Smidigaste vägen mellan två städer

Programmeringsprojekt SF1672

Samuel Kostic, Harriet Källberg, Mehrab Norouzi,
Patrik Oksanen och Lukas Sjögren

BAKGRUND

Hur tar en sig smidigast mellan två platser? Den frågan ställer sig de flesta i sin vardag och svaret beror självklart på vem som ska resa. Detta eftersom faktorer som ålder, antal resenärer och huruvida passagerarna har bråttom eller inte, påverkar vilken väg som anses vara smidigast.

Ponera att ett antal resenärer söker den smidigaste vägen från en punkt A till en punkt B. På hur många sätt kan detta göras, och vilket sätt är smidigast? För att hitta svaret på den första frågan kommer detta projekt utnyttja den linjära algebrans potenser av oviktade anslutningsmatriser. För att hitta svaret på den andra frågan kommer Dijkstras algoritm att appliceras. Inom grafteorin har nämligen en liknande fråga redan ställts. Denna brukar kallas “The Traveling Salesman Problem” (TSP), och Dijkstras algoritm är utformad för att lösa den. Algoritmen kommer ta oss från A till B genom den väg som är kortast eller ”kostar” minst. Reslutatet som erhålls är dock beroende av huruvida lösningsmetoden appliceras på en oviktad eller viktad graf. Genom oviktade grafer kan Dijkstras ta fram den väg som innebär färst byten, och genom en viktad graft erhålls istället den väg som tar kortast tid i varje steg. Det inses lätt att metodvalet här skulle påverka vilka resenärer som blir nöjda med sin resa.

PROBLEMFORMULERING

Hur kan då en dator hitta den bästa färdvägen utefter passagerarnas behov? Detta projekt kommer rangordna två olika sätt att hitta den smidigaste vägen mellan 2 av 14 svenska städer beroende på egenskaper hos passagerarna. Som grund till arbetet ligger den svenska järnvägen mellan 14 städer i centrala och södra Sverige, samt ett antal resenärattribut. Attributen begränsas till: hög ålder, har barn, bagage, sömnbrist, jobb, skola och event. I den första metoden kommer den väg med minst antal byten att väljas. I den andra kommer en viktad graf styra färden. Då väljs den väg som är stegvis kortast.

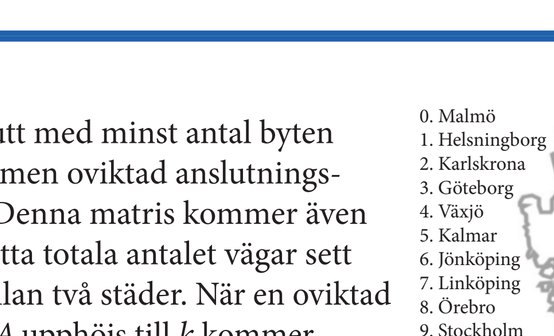
Informationen om passagernas attribut kommer ligga till grund för en jämförelse av den objektivt kortaste resan gentemot den färd med färst byten. Den som bäst överrensstämmer med resenärernas attribut kommer väljas. Med hjälp av matrispotenser fastställs även det totala antalet möjliga färdvägar med ett visst antal byten. Resultatet förväntas bli den rekommenderade resa som passar resenärerna bäst samt totala antalet alternativ som finns för resan.

LÖSNING

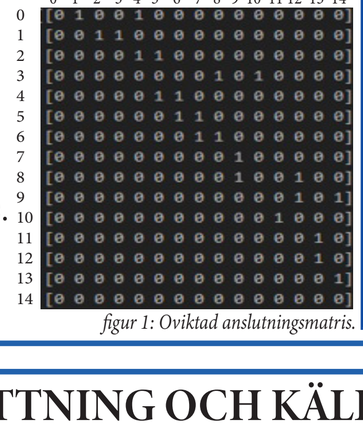
För att hitta den rutt med minst antal byten används en riktad men oviktad anslutnings-matris, se figur 1. Denna matris kommer även användas för att hitta totala antalet vägar sett till antal byten mellan två städer. När en oviktad anslutningsmatris A upphöjs till k kommer nämligen värdet på plats (i, j) motsvara det antal sätt en resenär kan ta sig från i till j med k steg.

För att hitta den stegvis kortaste rutten används en riktad och viktad anslutningsmatris vars graf illustreras i figur 2. Figur 3 visar Dijkstras algoritm så som den implementerats i projektet.

Passagerarattributen delas in i tid-stress-attribut som vill minimera ruttens tid (jobb, skola, event) och nod-stress-attribut som vill minimera antal byten (hög ålder, har barn, bagage, sömnbrist).



figur 2: Viktad och riktad graf.



figur 1: Oviktad anslutningsmatris.

```
class dijkstra:
    def __init__(self, graph):
”””intitierar djikstra algoritmen”””

    def get_nodes(self, graph):
”””Skapar alla städer i klassen,
kommer senare initerera över dessa”””

    def get_edges(self, graph, mode):
”””Beräknar vägarna mellan städerna,
beroende på mode valt så implementeras
weighted eller inte”””

    def pathfinder(self, opening, mode):
”””Hittar den kortaste vägen med avseede
på vilken metod man har valt”””

    def n_nodes():
”””Räknar hur många städer som passe-
rats.”””
```

figur 3: Dijkstras algoritm som klass

RESULTAT

SCENARIO 1:
Familj med fyra resenärer, två förädrar med ålder över 50 år och två barn.

familjens attribut är: ålder, bagage, barn och event
Förälder 1: Över 50 år. Har bagage. Har barn.
Förälder 2: Över 50 år. Har bagage. Har barn.
Barn 1: Har bagage. Ska på event.
Barn 2: Har bagage. Ska på event.

tid-stress-attribut:

	F1	F2	B1	B2
Jobb	0	0	0	0
Skola	0	0	0	0
Event	0	0	1	1

nod-stess-attribut:

	F1	F2	B1	B2
Hög ålder	1	1	0	0
Barn	1	1	0	0
Bagage	1	1	1	1
Sömnbrist	0	0	0	0

Familjen ska från Malmö till Gävle. Med följande information kommer de rekommenderas att ta en långsammare rutt med färre byten eftersom de har mycket baggage, äldre passage-rare och barn. Att barnen ska på event blir sekundärt.

Utskrift:

```
Calculating recommendation:
Take the slower one with less swaps
Number of stops:  5
Your path:  ['vaxjo', 'kalmar', 'linkoping', 'stockholm', 'gavle']
```

SCENARIO 2:
Grupp med tre ungdomar på väg till konsert, samtliga med ålder under 30 år.

Gruppens attribut är: ålder, bagage, sömn och event
Ungdom 1: Har bagage. Dålig sömn. Ska på event.
Ungdom 2: Har bagage. Bra sömn. Ska på event.
Ungdom 3: Har bagage. Bra sömn. Ska på event.

tid-stress-attribut:

	U1	U2	U3
Jobb	0	0	0
Skola	0	0	0
Event	1	1	1

nod-stess-attribut:

	U1	U2	U1
Hög ålder	0	0	0
Barn	0	0	0
Bagage	1	1	1
Sömnbrist	1	0	0

Gruppen ska från Malmö till Gävle. Med följande information kommer de rekommenderas att ta en snabbare rutt med fler byten eftersom de alla är under 30 år och har ett event de ska till. Att de har baggage och att en ungdom är trött blir sekundärt.

Utskrift:

```
Calculating recommendation:
Take the quickest one with more stops
Number of stops:  6
Your path:  ['helsingborg', 'goteborg', 'karlstad', 'stalldalen', 'borlange', 'gavle']
```

Antalet möjliga vägar från Malmö till Gävle:

Antal byten:	1	2	3	4	5	6	7	8	9	10
Antal vägar:	0	0	0	0	4	6	10	7	6	2

Totalt antal vägar: 35

SAMMANFATTNING OCH KÄLLOR

Sammanfattningsvis har detta projekt svarat på frågan hur en resenär tar sig smidigast mellan två svenska städer via tåg. Inledningsvis utnyttjades den linjära algebrans anslutningmatriser och Dijkstras algoritm för att fastställa den rutt med lägst antal byten mellan en stad A och B i södra Sverige. Vidare viktades anslutningsmatrisen i proportion mot det geografiska avståndet mellans städerna. Detta medförde att Dijkstras algoritm nu valde den väg som är stegvis kortast/”billigast” mellan start- och slutdestination. Slutligen jämfördes de två resultaten och rangordnades utifrån ett antal resenärattribut som antingen talar för att minska ruttens tidåtgång, eller antalet tågbyten längs vägen. Resultatet blev en personlig rekommendation av vilken rutt som är smidigast för de aktuella passagerarna, samt antalet resvägar mellan start- och slutdestination sett till antal byten.

En vidareutveckling på detta projekt skulle vara att lägga till fler städer och attribut till varje passagerare. Vidare skulle även en rangordning av resenärsegenskaperna vara vara av stor vikt för att förbättra rekommentationen av rutter. Detta skulle kunna uppnås genom att erbjuda alla resenärer en kort infyllnadsblankett över vilka egenskaper de tycker är viktigast att beakta vid rangordningen av rutter. Svaren från enkäten skulle sedan kunna ligga till grund för en viktning av attributen vilket skulle förbättra rekommendationerna till användaren.

Källor:

Dijkstras algorithm - Wikipedia:
Hämtad från: en.wikipedia.org
2022-12-06

Johnson, D. (1017). ”Graph Theory and Linear Algebra”.
Hämtad från: math.utah.org
2022-12-10