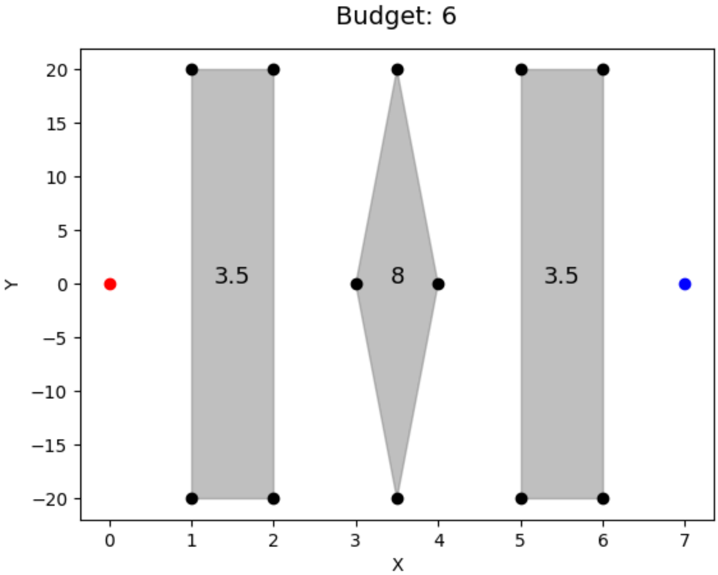
Najkrajša pot z odstranljivimi ovirami

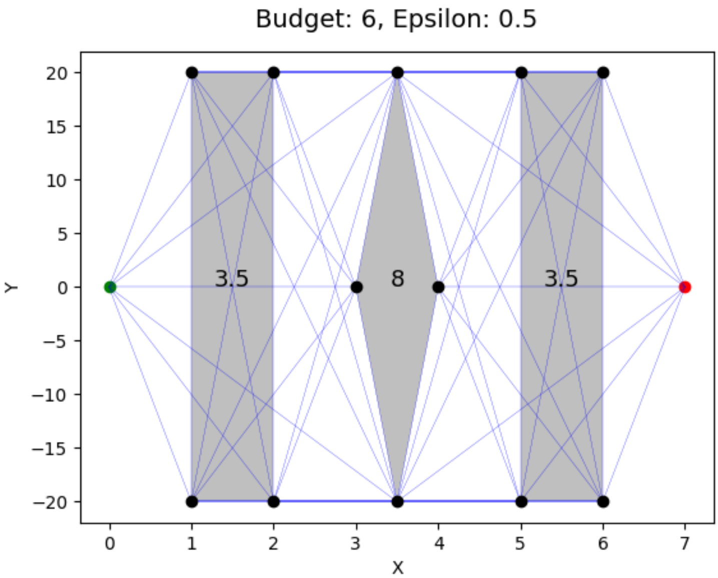
Gašper Terglav

18. junij 2024

Primer problema



Viability graf

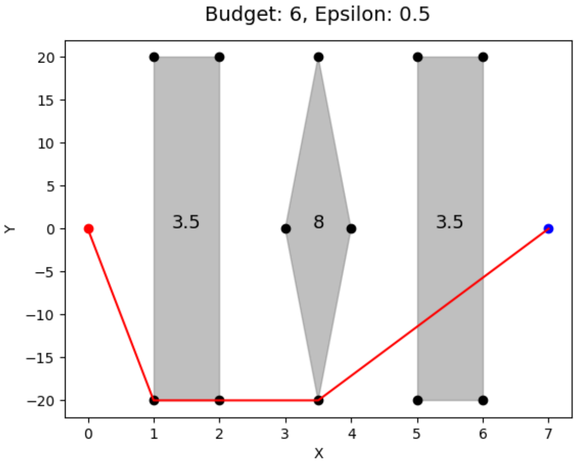


Rešitev

Izberemo parameter natančnosti ϵ. Če ima viabilty graf vozlišča $v \in V$, potem ima nov graf vozlišča oblike $v_i \in V$, kjer $i = 0, \epsilon, 2\epsilon, \ldots, \lceil budget/\epsilon \rceil$.

Rešitev

Izberemo parameter natančnosti ϵ. Če ima viabilty graf vozlišča $v \in V$, potem ima nov graf vozlišča oblike $v_i \in V$, kjer $i = 0, \epsilon, 2\epsilon, \ldots, \lceil budget/\epsilon \rceil$.



Naivni algoritem

Za vsak par točk v, u in vsak rob ovire e, pogledam,če $\overline{vu}$ seka e. Če ja, dodam ceno ovire e k ceni $\overline{vu}$. V graf dodam vse daljice s ceno manj od budgeta.

Naivni algoritem

Za vsak par točk v, u in vsak rob ovire e, pogledam,če $\overline{vu}$ seka e. Če ja, dodam ceno ovire e k ceni $\overline{vu}$. V graf dodam vse daljice s ceno manj od budgeta. Časovna zahtevnost $O(n^3)$. Zahtevnost celotnega algoritma je potem $O(n^3/\epsilon)$

Za vsak par točk v, u in vsak rob ovire e, pogledam,če $\overline{vu}$ seka e. Če ja, dodam ceno ovire e k ceni $\overline{vu}$. V graf dodam vse daljice s ceno manj od budgeta. Časovna zahtevnost $O(n^3)$. Zahtevnost celotnega algoritma je potem $O(n^3/\epsilon)$

	n (število oglišč)			
	40	180	360	860
Čas	0.5s	36s	301s	1ura 16min

Tabela: Vpliv n na čas iskanja poti

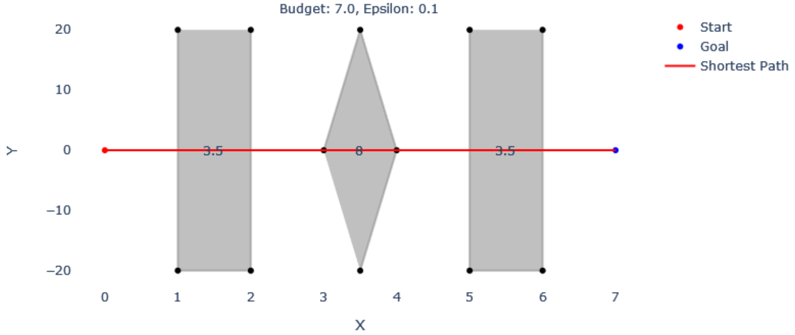
Naivni algoritem

	ϵ			
	0.01	10^{-3}	10^{-4}	10^{-5}
Čas	0.5s	5s	57s	memory full

Tabela: Vpliv vrednosti ϵ na čas iskanja poti

Naivni algoritem

Problem:

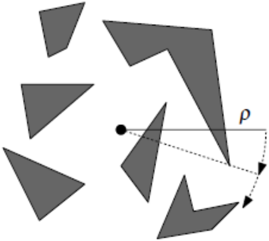


Sweep algoritem

Za vsako točko v, ostale točke uredim glede na kot z vodoravno premico in jih nato pregledam po vrsti. Robove ovir shranjujem v

AVL drevo.

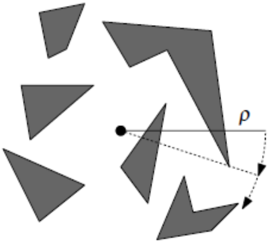
Za vsako točko v, ostale točke uredim glede na kot z vodoravno premico in jih nato pregledam po vrsti. Robove ovir shranjujem v



AVL drevo.

Sweep algoritem

Za vsako točko v, ostale točke uredim glede na kot z vodoravno premico in jih nato pregledam po vrsti. Robove ovir shranjujem v



AVL drevo. Časovna zahtevnost $O(n^3)$. Zahtevnost celotnega algoritma je potem $O(n^3/\epsilon)$

Sweep algoritem

	n (število oglišč)			
	40	180	360	860
Čas	0.7s	50s	393s	1ura 28min

Tabela: Vpliv n na čas iskanja poti

	ϵ			
	0.01	10^{-3}	10^{-4}	10^{-5}
Čas	1s	11s	126s	memory full

Tabela: Vpliv vrednosti ϵ na čas iskanja poti

Sparse algoritem

- Določimo navpično premico p, ki razdeli točke na dva množici približno enake moči. Za vsako točko v je v' njena projekcija na p.

Sparse algoritem

- Določimo navpično premico p, ki razdeli točke na dva množici približno enake moči. Za vsako točko v je v' njena projekcija na p.
- Poiščemo prvi rob ovire z navpičnim naklonom, ki seka $\overline{vv'}$. Označimo presečišče z x. Če obstaja presečišče roba s p (označimo ga z y). Dodamo v graf $\overline{xy}$.

Sparse algoritem

- Določimo navpično premico p, ki razdeli točke na dva množici približno enake moči. Za vsako točko v je v' njena projekcija na p.
- Poiščemo prvi rob ovire z navpičnim naklonom, ki seka $\overline{vv'}$. Označimo presečišče z x. Če obstaja presečišče roba s p (označimo ga z y). Dodamo v graf $\overline{xy}$.
- Ponovimo za prvi negativen rob.

Sparse algoritem

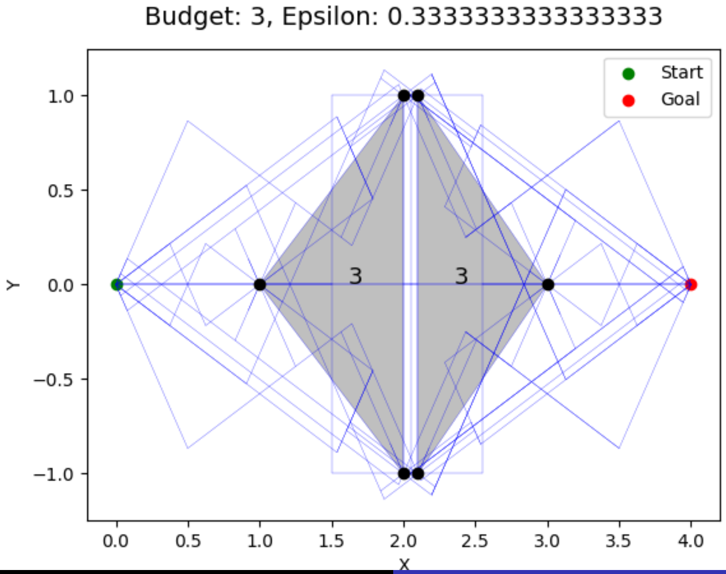
- Določimo navpično premico p, ki razdeli točke na dva množici približno enake moči. Za vsako točko v je v' njena projekcija na p.
- Poiščemo prvi rob ovire z navpičnim naklonom, ki seka $\overline{vv'}$. Označimo presečišče z x. Če obstaja presečišče roba s p (označimo ga z y). Dodamo v graf $\overline{xy}$.
- Ponovimo za prvi negativen rob.
- Rekurzivno ponovimo na levi in desni strani p.

Sparse algoritem

- Določimo navpično premico p, ki razdeli točke na dva množici približno enake moči. Za vsako točko v je v' njena projekcija na p.
- Poiščemo prvi rob ovire z navpičnim naklonom, ki seka $\overline{vv'}$. Označimo presečišče z x. Če obstaja presečišče roba s p (označimo ga z y). Dodamo v graf $\overline{xy}$.
- Ponovimo za prvi negativen rob.
- Rekurzivno ponovimo na levi in desni strani p.
- Ponovimo vse do sedaj $\lceil 1/\epsilon \rceil$-krat, le da vsakič zarotiramo ravnino za kot $2\pi\epsilon$.

Sparse algoritem

Primer:



Sparse algoritem