Using the A* algorithm

Anna Ursula Birkeland Brøyn og Erling Brækhus Haugsand

October 7, 2016

1 Problem A: Pathfinding in 2D Games

1.1 Problem A.1: Grids with obsticles

A* algoritmen ble implementert for å bruke "best-first-search" for å finne raskeste vei fra A til B i et todimensjonalt brett. Som språk ble Java brukt. Det ble laget tre klasser: AStar, Node og State. AStar ble brukt for å implementere selve algoritmen der Node- og State-klassen ble brukt. Node-klassen har generelle egenskaper for A* klassen. State-klassen er spesifikk for problemene i denne rapporten. Resultatene ble printet til fil og Matlab ble brukt for visualisering.

Figur 1 viser resultatet for fire forskjellige brett.

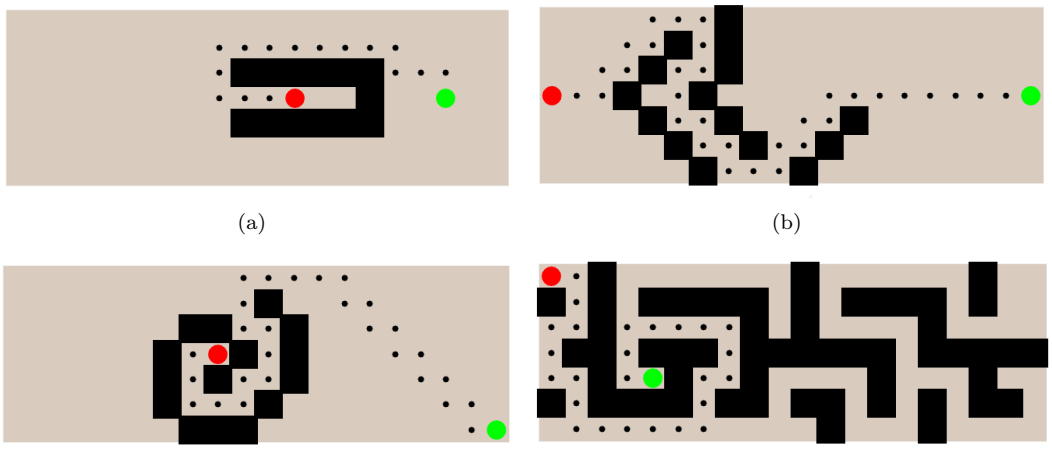


Figure 1: Figuren viser de fire brettene med raskeste vei indikert på hver figur.

1.2 Problem A.2: Grids with Different Cell Costs

I stedenfor å kun se på brett med åpne veier eller med hindringer så kan man ha et brett med ulike felt som *koster* ulikt å gå over. Vi definerer fem ulike typer "landskap" som har ulik kost. Vann (blått) har kost 100, fjell (grått) har kost 50, skog (mørkegrønt) har kost 10, gress (lysegrønt) har kost 5 og vei (beage) har kost 1. I forrige oppgave tilsvarte åpne veier veier med kost 1. Figur 2 viser raskeste vei for fire slike brett.

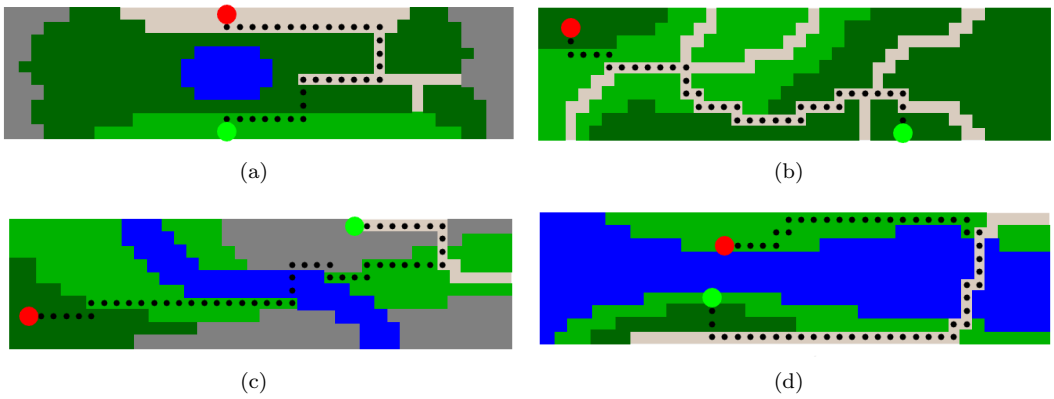


Figure 2: Figuren viser de fire brettene med ulik kost der raskeste vei er indikert på hver figur.

1.3 Problem A.3: Comparison with BFS and Dijkstra's Algorithm

Vi vil sammenligne A* algoritmen med Bredde-først-søk (BFS) og Dijkstra's algoritme for å finne raskeste vei. Figur 3, 4 og **??** viser raskeste vei funnet for de til sammen 8 brettene med henholdsvis BFS-algoritmen, Dijkstra's algoritme og A* algoritmen. Vi vil nå diskutere og sammenligne algoritmene ved å se på hvert brett (a til h).

På brett (a), (b) og (c) så finner alle algoritmene forskjellige veier. Men alle veiene er like lange så ingen av dem er bedre enn de andre med tanke på korrekthet. Derimot ser vi at A* algoritmen er mye mer effektiv enn de andre to ettersom den ikke har åpnet og lukket på langt nær like mange noder som det BFS og Dijkstra-algoritmen har. Dijkstra-algoritmen er litt bedre enn BFS-algoritmen i alle tilfeller.På brett (d) er forskjellen mindre. Alle algoritmene finner den samme veien og selv om A* star algoritmen er litt mer effektiv enn de andre er ikke antall åpnede og lukkede noder så veldig forskjellig for de tre algoritmene. At alle algoritmene finner samme vei skyldes naturligvis at det kun er én vei som er den korteste på dette brettet. At antall åpnede og lukkede noder heller ikke er særlig forskjellig for de tre algoritmene skyldes også at det

er et begrenset antall noder man kan åpne (pga alle hindringene).

På brettene der vi har varierende kost kommer forskjellen mellom de tre algoritmene enda tydligere fram. For brett (e) finner alle algoritmene forskjellige veier. Dijkstra og A* gir nesten lik vei og har samme totalkost, de kan derfor sies å være like "korrekte". BFS gir den korteste veien i luftlinje, men fordi den krysser et vann er dette ikke den korteste veien. BFS gir scorer derfor dårligst på korrekthet. Dijkstra og A* ser ut til å åpne og lukke omtrent like mange noder i dette tilfellet, og scorer omtrent likt totalt når det kommer til korrekthet og effektivitet.

For brett (f) så har alle nodene nesten like veier, men også her scorer nok BFS litt dårligere på korrekthet. A* star er mest effektiv, Dijkstra er mer effektiv enn BFS.

For brett (g) og (h) så har igjen Dijkstra og A* nesten like veier og begge er like raske. A* er igjen litt mer effektiv enn Dijkstra, fordi den åpner og lukker litt færre noder. BFS finner ikke den raskeste veien i noen av tilfellene.

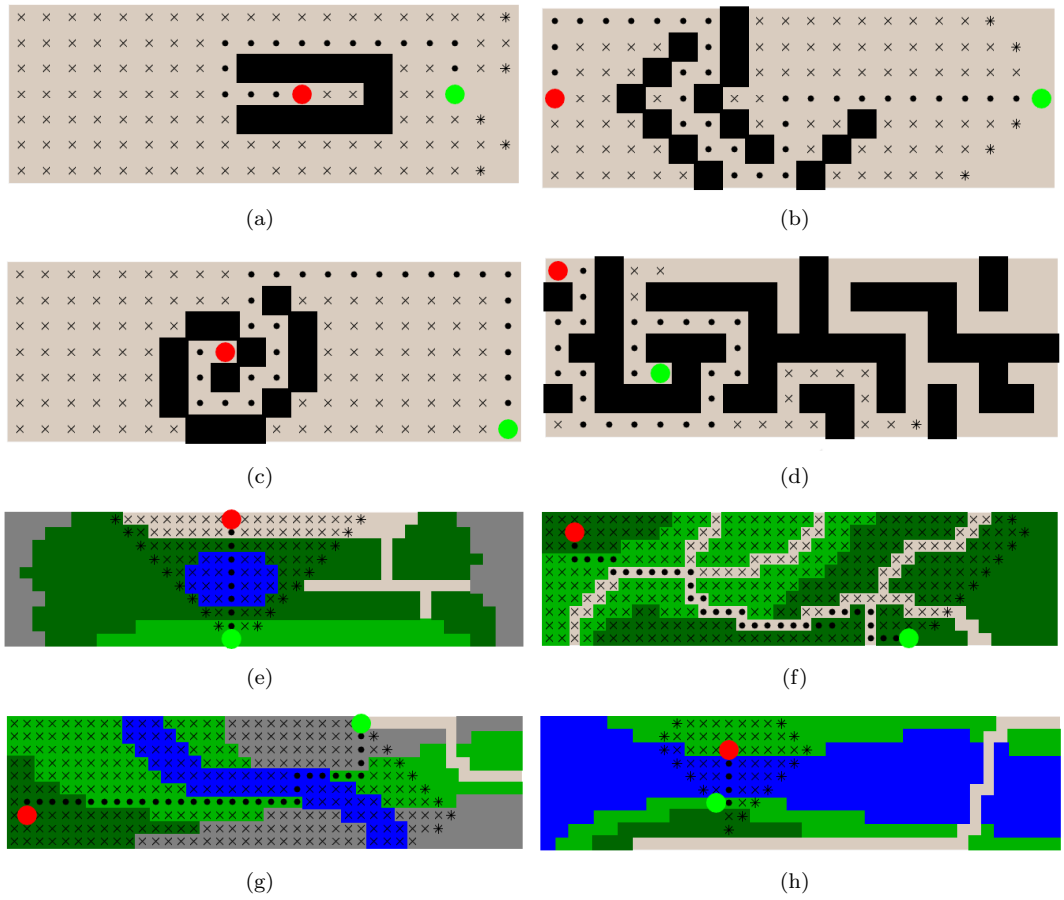
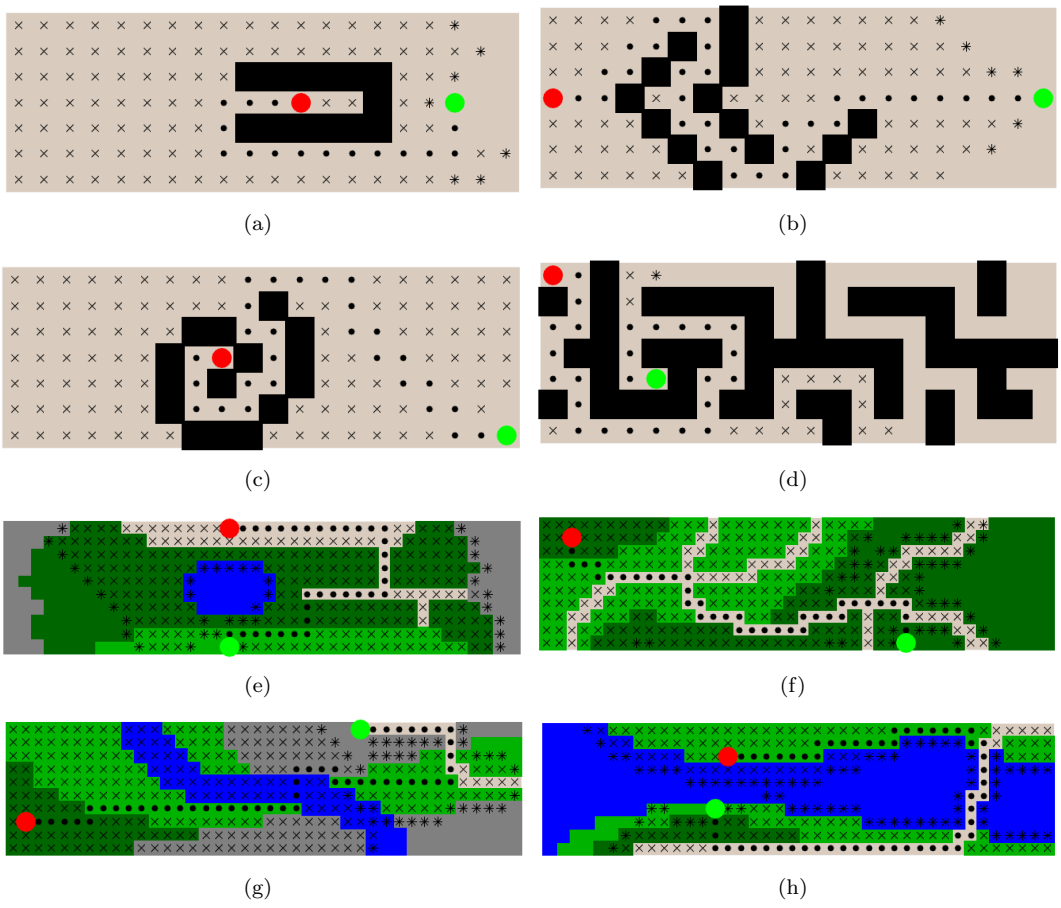
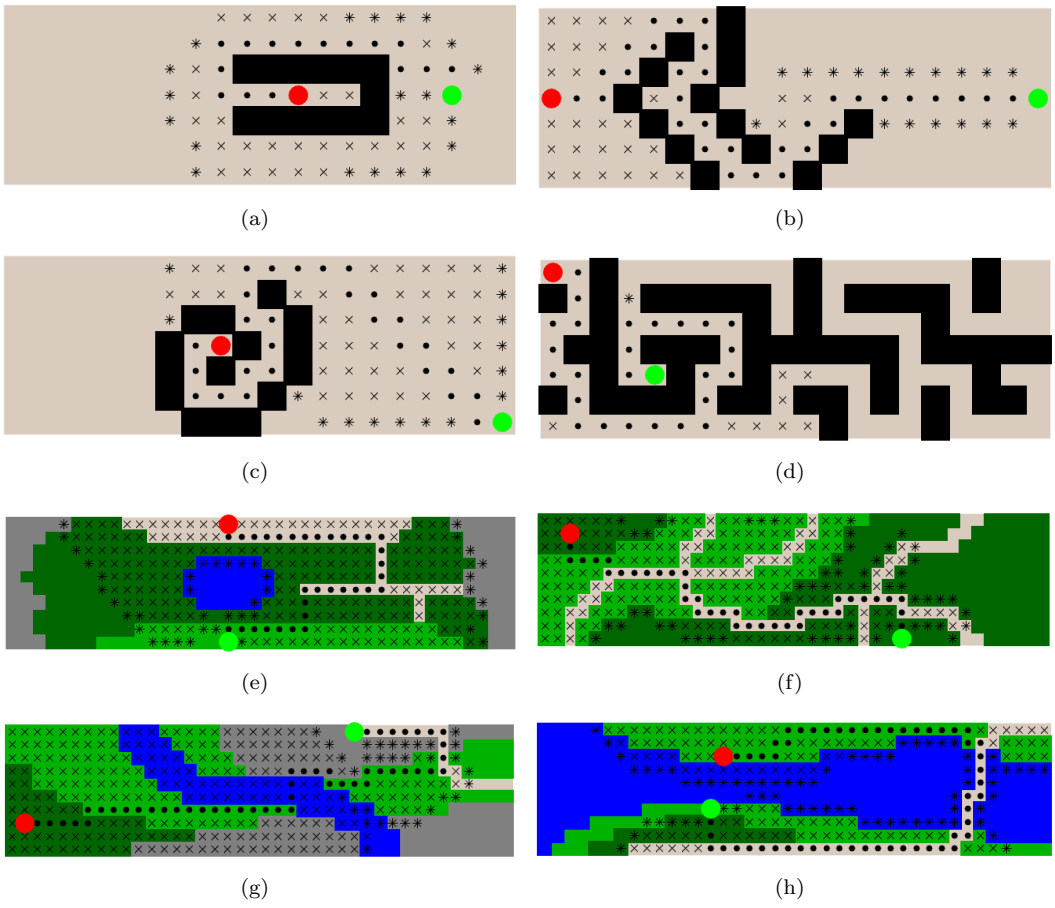


Figure 3: Figuren viser de fire brettene med ulik kost der raskeste vei funnet ved BFS er indikert på hver figur samt åpne (*) og lukkede (x) noder.



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 4: Figuren viser de fire brettene med ulik kost der raskeste vei funnet ved Dijkstra er indikert på hver figur samt åpne (*) og lukkede (x) noder.



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 5: Figuren viser de fire brettene med ulik kost der raskeste vei funnet ved A* er indikert på hver figur samt åpne (*) og lukkede (x) noder.

2 Problem B: Rush Hour Puzzle

Vi rakk dessverre ikke å gjøre ferdig denne delen av oppgaven. I den vedlagte koden kan man se at State-klassen er i ferd med å modifiseres til å inkludere Rush Hour, men vi ble ikke ferdige. Som heuristic function så planla vi å bruke følgende formel:

$$h(s) = x_g - x_{c_0} - \sum_i \delta(y_{c_i} - y_g)\theta(x_{c_i} - x_{c_0}). \tag{1}$$

der x_g og y_g er henholdsvis x- og y-koordinaten til målet, og x_{c_i} og y_{c_i} er henholdsvis x- og y-koordinaten til bil nr i. Vi har at $\delta(x) = 1$ for $x = 0$ og $\delta(x) = 0$ ellers og $\theta(x) = 1$ for $x > 0$ og $\theta(X) = 0$ ellers. På den måten gir vi et positiv bidrag for tilstander der bil 0 er nære målet og et negativt bidrag for biler som er i mellom bil 0 og målet. Det bør muligens være noen konstanter forran uttrykkene, men det er noe som man bør teste seg fram til.