


Универзитет „Св.Кирил и Методиј“ – Скопје
Факултет за информатички науки и компјутерско инженерство

Прва домашна работа по

Вештачка интелигенција

Пребарување со алгоритмите Uniform Cost и Recursive Best First Search

(18 група)

Стефан Андонов
индекс: 151020

професор:
д–р Катерина Здравкова

19 Март, 2017
Скопје

1. Проблем

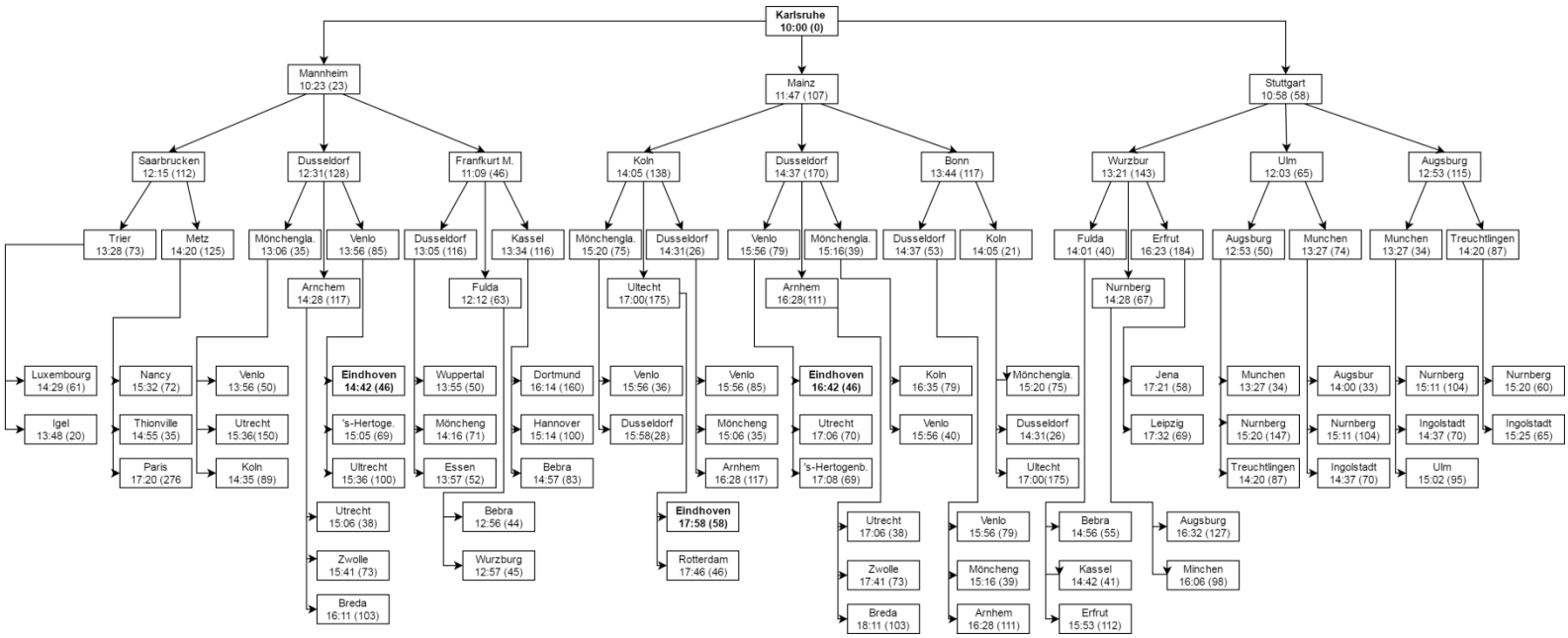
Проблемот којшто ќе го разгледувам во ова домашна задача е да се најде најкраткиот пат (временски) со воз од Karlsruhe, Германија, до Eindhoven, Холандија. Патувањето потребно е да започне на 15 април 2017 година во 10:00 часот. За решавање на проблемот ќе се искористат два алгоритми – Uniform Cost и Recursive Best First Search.

2. Познати податоци

Преку истражување на железничката мапа на Европа, прикажана на сликата погоре, како и истражување на возните редови достапни на веб страната https://www.trainline.eu, го креираме дрвото на пребарување за овој проблем. Дрвото на пребарување е развиено до максимална длабочина m=4, при што целта/дестинацијата се јавува 3 пати. Максималниот фактор на разгранување е b=3. Во некои случаи, некои јазли се разгранети на 2 деца, од причина што не беше возможно да се најдат соседи на некој град, кои што се поврзани со директна железничка линија.
Критериуми за развивање на дрвото беа:

- Да не се враќаме во веќе поминат град
- Помеѓу два јазли кои ги поврзуваме да постои **директна** железничка линија.

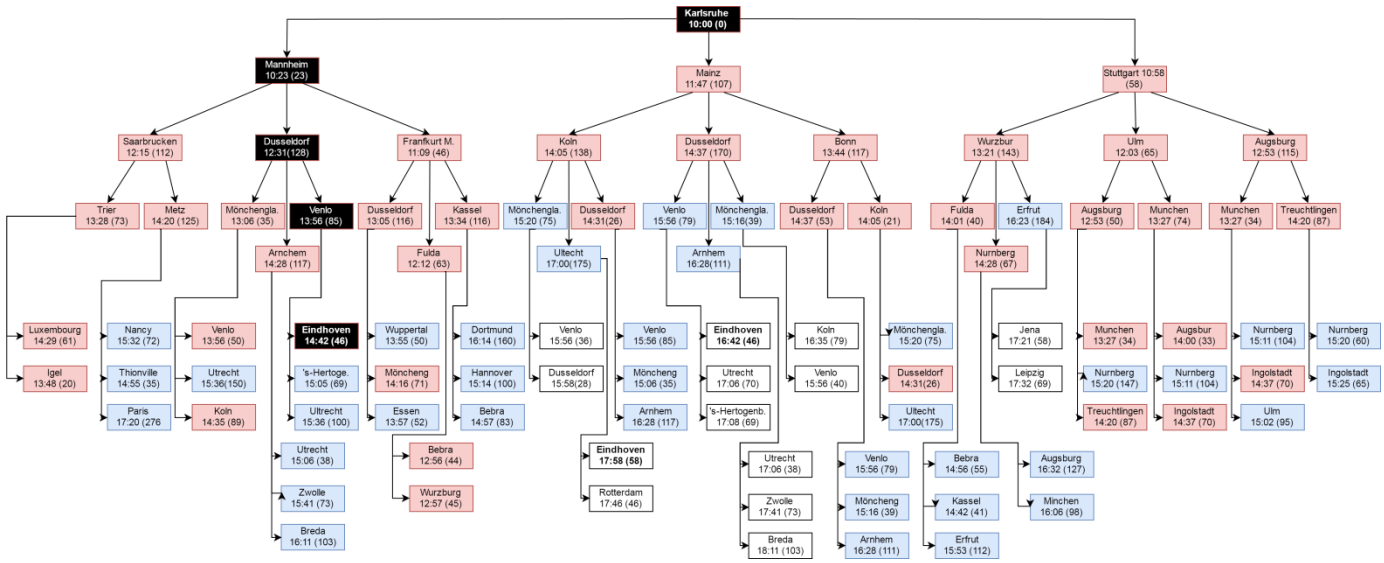
На сликата подолу е дадено дрвото на пребарување кое го креирав за овој проблем. Дрвото подобро можете да го видете во документот startsearchtree.pdf којшто се наоѓа во архивата.
За секој јазел се дадени името на градот, времето на пристигнување во тој град, како и цената за пристигнување во тој град од претходниот град во којшто се наоѓал, односно од неговиот родител. Во цената е вклучено и времето на чекање на станиците помеѓу две патувања.



3. Неинформирано пребарување низ дрвото со алгоритмот Uniform Cost

Алгоритмот Uniform Cost претставува алгоритам за неинформирано (слепо) пребарување. Тој функционира на начин што секогаш го разложува најефтиниот неразложен јазел, се додека не стигне до посакувана цел. Важно е да се напомени, дека овој алгоритам при пресметување на цената на еден јазел, ги додава и цените на неговите претходници, се до коренот на дрвото.

Комплетната илустрација на овој алгоритам, чекор по чекор, итерација по итерација, можете да ја погледните во документот UniformCostSimulation.pdf, којшто се наоѓа во архивата.



На сликата е прикажано дрвото на пребарување, после извршувањето на алгоритмот Uniform Cost. Јазлите обоени во црно ја прикажуваат најдената патека Karlsruhe- Mannheim –

Dusseldorf – Venlo – Eindhoven. Според овој алгоритам, најкраткото патување од Karlsruhe – Eindhoven трае 4 часа и 42 минути, односно 282 минути вкупно, на длабочина d=4.

4. Хевристичка функција

Со цел да го примениме алгоримот RBFS, потребно е најпрво да ја дефинираме хевристичката функција h(n) за секој јазел во пребарувачкото дрво. Јазлите кои што се однесуваат на исти градови, секако ќе имаат иста вредност за h(n).

Хевристичката функција ќе ја дефинираме како времето потребно во минути за да се измине од градот X до дестинацијата Eindhoven, преку воздушното растојание со просечен воз во Германија којшто се движи со брзина од 240 km/h, односно 4 km/min. Ова брзина се зема поради фактот дека понекогаш за транспорт не се користи ICE брзиот воз (којшто се движи со брзина од 330 km/h), туку понекогаш кога се работи за блиски места, се користат и локални железнички линии со многу помала брзина.

$$h(n) = \frac{\text{најкуса воздушна линија } (km)}{\text{просечна брзина на воз } (\frac{km}{min})} = \frac{\text{НВЛ } (km)}{4\ \frac{km}{min}} = \frac{\text{НВЛ}}{4}\, min$$

Ова хевристичка функција можеме да ја сметаме за прифатлива (admissible) поради тоа што секоја железничка пруга/автопат, е изградена подолга за разлика од должината на воздушното растојание меѓу две места, поради географски препреки, како што се планини, реки, езера и сл. Дополнително, кога патуваме со воз од место А до место Б, и тие две станици не се поврзани директно, секако ќе имаме време на чекање на станиците измеѓу, како и време за промена на воз.

Поради горе наведените причини, вредноста на хевристичката функција за било кој јазел не може да го надмине вистинското време на патување со воз од град X до дестинацијата Eindhoven.

Вредностите за хевристичката функција за секој град поединечно се прикажани во симулацијата за алгоритмот RBFS, како и во посебниот документ heuristicFunctionValues.pdf којшто се наоѓа во архивата.

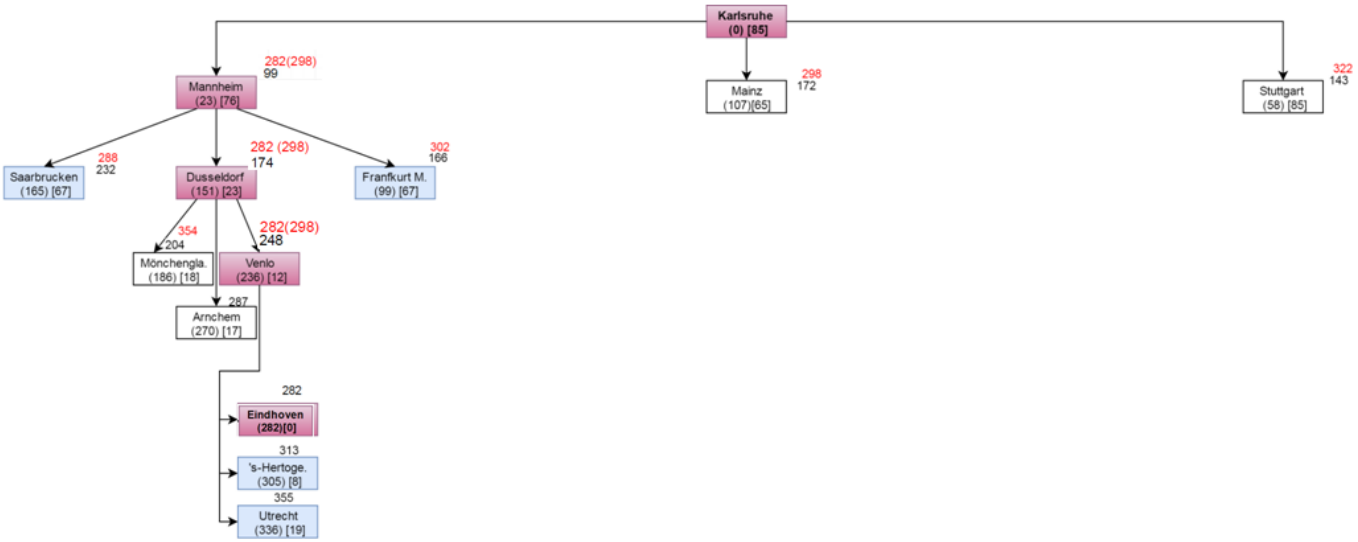
5. Информирано пребарување со алгоритмот Recursive Best First Search

Овој алгоритам претставува варијанта на IDA* алгоритмот и е алгоритам за информирано пребарување низ дрво. Функцијата за евалуација на секој јазел се пресметува како f(n)=g(n)+h(n), каде што h(n) е хевристичката функција, а g(n) е вистинската цена да се стигне од почетниот јазел – коренот на дрвото до моменталниот јазел n. Вредноста за g(n) ќе ја пресметаме со собирање на цените од почетното дрво на моменталниот јазел и сите негови претходници, се до коренот на дрвото. Затоа во дрвото за пребарување за овој алгоритам ќе има две промени:

- Вредноста на g(n) ќе биде дадена во отворени загради ( )
- Вредноста на h(n) ќе биде дадена во затворени загради [ ]
- Вредноста на f(n) ќе биде дадена над јазелот

Овој алгоритам функционира на начин што тој рекурзивно ја следи вредноста на f на најдобриот алтернативен пат. Овој алгоритам чува и дополнителна вредност F за секој разложен јазел, која

првично се поставува да биде еднаква на f. Дополнително се чува граница (bound) која што доколку биде надмината, се прекинува разложувањето на јазелот и се променува вредноста на F со минимумот од вредностите на f (кои се поголеми од моменталното F) на децата на јазелот. Постапката се повторува рекурзивно, се додека не се стигне до јазел којшто за вредноста на хевристичката функција h(n) има вредност 0 (нула).



На сликата погоре е прикажан изгледот на дрвото во неговата последна итерација, кога го пронаоѓа оптималниот пат од Karlsruhe до Eindhoven преку станиците во Mannheim, Dusseldorf и Venlo со цена од 282, односно траење од 4 часа и 42 минути, на длабочина d=4, комплетно исто како кај алгоритамот Uniform Cost.

Комплетната илустрација на овој алгоритам, чекор по чекор, итерација по итерација, можете да ја погледните во документот RBFSSimulation.pdf, којшто се наоѓа во архивата.

6. Оценка на решенијата со двата алгоритми

Алгоритмите за пребарување во дрво можеме да ги оцениме според следниве четири критериуми:

- Комплетност
- Оптималност
- Временска сложеност – број на изгенерирани јазли
- Просторна (мемориска) сложеност – најголем број на јазли во меморијата

Uniform Cost како алгоритам е комплетен и секогаш го наоѓа оптималното решение, доколку истото постои. Неговата временска и мемориска комплексност изнесува $O(b^{\lceil 1+\frac{C^*}{\varepsilon} \rceil})$, каде што b е максималниот фактор на разгранување, C* е цената на оптималното решение, а ε е цена која што секој чекор ја надминува.

Recursive Best First Search како алгоритам е комплетен и секогаш го наоѓа оптималното решение, под услов хевристичката функција да е прифатлива (admissible). Мемориската комплексност на овој алгоритам е $O(bd)$. Временската комплексност не може секогаш точно да се определи и зависи многу од начинот на избор на хевристичката функција. Според Корф, во

генерален случај може да смета дека временска сложеност изнесува $O(b^d)$, каде што b е асимптотскиот фактор на разгранување, а пак d е длабочината на пронајденото решение.

Во нашиот проблем, наоѓање на најкраток пат од Karlsruhe до Eindhoven, имаме креирано дрво со максимален фактор на разгранување b=3, со максимална длабочина m=4, и со длабочина на решението со двата алгоритми, d=4.

И двата алгоритми го наоѓаат најкраткиот пат до Eindhoven со цена 282 (минути). Постојат уште два начини да се стигне до Eindhoven од кои едниот со цена 478 минути, а другиот со цена 402.

Во спроведувањето на Uniform Cost алгоритмот, цената на оптималното решение е C*=282, а пак цената која што секој чекор ја надминува ε=20 (цената од Trier до Igel). Со заменување на вредностите во формулата, добиваме дека мемориската и временската комплексност на алгоритмот Uniform Cost изнесува:

$$O\left(b^{\left\lceil 1+\frac{C^*}{\varepsilon}\right\rceil}\right) = O\left(3^{\left[1+\frac{282}{20}\right]}\right) = O(3^{1+14.1}) \approx O(3^{15})$$

Приближно **3^{15} јазли** овој алгоритам генерира и чува во меморија.

Во спроведување на алгоритмот Recursive Best First Search, доколку ги замениме вредностите во формулите, мемориската комплексност изнесува O(12), додека пак временска сложеност во општ случај, изнесува $O(3^4)$. Имено, овој алгоритам во меморија чува максимум **12 јазли**, а генерира најмногу **81 јазел**.

Бидејќи и двата алгоритми ни се комплетни и го наоѓаат оптималното решение, слободно можеме да заклучиме дека многу подобро за времето на извршување, како и за меморијата на компјутерот е да го користиме алгоритмот **Recursive Best First Search**.