Najkrajša pot z odstranljivimi ovirami – kratko poročilo

Gašper Terglav

5. April 2024

Delo do sedaj

Predvsem sem bral članek in ga poskušal razumeti. Potem sem še na hitro napisal Dijkstra algoritem. Posledično je večina poročila samo predstavitev problema, nakoncu pa je še nek okviren načrt.

Predstavitev problema

Problem je posplošitev klasične verzije, v kateri se lahko oviram samo izogibamo. V $\mathbb{R}^2$ imamo dve točki s in t iščemo najkrajšo evklidsko pot med njima. Ovire v ravnini so konveksni večkotniki, vsak od njih ima ceno $c_i > 0$. Če imamo na voljo C "denarja", katere ovire se splača odstraniti, da dosežemo najkrajšo pot? V resničem življenju lahko npr. načrtovalci mest spremenijo cestno omrežje, da dosežejo boljšo pretočnost in tako odstranijo ovire za neko ceno. Še en primer omenjen v članku so skladišča v katerih delajo roboti. Kako spremeniti postavitev ovir v skladišču, da se roboti hitreje premikajo okoli? Problem je NP-težek (v članku je opisana redukcija problema PARTITION na naš problem), mi pa se bomo osredotočili na polinomske algoritme z napako.

Algoritem v polinomskem času z napako $(1 + \epsilon)$ v ceni

Začnimo s pomembnima opazkama o obnašanju najkrajše poti. Če bo šla naša pot skozi oviro, bo zaradi konveksnosti sekala samo dve stranici večkotnika. Smer bo pot spremenila samo v ogliščih večkotnikov. Tako bomo lahko problem prevedli na iskanje najkrajše poti v grafu, katerega vozlišča so vsa oglišča ovir ter točki s in t. Označimo ta graf z $G = (V, E)$, kjer $(u, v) \in E$, če je vsota cen vseh ovir, ki jih prečka daljica uv manjša ali enaka od našega proračuna C. Vsaka povezava v grafu ima dva parametra, dolžino in ceno. Ta graf bi radi modificirali tako, da bi na novem grafu samo pognali Dijkstra algoritem in dobili rešitev, zato hočemo odstraniti parameter cene iz povezav. Naj bo $K = min(\frac{C}{min_i\ c_i}, h)$, kjer je h število ovir, c_i pa njihove cene ter $\sum c_i = C$. Delimo vse cene z K/C, da je naš proračun za ovire na tako enak K. Potem naredimo $\lceil \frac{2K}{\epsilon} \rceil + 1$ kopij

vsakega vozličša v grafu G in jih označimo z $v_0, v_{\epsilon/2}, v_{\epsilon}..., v_K$. Za vsak rob $(u, v) \in E$ s ceno c in za vsak $0 \leq i \leq \lceil \frac{2K}{\epsilon} \rceil$, dodamo rob $(u_{i\epsilon/2}, v_{j\epsilon/2})$, kjer je $j \leq \lceil \frac{2K}{\epsilon} \rceil$ največje celo število da velja $j\frac{\epsilon}{2} \leq i\frac{\epsilon}{2} + c$. Dolžina povezav je enaka kot v originalnem grafu. Tako smo ceno zakodirali v vozlišča. Če se nahajamo v vozlišču v_i to pomeni, da smo za odstranitev ovir do sedaj plačali i približno denarja. Na koncu dodamo še novi vozlišči s in t, ter vsakega od njiju povežemo z vsemi njunimi kopijami. Te povezave imajo dolžino 0. Ker imajo povezave samo še dolžino, lahko sedaj za iskanje najkrajše poti uporabimo Dijkstro. Novi graf ima $O(|V|K/\epsilon)$ vozlišč in $O(|E|K/\epsilon)$ robov. Ker je Dijkstra najbolj časovno zahtevna stvar, je potem časovna zahtevnost celotnega algoritma enaka $O(\frac{K}{\epsilon}(|E| + |V| \log \frac{|V|}{\epsilon}))$. Če je n število vseh oglišč ovir, je to v najboljšem primeru $\Omega(\frac{n^3}{\epsilon})$.

Hitrejši algoritem

Ta algoritem je bolj zapleten in doseže večjo hitrost z zmanjšanjem povezav v grafu G. V zameno pa je tukaj lahko napaka $(1 + \epsilon)$ tudi pri dolžini poti. Ozaničmo nov graf z manj povezavami s $H = (X, \Gamma)$. Zanimivo je da ima H več vozlišč kot G, bolj natančno $|X|, |\Gamma| = O(n \log n)$. Ko enkrat skonstruiramo H pa lahko spet uporabimo idejo iz prejšnjega algoritma in naredimo $O(1/\epsilon)$ kopij ter uporabimo Dijkstro. Ta algoritem ima časovno zahtevnost $O(\frac{nh}{\epsilon^2} \log n \log \frac{n}{\epsilon})$ kar je približno za potenco n manj kot prejšnji algoritem.

Okviren načrt dela

Najprej nameravam v pythonu (ker imam z njim največ izkušenj) implementirati prvi algoritem za iskanje poti. To iskanje bi tudi grafično prikazal, za enkrat je ideja s pomočjo *matplotlib*, mogoče namesto tega spletni vmesnik. Naredil bom nekaj testnih primerov in opazoval kako se algoritem obnaša za različne ϵ. Kasneje bom poskusil z implementacijo drugega algoritma in podatkovne strukture, ki omogoča hitrejše poizvedbe.

Vir

[1] Pankaj K. Agarwal, Neeraj Kumar, Stavros Sintos, and Subhash Suri. *Computing Shortest Paths in the Plane with Removable Obstacles.* In 16th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2018). Leibniz International Proceedings in Informatics (LIPIcs), Volume 101, pp. 5:1-5:15, Schloss Dagstuhl – Leibniz-Zentrum für Informatik (2018)