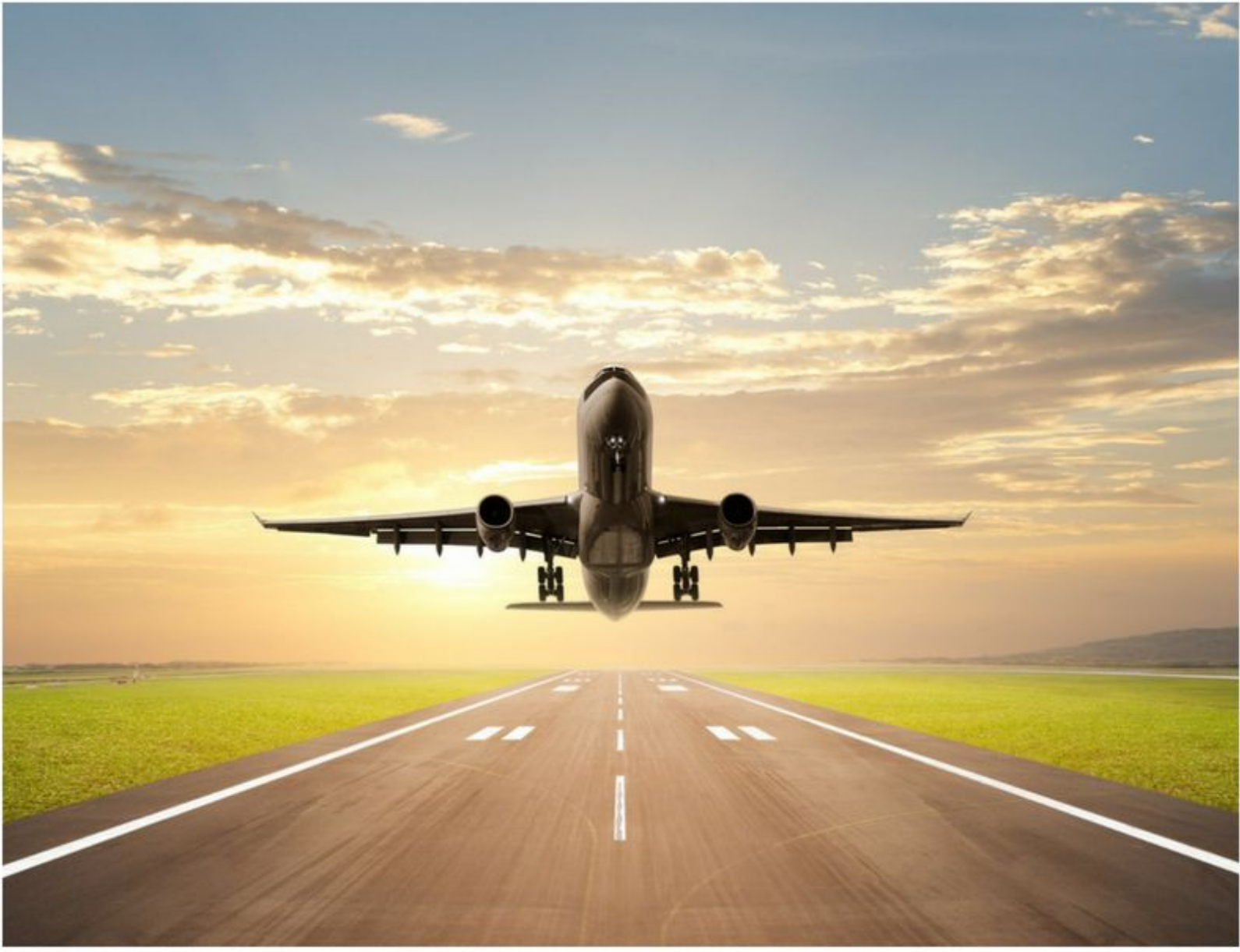

ULTIMATE TRIP PLANNER

JORDI FOIX - MIQUEL GÓMEZ - GUILLEM ROSSELLÓ
INTEL·LIGÈNCIA ARTIFICIAL

PRÀCTICA DE PLANIFICACIÓ

JUNY 2018

ÍNDEX

1. **Introducció i metodologia de treball**

2. **Nivell bàsic**
   2.1. Modelat del domini
   2.2. Modelat dels problemes
   2.3. Conjunts de prova

3. **Extensió 1**
   2.1. Modelat del domini
   2.2. Modelat dels problemes
   2.3. Conjunts de prova

4. **Extensió 2**
   2.1. Modelat del domini
   2.2. Modelat dels problemes
   2.3. Conjunts de prova

5. **Extensió 3**
   2.1. Modelat del domini
   2.2. Modelat dels problemes
   2.3. Conjunts de prova

6. **Extensió 4**
   2.1. Modelat del domini
   2.2. Modelat dels problemes
   2.3. Conjunts de prova

7. **Resultats i conclusions**

1 Introducció i metodologia de treball

Per al desenvolupament d'aquesta pràctica hem decidit partir no de zero, sinó del punt on vam començar l'anterior pràctica. Així doncs, hem adaptat el generador d'instàncies que vam fer per a CLIPS i amb ell hem anat generant tots els problemes d'aquesta pràctica. I per a cadascun d'ells n'hem fet el seu domini segons indicava l'enunciat.

En les properes pàgines expliquem detalladament com hem estructurat i programat cadascún dels casos: Cas bàsic, extensió 1, extensió 2, extensió 3 i extensió 4.

En quant a la metodologia emprada en el desenvolupament de la pràctica, ens ha estat molt fàcil seguir una estratègia de prototipat incremental ja que la mateixa pràctica estava preparada per a ser atacada així si s'optava per a fer totes les extensions, com hem fet en la nostra pràctica. Això ens ha permès fer un desenvolupament de les extensions més modular i senzill, sobretot destacant també l'importància del generador de problemes parametritzat, que ens ha anat molt bé per anar provant les diferents extensions. amb problemes diversos d'una manera molt ràpida.

2 Nivell bàsic

Tipus i predicats

Al nivell bàsic el nostre domini esta format per 2 tipus que són **city** i **hotel** i els següents predicats que les relacionen:

inCity: el qual indica a quina ciutat es troba un hotel
flight: que presenta vols per les diferents parelles de ciutats

També fem ús d'altres predicats en l'execució del problema:

visited: Indica que una ciutat ja és visitada en el viatge i ens permet evitar agafar-la un segon cop durant el viatge
selected-flight: Indica quin vol s'agafa en el transport d'una ciutat a l'altra
sleepIn: Indica l'allotjament de l'usuari en cada ciutat que conforma el viatge
last-city: Representa la última ciutat visitada en el viatge construït fins el moment, el qual ens permet determinar el vol a la següent ciutat fent ús de la variable inCity

Funcions

Les funcions definides en el nostre problema són les següents:

visited-cities: ens permet tenir un control del nombre de ciutats visitades, de manera que en la representació del goal podrem comprovar que tenim un nombre correcte.
total-days: valor que representa el nombre de dies totals del viatge, també usat per a assegurar-ne un mínim total.

Accions

Les accions definides en el nostre problema són les següents:

select-first-city: És la primera acció a executar-se, ja que només en la inicialització del problema la funció total-days té valor 0. En aquesta acció seleccionem una ciutat i un hotel pertanyent a aquesta. Augmentarem en 1 també el valor de la funció total-days i visited-cities.
select-next-city: Té el mateix funcionament que l'acció select-first-city (tot i que no canvia d'hotel), però a més agafa un vol desde la darrera ciutat (identificada per last-city) a l'actual.

3 Extensió 1

Millora del nivell bàsic, **canvis ressaltats**

Tipus i predicats

A l'extensió 1 el nostre domini esta format per 2 tipus que són **city** i **hotel** i els següents predicats que les relacionen:

inCity: el qual indica a quina ciutat es troba un hotel
flight: que presenta vols per les diferents parelles de ciutats

També fem ús d'altres predicats en l'execució del problema:

visited: Indica que una ciutat ja és visitada en el viatge i ens permet evitar agafar-la un segon cop durant el viatge
selected-flight: Indica quin vol s'agafa en el transport d'una ciutat a l'altra
sleepIn: Indica l'allotjament de l'usuari en cada ciutat que conforma el viatge
last-city: Representa la última ciutat visitada en el viatge construït fins el moment, el qual ens permet comprovar que hi estem un nombre de dies correcte (dins el rang pertinent) i també ens permet trobar el vol a la següent ciutat fent ús de la variable inCity
incomplete: Aquest predicat també s'usa per a assegurar que la darrera ciutat visitada tingui un nombre de dies dins el rang que li pertoca, el posem a false si quan afegim una nova ciutat (per defecte amb un sol dia) no assolim el mínim de dies i el posem a true en el cas que executem la acció stay-one-more-day i el nombre de dies a la ciutat ja hagi assolit el mínim necessari

Funcions

Les funcions definides en el nostre problema són les següents:

visited-cities: ens permet tenir un control del nombre de ciutats visitades, de manera que en la representació del goal podrem comprovar que tenim un nombre correcte.
total-days: valor que representa el nombre de dies totals del viatge, també usat per a assegurar-ne un mínim total.
stay: aquesta funció està definida per cada ciutat, i ens indica el nombre de dies que ens quedem a cada ciutat, juntament amb les funcions minDaysCity i maxDaysCity, podem comprovar que s'hi passaran un nombre de dies correcte
minDaysCity: determina el mínim de dies a passar en una ciutat
maxDaysCity: determina el màxim de dies a passar en una ciutat

Accions

Les accions definides en el nostre problema són les següents:

select-first-city: És la primera acció a executar-se, ja que només en la inicialització del problema la funció total-days té valor 0. En aquesta acció seleccionem una ciutat i un hotel pertanyent a aquesta. Modificarem el valor de la funció stay per la ciutat concreta, augmentant-ne el valor de 0 a 1, augmentarem en 1 també el valor de la funció total-days i visited-cities.
stay-one-more-day: En aquesta acció augmentarem en un el valor de stay de la last-city, així com el nombre total de total de dies. En cas que assolim el mínim de dies posarem a false el predicat incomplete.
select-next-city: Té el mateix funcionament que l'acció select-first-city, però a més comprova que la darrere ciutat ja tingui un nombre de dies correcte (amb el predicat incomplete) i ja agafa un vol desde la darrera ciutat (identificada per last-city) a l'actual.

4 Extensió 2

Millora de l'extensió 1, **canvis ressaltats**

Tipus i predicats

A l'extensió 2 el nostre domini esta format per 2 tipus que són **city** i **hotel** i els següents predicats que les relacionen:

- **inCity**: el qual indica a quina ciutat es troba un hotel
- **flight**: que presenta vols per les diferents parelles de ciutats

També fem ús d'altres predicats en l'execució del problema:

- **visited**: Indica que una ciutat ja és visitada en el viatge i ens permet evitar agafar-la un segon cop durant el viatge
- **selected-flight**: Indica quin vol s'agafa en el transport d'una ciutat a l'altra
- **sleepIn**: Indica l'allotjament de l'usuari en cada ciutat que conforma el viatge
- **last-city**: Representa la última ciutat visitada en el viatge construït fins el moment, el qual ens permet comprovar que hi estem un nombre de dies correcte (dins el rang pertinent) i també ens permet trobar el vol a la següent ciutat fent ús de la variable inCity
- **incomplete**: Aquest predicat també s'usa per a assegurar que la darrera ciutat visitada tingui un nombre de dies dins el rang que li pertoca, el posem a false si quan afegim una nova ciutat (per defecte amb un sol dia) no assolim el mínim de dies i el posem a true en el cas que executem la acció stay-one-more-day i el nombre de dies a la ciutat ja hagi assolit el mínim necessari

Funcions

Les funcions definides en el nostre problema són les següents:

- **visited-cities**: ens permet tenir un control del nombre de ciutats visitades, de manera que en la representació del goal podrem comprovar que tenim un nombre correcte.
- **total-days**: valor que representa el nombre de dies totals del viatge, també usat per a assegurar-ne un mínim total.
- **city-interest**: El seu valor marca l'interès d'una citutat determinada, sent aquest un valor [1,2,3] (1 és màxim interès).
- **total-interest**: És la suma del total de city-interests de totes les ciutats que hem visitat fins al moment. És optimitzat per la mètrica **minimize**.
- **stay**: aquesta funció està definida per cada ciutat, i ens indica el nombre de dies que ens quedem a cada ciutat, juntament amb les funcions minDaysCity i maxDaysCity, podem comprovar que s'hi passaran un nombre de dies correcte
- **minDaysCity**: determina el mínim de dies a passar en una ciutat
- **maxDaysCity**: determina el màxim de dies a passar en una ciutat

Accions

Les accions definides en el nostre problema són les següents:

- **select-first-city**: És la primera acció a executar-se, ja que només en la inicialització del problema la funció total-days té valor 0. En aquesta acció seleccionem una ciutat i un hotel pertanyent a aquesta. Modificarem el valor de la funció stay per la ciutat concreta, augmentant-ne el valor de 0 a 1, augmentarem en 1 també el valor de la funció total-days i visited-cities. Ara a més sumem el cityInterest de la ciutat al total-interest del viatge, que optimitzarem en la mètrica **minimize**.

stay-one-more-day: En aquesta acció augmentarem en un el valor de stay de la last-city, així com el nombre total de total de dies. En cas que assolim el mínim de dies posarem a false el predicat incomplete.

select-next-city: Té el mateix funcionament que l'acció select-first-city, però a més comprova que la darrere ciutat ja tingui un nombre de dies correcte (amb el predicat incomplete) i ja agafa un vol desde la darrera ciutat (identificada per last-city) a l'actual. Com a select-first-city, sumem el cityInterest de la ciutat al total-interest del viatge.

5 Extensió 3

Millora de l'extensió 1, **canvis ressaltats**

Tipus i predicats

A l'extensió 3 el nostre domini esta format per 2 tipus que són **city** i **hotel** i els següents predicats que les relacionen:

inCity: el qual indica a quina ciutat es troba un hotel
flight: que presenta vols per les diferents parelles de ciutats

També fem ús d'altres predicats en l'execució del problema:

visited: Indica que una ciutat ja és visitada en el viatge i ens permet evitar agafar-la un segon cop durant el viatge
selected-flight: Indica quin vol s'agafa en el transport d'una ciutat a l'altra
sleepIn: Indica l'allotjament de l'usuari en cada ciutat que conforma el viatge
last-city: Representa la última ciutat visitada en el viatge construït fins el moment, el qual ens permet comprovar que hi estem un nombre de dies correcte (dins el rang pertinent) i també ens permet trobar el vol a la següent ciutat fent ús de la variable inCity
incomplete: Aquest predicat també s'usa per a assegurar que la darrera ciutat visitada tingui un nombre de dies dins el rang que li pertoca, el posem a false si quan afegim una nova ciutat (per defecte amb un sol dia) no assolim el mínim de dies i el posem a true en el cas que executem la acció stay-one-more-day i el nombre de dies a la ciutat ja hagi assolit el mínim necessari

Funcions

Les funcions definides en el nostre problema són les següents:

visited-cities: Ens permet tenir un control del nombre de ciutats visitades, de manera que en la representació del goal podrem comprovar que tenim un nombre correcte.
total-days: Valor que representa el nombre de dies totals del viatge, també usat per a assegurar-ne un mínim total.
total-price: Preu total del viatge fins al moment.
stay: aquesta funció està definida per cada ciutat, i ens indica el nombre de dies que ens quedem a cada ciutat, juntament amb les funcions minDaysCity i maxDaysCity, podem comprovar que s'hi passaran un nombre de dies correcte.
minDaysCity: Determina el mínim de dies a passar en una ciutat.
maxDaysCity: Determina el màxim de dies a passar en una ciutat.
flightPrice: Preu del vol d'una ciutat a una altra.
hotelPrice: Preu d'un hotel determinat.
minPriceTrip: Preu mínim del viatge indicat des del generador del problema, i usat en la comprobació del goal state.
maxPriceTrip: Preu màxim del viatge indicat des del generador del problema, i usat en la comprobació del goal state.

Accions

Les accions definides en el nostre problema són les següents:

select-first-city: És la primera acció a executar-se, ja que només en la inicialització del problema la funció total-days té valor 0. En aquesta acció seleccionem una ciutat i un hotel pertanyent a aquesta. Modificarem el valor de la funció stay per la ciutat concreta, augmentant-ne el valor de 0 a 1,

augmentarem en 1 també el valor de la funció total-days i visited-cities. Ara a més sumem el preu de l'hotel de la ciutat al preu total del viatge, que optimitzarem en la mètrica **minimize**.

stay-one-more-day: En aquesta acció augmentarem en un el valor de stay de la last-city, així com el nombre total de total de dies. En cas que assolim el mínim de dies posarem a false el predicat incomplete. Aquí també sumem el preu de l'hotel de la ciutat al preu total del viatge.

select-next-city: Té el mateix funcionament que l'acció select-first-city, però a més comprova que la darrere ciutat ja tingui un nombre de dies correcte (amb el predicat incomplete) i ja agafa un vol desde la darrera ciutat (identificada per last-city) a l'actual. Aquí també sumem el preu de l'hotel de la ciutat al preu total del viatge i també el preu del vol entre les dues ciutats.

6 Extensió 4

Característiques i comentaris:

Aquesta extensió és bàsicament una agregació de les extensions 2 i 3, i com a tal, té en compte tant els interessos de les ciutats que formen part del recorregut com el preu dels seus hotels i el dels vols que les connecten.

La mètrica minimize que hem especificat en aquesta extensió fa una suma ponderada d'aquests dos factors (interès total i preu total) a partir dels valors normalitzats i multiplicats per un factor que podem ajustar.

Una ponderació d'importància del cost respecte l'interès la podríem obtenir per exemple del maxPriceTrip especificat, ja que en general si es vol gastar poc es primarà molt més el cost que l'interès i en canvi si es pot gastar molt cobra molta més importància l'interès. Però com que tampoc podem estar mai segurs d'això, hem observat que una proporció de ⅔ pel preu i ⅓ per l'interès és bastant adequada en la majoria de casos.

6 Resultats i conclusions

Per generar els problemes en totes les extensions i de diferents mides, hem decidit adaptar el generador que vam utiiltzar en l'anterior pràctica. L'hem parametritzat de forma que podem indicar-hi el nombre de ciutats i hotels, quina extensió volem, el nombre de ciutats a visitar, la durada mínima del viatge i el pressupost mínim i màxim. I per a tenir un control més ràpid sobre el canvi d'aquestes variables hem incorporat un seed que incrementa aquests paràmetres per a obtenir problemes més grans fàcilment.

En general la principal limitació que hem tingut ha estat a l'hora d'intentar representar problemes prou grans (per seeds més altes), quelcom problemàtic ja que si vulguéssim una escalabilitat mínima per fer-ho servir en un cas real ens passaríem sempre del límit, ja que connexions en avió en tindríem sempre moltes més de les que el fitxer del problema ens permet en quan a mida (tenim la limitació de 200 línies del Metric-FastForward).

Així i tot, amb problemes amb aproximadament 10-15 ciutats i un graf complet de vols hem pogut observar com es manifesta l'efecte de l'optimització del preu en les extensions 3 i 4.

pER Ú hem observat com la sol·lució optimitza prou l'interès en l'extensió 2, i també en la 4 quan li donem més importància en comparació amb el preu.