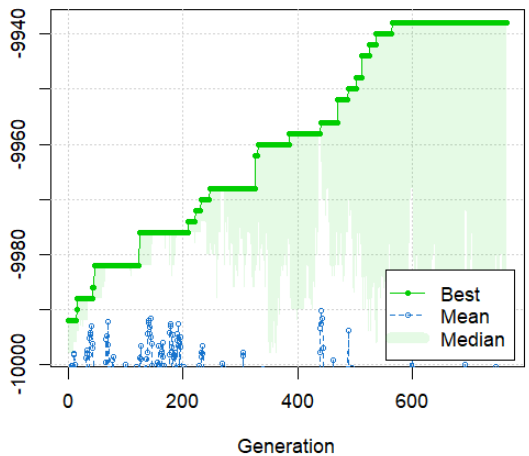
Mutacija

Implementirala sva tudi mutacijo, ki preveri, če se je osebek zaletel v zid, in če se je, na tisti točki popravi njegovo smer. Mutacija se je izkazala za uporabno predvsem na večjih labirintih z veliko zidovi:

```
# # # # # # # # # #
# S→ → → → → → → ↓ #
# # # # # # # # ↓ #
# ↓ ← ← ← ← ← ← ← #
# ↓ # # # # # # # #
# → → → → → → → ↓ #
# # # # # # # # ↓ #
# ↓ ← ← ← ← ← ← ← #
# ↓ # # # # # # # #
# E # # # # # # # #
```



S popravljeno mutacijo



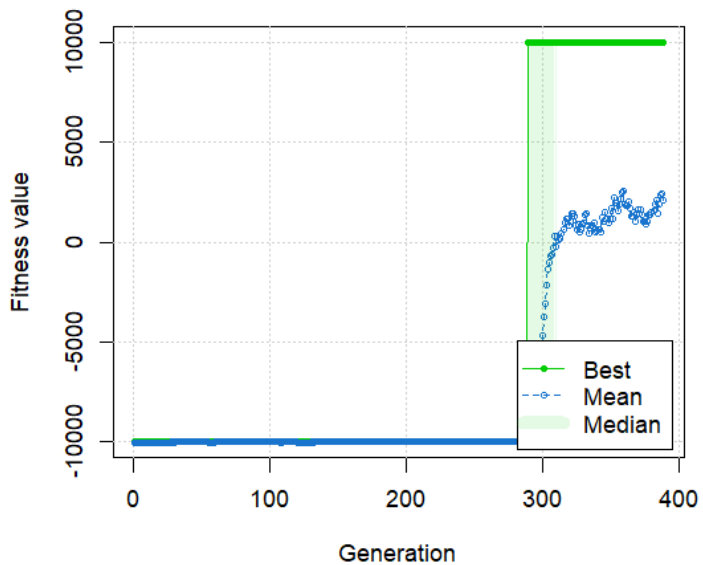
Brez popravljene mutacije

1. pristop

Primer rešitve večjega labirinta

Eden izmed večjih labirintov, ki jih je ta algoritem zmogel rešiti je 5. in sicer je pri populaciji velikosti 1000 poiskal rešitev okoli 300-tega koraka:

```
# # # E # # # # # # # # # # # #
# . . ↑ ← ← ← ← ← # # # . . #
# # # # # # # # ↑ . . . . # #
# . . . . . # # ↑ # # # . # #
# . # # # . . . ↑ ← # # . . #
# . . . . . # # # ↑ ← ← # # #
# # # # . # # # # . # ↑ # # #
# # . . . . . . # → → ↑ . . #
# # # # # # # . # ↑ # # # . #
# . . . . . . → → ↑ # # # # #
# . # # # # . S↑ # . . . . # #
# # . . . . # . # # # # # # #
# . . # # . . . . . . . . . #
# # . . . # # # . # # . # # #
# # # # # # # # # # # # # # #
```



1. pristop

Primer rešitve labirinta z zakladi

Iskanja zakladov sva se lotila da sva dala nagrado, ko je zaklad dobil in kazen, če je prišel na cilj preden je pobral vse zaklade.

$$Rezultat = Rezultat + K_1$$

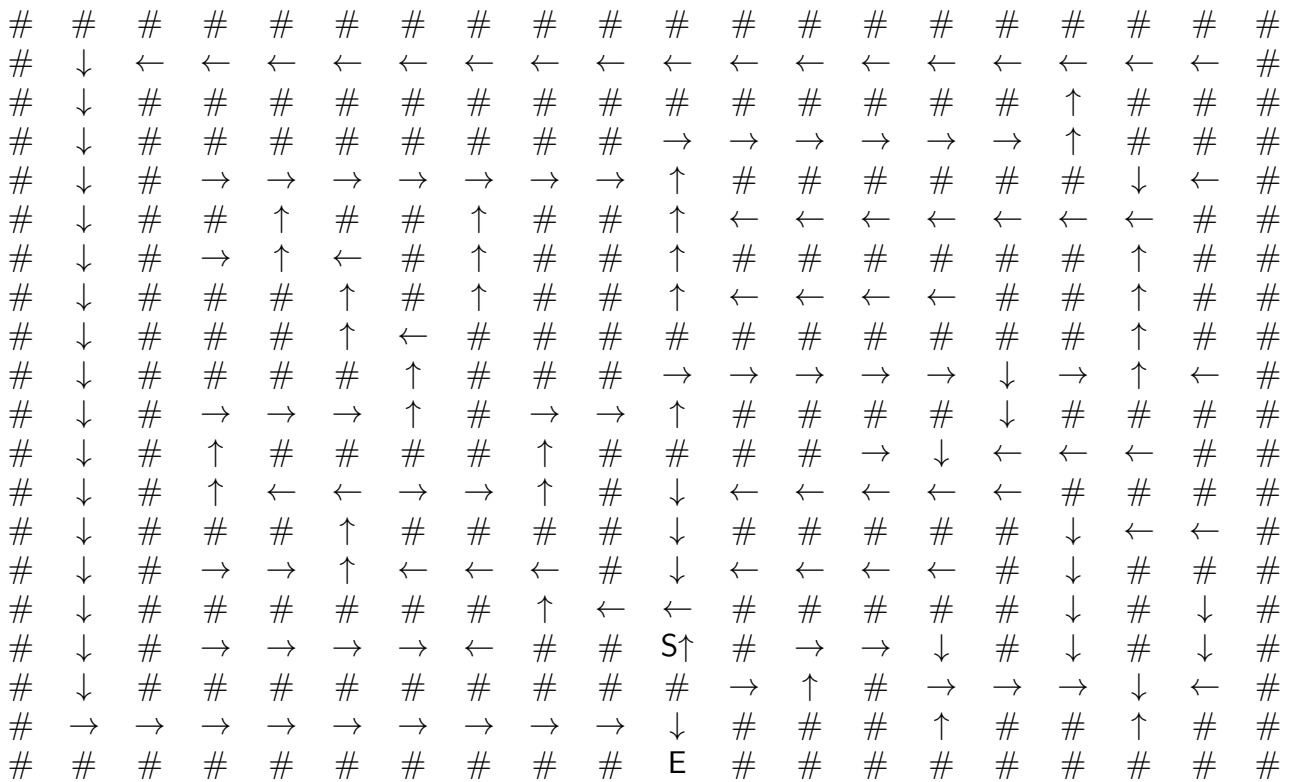
$$if\,|vsi| > |pobrani| : \;\; Rezultat = Rezultat - K_1$$

Pri populaciji velikosti 5000 je rešitev poiskal po 365-ti iteraciji.

#	#	#	#	#	E	#	#	#	#
#	.	.	.	#	$\uparrow_{27}$	$\leftarrow_{26}$	$\leftarrow_{25}$	$\leftarrow_{24}$	#
#	.	#	#	#	#	#	#	$\uparrow_{23}$	#
#	.	#	T $\rightarrow_{17}$	$\leftarrow_{16}$ $\rightarrow_{18}$	$\leftarrow_{15}$ $\rightarrow_{19}$	$\rightarrow_{20}$	$\rightarrow_{21}$	$\uparrow_{22}$	#
#	.	#	#	#	$\uparrow_{14}$	#	#	#	#
#	.	.	.	.	S $\downarrow_1\uparrow_{13}$	.	.	.	#
#	#	#	#	#	$\downarrow_2\uparrow_{12}$	#	#	.	#
#	.	.	#	#	$\downarrow_3\uparrow_{11}$	#	#	#	#
#	#	.	.	$\rightarrow_5$	$\leftarrow_4\rightarrow_6\uparrow_{10}$	T$\rightarrow_7\leftarrow_9$	$\leftarrow_8$	.	#
#	#	#	#	#	#	#	#	#	#

2. pristop

Pri prvem pristopu so vsi koraki odvisni od prejšnjih zato se nama crossover ni zdel smiseln. Ideja je bila, da bi lahko spustili več agentov istočasno na različnih mestih in nato prilagodili DNA. Rešitev je bila postaviti smerne vektorje na veljavna mesta labirinta, tako da če iz poljubne točke spustimo agenta bo sledil puščicam in prišel na cilj.



Vektorsko polje

Kodiranje

Osebek, ki je bil prej agent postane vektorsko polje. Na vsako veljavno mesto postavimo smerni vektor, ki pove v katero smer, se premakne agent, ko pride na to polje. Na polju je tudi cilj, v katerega poskušamo spraviti agente.

	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	1	0	0	0	0
2	0	3	5	7	0	9	11	13	15	0
3	0	17	0	0	0	0	0	0	19	0
4	0	21	0	23	25	27	29	31	33	0
5	0	35	0	0	0	37	0	0	0	0
6	0	39	41	43	45	47	49	51	53	0
7	0	0	0	0	0	55	0	0	57	0
8	0	59	61	0	0	63	0	0	0	0
9	0	0	65	67	69	71	0	73	75	0
10	0	0	0	0	0	0	0	0	0	0

Vektorsko polje

Fitness

1. pretvori binaren osebek v uporabno strukturo za delo
2. najdi pozicije, kjer puščica kaže proti zidu
3. najdi pozicije, kjer se zgodi protislovje (dve puščici kažeta ena proti drugi)
4. za štartno pozicijo, vsak zaklad in N naključnih veljavnih pozicij evaluiraj agenta, ki začne iz te pozicije. (Evaluacijo izvedem po fitness funkciji iz 1. pristopa)
   $O = \{start, z_1, z_2, \ldots, z_k, r_1, r_2, \ldots, r_N\}$

$$Agenti = \sum_{o \in O} eval(o)$$

$$Zid = K_1 * |Tocke\ proti\ zidu|$$

$$Proti = K_2 * |Protislovja|$$

$$Vrni = Agenti - Zid - Proti$$

(K1 = 1000, K2 = 1000)

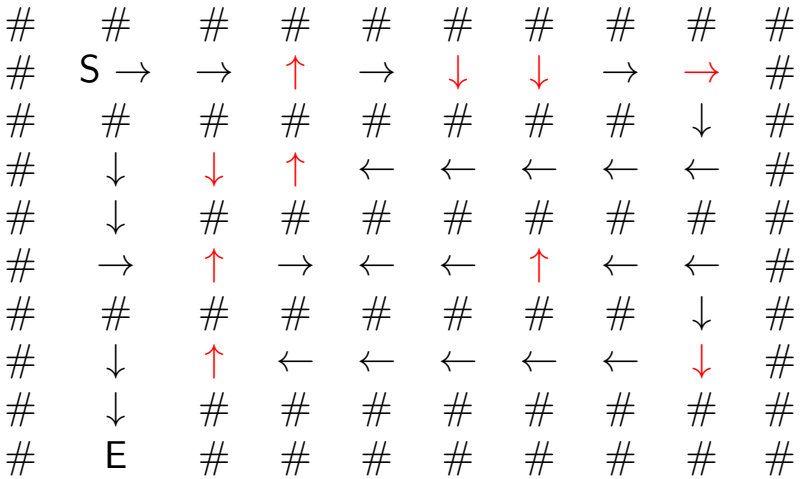
Mutacije

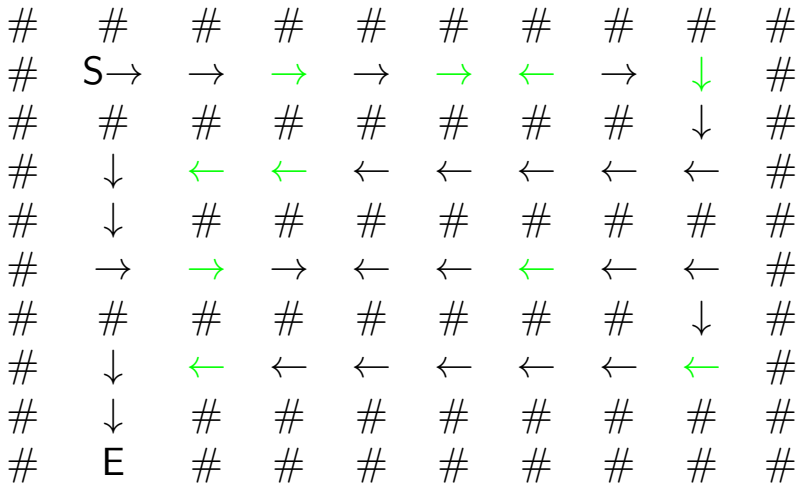
1. Z verjetnostjo P zamenjaj K bitov na naključnih mestih osebka
2. Pretvori binaren osebek v uporabno strukturo za delo
3. Če katera puščica kaže v zid, jo popravi
4. Če nobena puščica ne kaže v zid popravi eno protislovje

Mutacije

2.

1. Zaznaj vse puščice, ki kažejo v zid.
2. Za vsako puščico poišči vse pravilne smeri
3. Naključno izberi eno



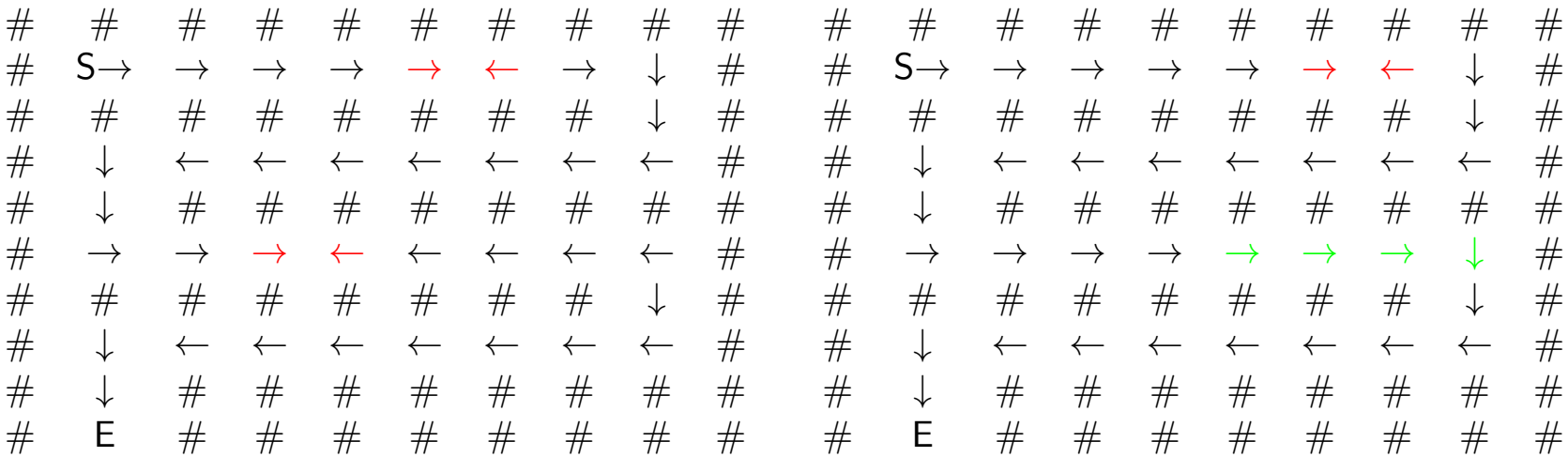


Mutacije

3.

Druga stvar, ki sva jo želela popraviti z mutacijami je bila, ko dve puščici kažeta druga proti drugi. To sva rešila tako, da sva izbrala eno od puščic in celo pot na tisti strani obrnila.

1. Zaznaj pare protislovij $P = P_1, P_2, \ldots, P_k$
2. Naključno izberi par $P_i = (p_{i1}, p_{i2})$
3. Naključno izberi enega izmed (p_{i1}, p_{i2})
4. Menjava orientacije za točko p_{ij} in iterativno točkam, ki vodijo v to točko



Mutacija

cikli

Včasih v polju nastane cikel, ki ne povzroča protislovij ter puščic v zid, ampak ko se agent vanj ujame ne pride do cilja. Rešitev je pri 1. koraku, torej da se poveča število bitov, ki se naključno spremenijo ter da se 1. korak izvaja bolj verjetno.

#	#	E	#	#	#
#	→	↑	←	←	#
#	↓	←	←	←	#
#	↓	↑	↓	↑	#
#	↓	↑	→	↑	#
#	→	→	↑	↑	#
#	#	#	#	#	#

Crossover

Crossover funkcije nisva spreminjala, pustila sva *gabin_uCrossover_R*, ki zamenja naključne istoležne bite.

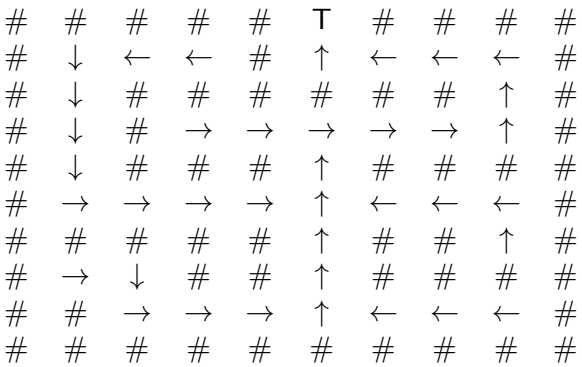
Zakladi

Problem pri rešitvi z vektorskim poljem je, da lahko gremo samo enkrat po eni smeri in imamo le en cilj.
Rešitev: Za vsak zaklad rešimo svoj labirint, in zaklad tretiramo kot cilj.
Dobimo $NR = |\{end, tr_1, tr_2, \ldots, tr_n\}|$ rešitev, ki jih moramo združiti v skupno pot. Nov problem bo iskanje permutacije zakladov, pri kateri je pot najkrajša.

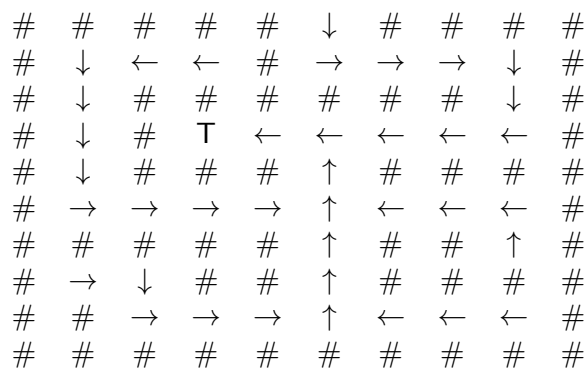
Zakladi

Primer rešitve labirinta z zakladi

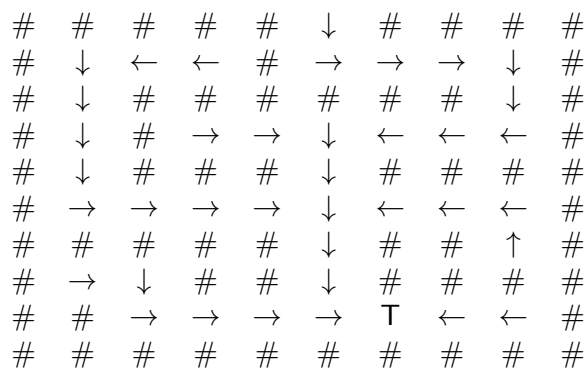
end



tr_1



tr_2



Združevanje rešitev

1. izračunamo število korakov od starta, do vsakega zaklada
2. izračunamo število korakov iz vsakega zaklada in cilja do vsakega zaklada in cilja
3. naredimo matriko DISTANCES, ki vsebuje razdalje med točkami
4. Matriko DISTANCES minimiziramo v MIN_DIST

	1	2	end	start
1	0	8	11	4
2	8	0	13	4
end	9	15	0	11

	1	2	end	start
1	0	8	9	4
2	8	0	13	4
end	9	13	0	11

Združevanje rešitev

Če je število zakladov majhno lahko izračunamo vse permutacije in za vsako seštejemo vse poti od zaklada do zaklada, ki ustreza permutaciji.

$$DIST(\sigma_i) = dist(start, \sigma_i(1)) + \sum_{j=2}^{n} dist(\sigma_i(j-1), \sigma_i(j)) + dist(\sigma_i(N), end)$$

Najboljša permutacija je tista, ki minimizira $DIST$. Če je število zakladov T veliko, je vseh možnih permutacij $T!$ lahko preveč. Takrat lahko uporabimo hevristično reševanje TSP z genetskimi algoritmi, ki smo spoznali na vajah.

Primer rešitve labirinta z zakladi