Ant Colony Optimization för Travelling Salesman Problem

Teknikhögskolan

December 2021

1 Uppgiftsbeskrivning

Första deluppgiften kretsar kring kontinuerlig optimering. Givet en viss funktion söker vi max- respektive minimumvärde och koordinaterna för dessa. Applikationsområden för är bland annat livscykelsanalys och kostnad, signalbehandling, biomedicin, beräkningar inom elektromagnetism m.m..

Andra deluppgiften kretsar kring Travelling Salesman Problem, vilket handlar om att i en sammanhängande graf med kopplingar mellan varje nod besöka varje nod med så liten total distans som möjligt. Alla noder kan alltså nås direkt från vilken nod som helst och ingen nod får besökas mer än en gång.

Problemet är inte bara av intresse för turister och resenärer, utan har även applikationer inom industri (t.ex. optimering av lödning av kretskort eller annan processering av komponenter) och logistik (t.ex. leveransoptimering).

Målet i inlämningsuppgiften är att med Ant Colony Optimization ge förslag på en rutt mellan 120 städer som är så kort som möjlig. Detta ger oss 120! möjligheter (runt 10^{198}, ungefär en noniljards unvigintiljarder), så vi får vara lite fiffiga istället.

OBS! En perfekt lösning är mycket osannolik att påträffas.

2 För G:

2.1 Kontinuerlig optimering

Hitta koordinater och funktionsvärde för högsta värde i

$$f(x,y) = \frac{1}{1+|x|+|y|}(cos(x)+sin(y))^2 \tag{1}$$

där $x \in [-5,5], y \in [-5,5]$. Lösning gjord för hand med bevis för globalt optimum godtas också.

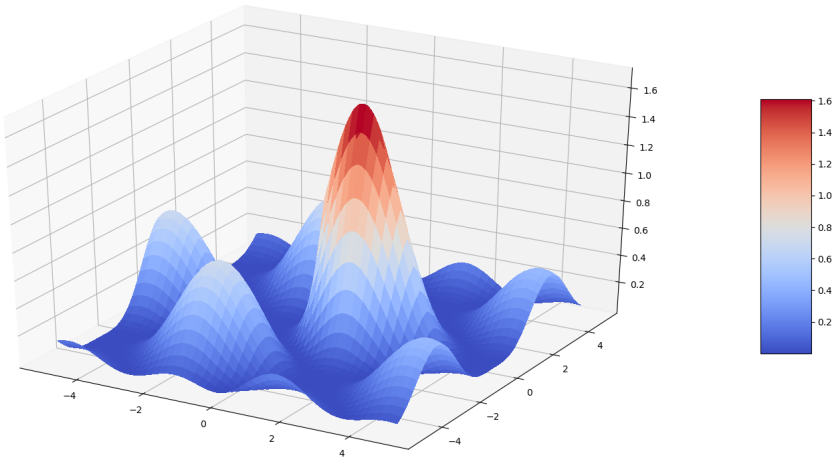

Figure 1: a nice plot

2.2 TSP - Baseline

Utgå från given csv-fil

Läs in csv-filen som innehåller avstånd mellan varje par av städer och sätt upp ett testscenario där du börjar i en slumpad stad och därefter i varje stad slumpar nästa stad att besöka tills alla städer är besökta. Gör detta test 10000 gånger och spara den totala distansen för varje simulering. Visualisera distributionen av distanser för varje väg du slumpat fram med t.ex. ett histogram från matplotlib eller seaborn.

2.3 TSP - Ant Colony Optimization

Läs in csv-filen och implementera Ant Colony Optimization från scikit-opt Jämför ditt resultat med ditt bästa resultat från dina slumpade simuleringar. Utred för parametrar $max\_iter = [50, 100, 150]$ och $size\_pop = 50$.

3 För VG:

3.1 Kontinuerlig optimering

Hitta koordinater och funktionsvärde för högsta respektive lägsta för värde i kontinuerliga funktionen

$$f(x,y) = e^{-0.05(x^2+y^2)}(arctan(x) - arctan(y) + e^{-(x^2+y^2)}cos^2(x)sin^2(y)) \quad (2)$$

där $x \in [-5,5], y \in [-5,5]$. Lösning gjord för hand med bevis för globalt optimum godtas också.

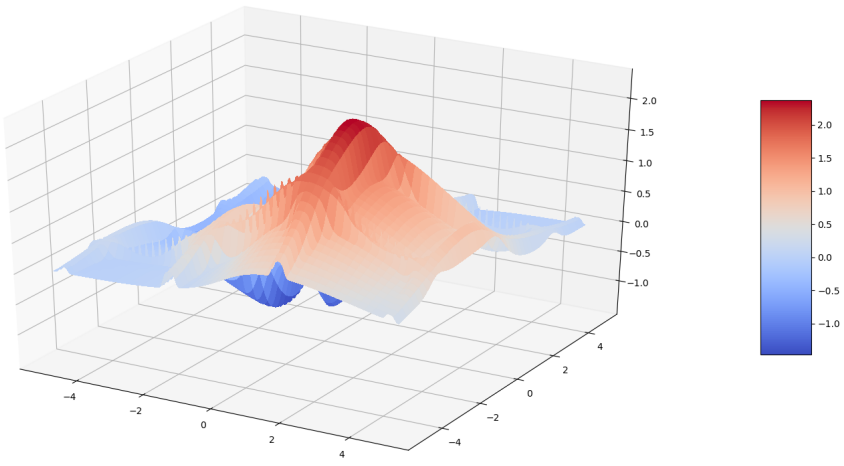

Figure 2: Kisar ni ser ni att det är många ”små toppar” (många lokala minima). Försvårar till icke-heuristiska metoder som t.ex. gradient descent.

3.2 TSP - Datainsamling och preprocessing

Skapa din egen csv-fil likt den bifogade innehållandes avstånd mellan varje par av städer baserat på tabellerna för deras latituder och longituder i den här länken.

Ett par av latituder och longituder ger oss möjlighet att approximera avståndet mellan två punkter på jorden. Länk för beräkning av distans mellan två punkter på jorden

Länk för Latitud/Longitud till grader

Formlerna antar en perfekt sfärisk jord, vilket ger oss en liten felmarginal som vi kan bortse ifrån. Den är ju trots allt platt.

Stäm av dina slutgiltiga resultat mot csv/xlsx-filen innan du fortsätter!

3.3 Baseline 1

Använd datan för distanser mellan alla städer du har sammanställt och utför samma slumpade baseline som i G-uppgiften.

3.4 Baseline 2

Implementera en "greedy" algorithm som ståendes i varje stad besöker den närmsta staden som ännu inte besökts. Spara den totala distansen för startpunkt i varje stad. Skiljer sig resultaten åt för olika startpunkter?

3.5 Baseline 3

Använd multiprocessing för att starta lika många körningar av Genetic Algorithm TSP (GA_TSP) från scikit-opt som du har cores. Mät tiden det tar och jämför med att köra lika många runs utan multiprocessing. Testa för alla kombinationer av $max\_iter = [100, 500, 1000]$ och $prob\_mut = [0.001, 0.01, 0.05]$.

Får du konsekvent sämre resultat från multiprocessingen kan du behöva använda time.sleep() i processerna.

3.6 Ant Colony Optimization

Testa slutligen ant colony optimization för problemet. Testa för $max\_iter = [100250, 500]$ och $size\_pop = 50$. Slår du dina baselines?