Iosif Solntsev

Zadanie č. 1c Taby search

Problém obchodného cestujúceho

Úloha: Cestujúci obchodník potrebuje navštíviť viacero miest a je dôležité minimalizovať cestovné náklady, ktoré závisia od dĺžky trasy. Musí nájsť najkratšiu cestu, aby každé mesto navštívil iba raz. Keďže sa na konci cesty musíte vrátiť do východiskového mesta, jeho trasa musí byť uzavretá.

Máme od 20 do 40 miest, reprezentovaných súradnicami X a Y, sú náhodne generované, veľkosť našej mapy je 200 * 200. Cena cesty je určená euklidovskou vzdialenosťou. Naším cieľom je nájsť optimálne a najlacnejšie poradie miest.

Algoritmus:

Na nájdenie optimálneho riešenia používame algoritmus tabu search, ktorý je zameraný na vyhýbanie sa slučkám a lokálnym minimám pomocou zoznamu zakázaných stavov na dočasné zamedzenie návratu k nedávno zvažovaným riešeniam.

Prezentácia údajov

Mestá sú reprezentované ako zoznam náhodných súradníc v dvojrozmernej rovine, kde každý bod na rovine je opísaný svojimi súradnicami X a Y:
Trasa je definovaná ako zoznam indexov reprezentujúcich poradie, v ktorom sa prechádza cez mestá:

Popis algoritmu

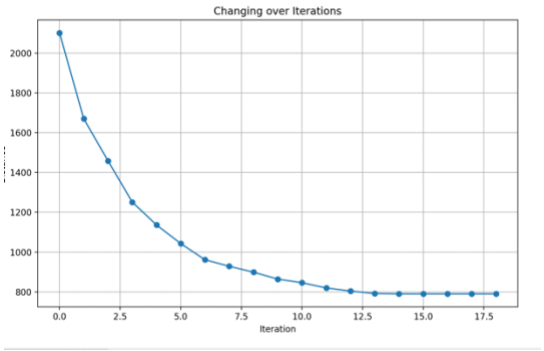
1)Generovanie počiatočného riešenia:
Spočiatku sa trasa vyberie náhodne premiešaním zoznamu miest.
2)Generácia susedov:
Pri každej iterácii sa vygenerujú všetci možní susedia pre aktuálnu trasu výmenou dvoch náhodných miest na trase. To umožňuje algoritmu preskúmať blízke riešenia.
3)Zoznam tabu:
Algoritmus pridáva nedávno skontrolované riešenia do zoznamu tabu. Tým sa zabráni návratu k trasám, ktoré už boli zvážené, a zabráni sa zacykleniu.
Veľkosť zoznamu tabu je obmedzená a staré riešenia sa odstraňujú, keď sa pridávajú nové.
4)Hodnotenie riešenia:
pre každú susednú trasu sa vypočíta jej dĺžka (celková vzdialenosť). Pri každej iterácii sa vyberie sused s najkratšou dĺžkou, ktorý nie je v tabuizovanom zozname.
5) Aktualizácia najlepšieho riešenia:

Ak je nájdený sused lepší ako aktuálne najlepšie riešenie, stane sa novou optimálnou trasou.
Iterácie pokračujú, kým sa výsledok nestabilizuje – keď sa dĺžka najlepšej trasy počas niekoľkých iterácií nemení.

Veľkosť zoznamu tabu priamo ovplyvňuje výkon algoritmu. Väčší zoznam tabu pomáha vyhnúť sa návratu k predtým preskúmaným riešeniam, ale môže spomaliť proces hľadania nových riešení.

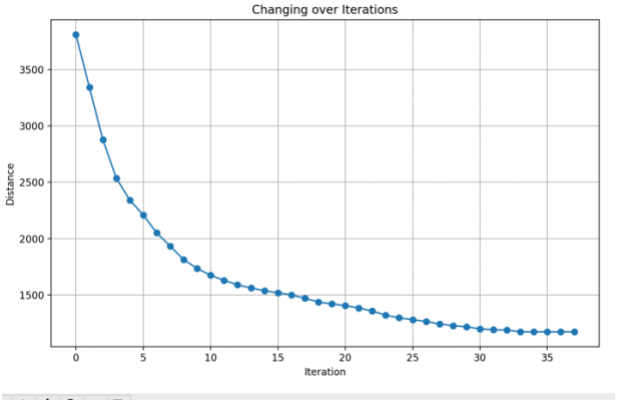
Testy s rôznym počtom miest:

Počet miest: 20
Pre 20 miest s tabu zoznamom dĺžky 40 sa našlo optimálne riešenie v niekoľkých desiatkach iterácií. Najlepšia trasa má dĺžku blízkou optimálnej a zmeny sa zastavia po 15-20 iteráciách.
Iterácie až do stabilizácie: ~20
Najlepšia trasa: ~700-800 km



Počet miest: 30
Pre 30 miest s tabu zoznamom dlzky 70 proces trval viac iterácií, ale konečné riešenie je tiež takmer optimálne. Dlhšia trasa zväčšuje priestor na vyhľadávanie, čo si vyžaduje viac času na stabilizáciu.
Iterácie až do stabilizácie: ~40

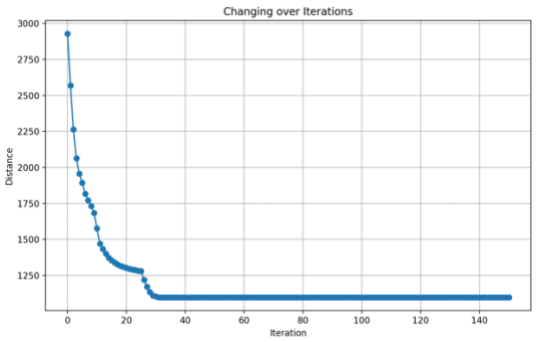
Najlepšia trasa: ~1000-1200 km



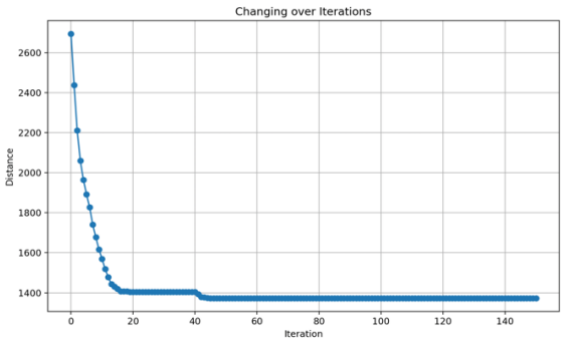
Vplyv dĺžky tabuizovanej listu

S malým tabu zoznamom (napríklad 10 prvkov) sa algoritmus môže zacykliť, pretože sa nemôže účinne vyhnúť návratu k nedávnym rozhodnutiam. Zväčšenie dĺžky zoznamu tabu (napríklad na 30 prvkov) zlepšuje kvalitu riešení rozšírením vyhľadávacieho priestoru, ale proces stabilizácie sa spomalí a algoritmus môže preskúmať viac riešení bez výrazného pokroku.

V tomto prípade dĺžka zoznamu tabu 70 veľmi neovplyvňuje rýchlosť vykonávania, ale nevytvára problém s opakovaním, takže toto je vhodná dĺžka pre náš zoznam. Zoznam 70 prvkov ukazuje o niečo lepší výsledok na rovnakých údajoch ako zoznam 30



dlzka 70



dlzka 30

Záver a zlepšenie:

Naše riešenie vo všeobecnosti vykazuje dobrú efektivitu a ukazuje nám výsledok blízky optimálnemu.

Pre vylepšenie programu by som navrhol zamyslieť sa nad podmienkami zastavenia programu, pretože môj spôsob s výstupom podmieneným zastavením zmien dĺžky najlepšej trasy asi nie je optimálny.
Môžete tiež pokračovať v experimentovaní s hľadaním ideálnej dĺžky zoznamu tabu.

Na spustenie tohto kódu si budete musieť stiahnuť knižnicu matplotlib a itertools