fejezet

Modellek

Az előbbiekben ismertetett probléma megoldását nem egy lépésben oldottam meg. Először is definiáltam különböző feladatosztályokat, amiket fokozatosan bővítettem. A feladatosztályokhoz kapcsolódóan pedig MILP modelleket írtam fel. Ezeket mutatom be jelen fejezetben.

Feladatosztály 1 – Járatok közötti mozgás minimalizálása depóhelyek nélkül

Ez az osztály foglalkozott az alapesettel, amiben még nincsen a buszoknak depója, vagyis az a hely, ahonnan a járatok elvégzése előtt elindul és ahova a járatok elvégzése után visszatér. Ebben az esetben még onnan indult egy busz ahonnan akart. A buszok üzemanyag szintjét és fogyasztását (energia-felhasználását) illetve tankolását (töltését) sem vettem még itt figyelembe. Úgymond „végtelen” üzemanyagtartályt (akkumulátor-kapacitást) feltételeztem az első feladatosztályban.

A kiindulási helyzetet is több irányból közelítettem meg. Felírtam két modellt (Modell1A és Modell1B), amelyek közül a későbbiekben már csak az egyiket fejlesztettem tovább.

A cél azon köztes utazások (más néven üresjáratok) távolságainak minimalizálása volt, amelyek az ütemezés során nem szállítottak utast, csak két menetrendszerinti járat befejezési és kezdési helye között haladtak át.

Bemeneti adatok

A főbb bemeneti adatok a buszok, a buszok járatai, valamint a járatok kezdő és végpontjául szolgáló helyek. Ezeket az adatokat halmazokba rendeztem, és különböző kiegészítő paramétereket definiáltam hozzájuk.

A `Helyek` halmaz azon földrajzi helyeket tartalmazza, amelyek a buszjáratok kezdő és végpontját adják. A halmaz elemeinek egymáshoz viszonyított értékeit a `tav` és az `ido`

paraméterek írják le. A `tav` néven definiált paraméter az adott helyiségek közötti kilométerben megadott távolságot mutatja meg. Az `ido` nevű paraméter a két helyiség közötti átlagos percben számolt átjutási időt jelzi.

Az elvégzendő buszjáratok a `Jaratok` halmazban találhatóak. A halmazhoz tartozó paraméterek a következők:

- `honnan` - a járat kiindulási helye a `Helyek` halmazból
- `hova` - a járat befejezésének helye a `Helyek` halmazból
- `mikortol` - a járat percben megadott kezdési időpontja
- `meddig` - a járat percben megadott befejezésének időpontja
- `jaratszam` - a végrehajtandó buszjáratok száma.

A rendelkezésre álló járműveket a `Buszok` halmaz fogja egybe. Az utazáshoz `buszszam` mennyiségű busz áll rendelkezésre.

Modell 1A

Változók

A modell két bináris változót használ. Ezek közül az egyik a `hozzarendel{Jaratok,Buszok}` változó, melynek értéke jelzi azt, hogy melyik buszjáratot ténylegesen melyik busz végzi el. Amennyiben az adott járat az adott buszhoz tartozik, akkor a változó értéke 1, ellenkező esetben 0.

Két buszjárat közötti átjárást az `atmenet{Buszok,Jaratok,Jaratok}` változó értékével fejeztem ki. Ha egy b busz végrehajt egy j1 járatot és utána közvetlenül a j2 járatot is, akkor az `atmenet[b,j1,j2]` változó értéke 1 lesz a későbbi ehhez kapcsolódó korlátozások hatására, minden más esetben pedig a 0 értéket veszi fel. Ez az `atmenet` változó a `hozzarendel` változó értékének hatására vesz fel értéket, így nem számít független döntésnek.

Bevezettem továbbá egy halmazt azoknak a járatpároknak, amiket nem lehet egy buszhoz rendelni, mivel ütközés lépne fel a végrehajtás és köztes utazások összes időtartamában. Ezek a párok a `Kulonbozobusz` halmaz elemeiként jelennek meg a modellben.

```
set Kulonbozobusz:= setof{j in Jaratok, j2 in Jaratok:
  mikortol[j] <= mikortol[j2]
  &&
  mikortol[j2] < meddig[j] + ido[hova[j],honnan[j2]]
  &&
  j != j2}
    (j,j2);
```

Ha például veszünk két egymástól eltérő j és j2 járatot, és j2 járat kezdési időpontja előbb van, mint ahogy egy busz az előző j járata után odaérne a j2 járat kiindulási helyére, akkor ez azt jelenti, hogy a két járatot külön busszal lehet csak végrehajtani. Ilyen párokból áll az előbb említett halmaz.

Korlátozások

A `JaratokElvegzese` korlátozás minden járathoz hozzárendel egy buszt, így biztosítja, hogy minden járat végre legyen hajtva.

```
s.t. JaratokElvegzese {j in Jaratok}: sum {b in Buszok} hozzarendel[j,b] = 1;
```

Csak különböző buszokkal elvégezhető járatpárokat nem rendelhetünk ugyanahhoz a buszhoz. Ennek érdekében jött létre az `OsszeferhetetlenJaratok` nevű korlátozás. A `Kulonbozobusz` halmazból vett járatpárt nem lehet egyszerre az adott buszhoz rendelni, legfeljebb a pár egyik tagját.

```
s.t. OsszeferhetetlenJaratok {(j,j2) in Kulonbozobusz, b in Buszok}:
  hozzarendel[j,b] + hozzarendel[j2,b] <= 1;
```

Amennyiben két járat (j és j2 is) hozzá van rendelve egy b buszhoz és nincs a két járat között más köztes járat hozzárendelve, akkor ez azt jeleni, hogy a busz az egyik járatból a másikba közvetlenül átmegy. Ilyen esetben az `atmenet[b,j,j2]` változó értéke 1-et vesz fel, egyébként pedig 0-t. Ezt a megállapítást tükrözi az `AtmenetKorlatozas`.

```
s.t. AtmenetKorlatozas
  {b in Buszok, j in Jaratok, j2 in Jaratok: mikortol[j2] >= meddig[j]}:
    atmenet[b,j,j2]
    +
    sum {jkoztes in Jaratok:
      mikortol[jkoztes] >= meddig[j]
      &&
      meddig[jkoztes] <= mikortol[j2]} hozzarendel[jkoztes,b]
    >=
    -1 + hozzarendel[j,b] + hozzarendel[j2,b];
```

Az előbbiekben bemutatott korlátozás két másikkal tovább élesíthető. (`AtmenetKorlatozas_elesito1`, `AtmenetKorlatozas_elesito2`).

```
s.t. AtmenetKorlatozas_elesito1
  {b in Buszok, j in Jaratok, j2 in Jaratok}:
    atmenet[b,j,j2] <= (hozzarendel[j,b] + hozzarendel[j2,b])/2;
```

```
s.t. AtmenetKorlatozas_elesito2
  {b in Buszok, j in Jaratok, j2 in Jaratok, jkoztes in Jaratok:
    mikortol[jkoztes]>=meddig[j]
    &&
    meddig[jkoztes] <= mikortol[j2]}:
  atmenet[b,j,j2]<=1-hozzarendel[jkoztes,b];
```

Cél

A cél a buszjáratok közötti áthaladások távolságainak minimalizálása. A célfüggvény felírásakor összegezni kell a járatok közötti átmenetek (üresjáratok) kilométeradatait.

```
minimize Koztestav:
  sum {b in Buszok, j1 in Jaratok, j2 in Jaratok}
    tav[hova[j1],honnan[j2]] * atmenet[b,j1,j2];
```

Modell 1B

Változók

A változók felírásához szükség volt még néhány újdonságra Modell1A-hoz képest, ezért bevezetésre került egy `maxjarat` nevű paraméter. Ez az érték az egy nap egy busz által elvégezhető járatok maximális számát jelöli. Ennek segítségével tudjuk definiálni a `Sorszam` elnevezésű halmazt, amely `maxjarat` mennyiségű elemet tartalmaz.

Ez a modell is két változóval dolgozik, azonban az előzőhöz képest itt már nem csak bináris változókról van szó. A `hozzarendel{Buszok,Sorszam,Jaratok}` nevű bináris változó 1 értéket akkor szolgáltat, amennyiben az adott busz az adott sorszámú járataként elvégzi az adott járatot. Minden más esetben a változó értéke 0 lesz.

A másik változó a `koztesutazas{Buszok, s in Sorszam:s!=maxjarat}` amit a modell arra használ, hogy az egyes járatok közötti áthaladások kilométerben kifejezett köztes távolságait számolhassuk vele. Értelem szerűen a változó csak nemnegatív értéket vehet fel.

Korlátozások

Az egyik korlátozás itt is az előírt járatok mindegyikének teljesítését hivatott szavatolni. A nevében is ugyanolyan `JaratokElvegzese` korlátozás azt mondja ki, hogy minden járatot hozzá kell rendelni egy busz valahányadik járatának.

```
s.t. JaratokElvegzese {j in Jaratok} : sum {b in Buszok, s in Sorszam}
  hozzarendel[b,s,j] = 1;
```

Az `EgyHelyreEgyet` nevű korlátozás szerint minden busz adott sorszámú járatának csak egyetlen járatot lehet választani.

```
s.t. EgyHelyreEgyet {b in Buszok, s in Sorszam} : sum {j in Jaratok}
  hozzarendel[b,s,j] <= 1;
```

Ki kell küszöbölni azt az esetet, hogy ha nincs egy járatnak s-edik járata, akkor ne lehessen s+1-edik sem. Ezt teszi meg az `EgymasMellettiJaratok` korlátozás.

```
s.t. EgymasMellettiJaratok {b in Buszok, s in Sorszam: s != maxjarat}:
  sum{j in Jaratok}hozzarendel[b,s,j]
  >=
  sum{j in Jaratok}hozzarendel[b,s+1,j];
```

Mivel ebben a modellben az előzőhöz képest nincsen `Kulonbozobusz` halmaz, ami a járatok közötti ütközések elkerülését hivatott segíteni, így felírtam egy korlátozást, ami ugyanezt a funkciót látja el. Az adott busznak elegendő idő kell ahhoz, hogy a j járat befejezési helyétől átérjen a j2 járat kezdési helyére. Csak akkor rendelhető hozzá az adott buszhoz mindkét (j, j2) járat, amennyiben teljesülnek az `Idokorlat`–ban megfogalmazottak.

```
s.t. Idokorlat
  {b in Buszok, s in Sorszam, j in Jaratok, j2 in Jaratok: s != maxjarat}:
  mikortol[j2]
  >=
  (meddig[j] + ido[hova[j],honnan[j2]])
    *
    (-1 + hozzarendel[b,s,j] + hozzarendel[b,s+1,j2]);
```

A járatok közötti köztes kilométerek számolására is létre kellett hozni egy korlátozást, ami a `KoztesKmBeallitas1` nevet kapta. Ha j járat után j2 következik a busz menetrendjében és nem egyezik a j járat befejezési helye a j2 járat kezdési helyével, akkor hozzáadódik a `koztesutazas` változó értékéhez a megfelelő távolságadat.

```
s.t. KoztesKmBeallitas1
  {b in Buszok, s in Sorszam, j in Jaratok, j2 in Jaratok: s != maxjarat}:
    koztesutazas[b,s]
    >=
    tav[hova[j],honnan[j2]]
    *
      (-1 + hozzarendel[b,s,j] + hozzarendel[b,s+1,j2]);
```

Azonban az előbb említett korlátozás túl sok korlátozást generál, ami nagyban lassítja a modell végrehajtását, ezért két új korlátozás is bevezetésre került. Ezek a korlátozásokból

generált táblázat sorainak és oszlopainak egybevonásával segítik a modell gyorsítását (`KoztesKmBeallitas2`, `KoztesKmBeallitas3`).

```
s.t. KoztesKmBeallitas2
{b in Buszok, s in Sorszam, j in Jaratok:s!=maxjarat}:
  koztesutazas[b,s]
  >=
  (sum {j2 in Jaratok} tav[hova[j],honnan[j2]] * hozzarendel[b,s+1,j2])
  -
  (max{j2 in Jaratok} tav[hova[j],honnan[j2]]) * (1 - hozzarendel[b,s,j]);
```

```
s.t. KoztesKmBeallitas3
  {b in Buszok, s in Sorszam, j2 in Jaratok:s!=maxjarat}:
  koztesutazas[b,s]
  >=
  (sum {j in Jaratok} tav[hova[j],honnan[j2]] * hozzarendel[b,s,j])
  -
  (max{j in Jaratok} tav[hova[j],honnan[j2]]) * (1 - hozzarendel[b,s+1,j2]);
```

Cél

A cél itt is az átmeneti kilométerek minimalizálása volt, mint Modell1A-ban, azonban ennek a modellnek a célfüggvényében ezt az összeget a `koztesutazas` nevű változó segítségével fejeztem ki.

```
minimize Koztestav:
  sum{b in Buszok, s in Sorszam: s != maxjarat}
    koztesutazas[b,s];
```

Összegzés

Mivel Modell 1B még a gyorsító korlátozások ellenére is sokkal lassabban működött (ezt majd a Tesztelés fejezetben jobban kifejtem), mint Modell 1A, így az először bemutatottat fejlesztettem tovább.

Feladatosztály 2– Járatok közötti mozgás minimalizálása depóhelyekkel

Ebben az esetben már minden egyes busz rendelkezik egy olyan hellyel ahonnan a járatok előtt elindul és ahova a járatok elvégzése után visszatér. Azonban még nem vettem figyelembe az olyan buszokra jellemző attribútumokat, mint például az üzemanyag, ami jelen esetben mivel elektromos buszjáratokról van szó, az elektromos áram. A későbbiekben ez a tulajdonság töltöttségként fog megjelenni. Jelen esetben még a járatok elvégzéséhez szükséges üzemanyagszint meglétét feltételeztem, tehát még továbbra sem foglalkoztam a tankolással (töltéssel). A második feladatosztály célja az elsőhöz hasonlóan az üresjáratok összes távolságának minimalizálása volt.

Bemeneti adatok

Az első feladatosztály bemeneti adatait ebbe az osztályba is átemeltem, azonban itt már minden buszhoz hozzárendeltem egy `depo` nevű paramétert, ami egy `Helyek` halmazbeli helyet jelöl, annak jelzéseképp, hogy honnan kezdi, és hol fejezi be a járatait egy busz.

Modell 2

Változók

Modell 1A-hoz képest kettővel több, azaz négy bináris változóval dolgoztam Modell 2-ben.

A két új bináris változó, amit bevezettem az `elsojarat{Jaratok,Buszok}` és az `utolsojarat{Jaratok,Buszok}` nevet viselik. Azért hoztam őket létre, hogy legyen olyan kapcsolóm, amivel ki tudom fejezni, azt hogy egy adott járat egy busznak az első illetve az utolsó járata-e, vagyis elsőként illetve utolsóként azt a járatot végzi-e el az adott busz. A depóból induló illetve ide érkező járatot is vehetjük első illetve utolsó járatnak. Létrehozásukkal átláthatóbbá vált a modell. Ezek az új változók is a `hozzarendel` változó értéke szerint kaptak értéket a modellben.

Korlátozások

Az alapmodellhez (Modell 1A) hasonlóan itt is szükség van a `JaratokElvegzese`, az `OsszeferhetetlenJaratok`, és az `AtmenetKorlatozas` nevű korlátozásokra. A modell továbbá tartalmaz az első és utolsó járatokkal kapcsolatos és azokhoz tartozó redundánsan működő korlátozásokat is. A következőkben ezeket mutatom be.

A buszok első és utolsó járatainak meghatározásához hasonló korlátozások lettek létrehozva, így ezekről párhuzamosan teszek említést. Akkor lehet egy járat első illetve utolsó járata egy busznak, ha azt a járatot az adott busz végzi el. Ellenkező esetben, ha a `hozzarendel[j,b]` változó értéke 0, akkor `ElsoHozzarendeles` és az `UtolsoHozzarendeles` korlátozásokban használt `elsojarat` és `utolsojarat` változók értéke is szükségképpen 0 lesz.

```
s.t. ElsoHozzarendeles {b in Buszok, j in Jaratok}:
  elsojarat[j,b] <= hozzarendel[j,b];
```

```
s.t. UtolsoHozzarendeles {b in Buszok, j in Jaratok}:
  utolsojarat[j,b] <= hozzarendel[j,b];
```

Az előbbiekkel redundáns a következő két korlátozás, melyek szerint csak akkor lehet egy busznak első illetve utolsó járatáról beszélni, ha van bármilyen járat hozzárendelve ahhoz a buszhoz.

```
s.t. ElsoHozzarendeles2 {b in Buszok}:
  sum{j in Jaratok} elsojarat[j,b] <= sum{j in Jaratok} hozzarendel[j,b];
```

```
s.t. UtolsoHozzarendeles2 {b in Buszok}:
  sum{j in Jaratok} utolsojarat[j,b] <= sum{j in Jaratok} hozzarendel[j,b];
```

A `LegfeljebbEgyElso` és `LegfeljebbEgyUtolso` nevű korlátozások szerint minden busznak maximum egy első és utolsó járata lehet.

```
s.t. LegfeljebbEgyElso {b in Buszok}:
  sum{j in Jaratok} elsojarat[j,b] <= 1;
```

```
s.t. LegfeljebbEgyUtolso {b in Buszok}:
  sum{j in Jaratok} utolsojarat[j,b] <= 1;
```

Amennyiben legalább egy járat hozzá van rendelve egy buszhoz, akkor már mindenképpen szükséges a busz első illetve utolsó járatáról beszélni (`SzuksegesElso`, `SzuksegesUtolso`).

```
s.t. SzuksegesElso {b in Buszok}:
  sum{j in Jaratok} elsojarat[j,b] >=
  sum{j in Jaratok} hozzarendel[j,b] / card(Jaratok);
```

```
s.t. SzuksegesUtolso {b in Buszok}:
  sum{j in Jaratok} utolsojarat[j,b] >=
  sum{j in Jaratok} hozzarendel[j,b] / card(Jaratok);
```

Egy olyan j járat, ami egy b buszhoz van rendelve és egy másik hozzárendelt j2 járatnál később kezdődik, nem lehet egy busz első járata, valamint egy olyan j járat, ami egy b buszhoz van rendelve és egy másik hozzárendelt j2 járatnál korábban kezdődik, nem lehet egy busz utolsó járata. Ezeket a feltételeket úgy illesztettem bele a modellbe, hogy két korlátozást hoztam létre, `KesobbiNemElso` és `KorabbiNemUtolso` néven.

```
s.t. KesobbiNemElso
  {b in Buszok, j in Jaratok, j2 in Jaratok: mikortol[j] > mikortol[j2]}:
    elsojarat[j,b] <= (1 - hozzarendel[j2,b]);
```

```
s.t. KorabbiNemUtolso
  {b in Buszok, j in Jaratok, j2 in Jaratok: mikortol[j] < mikortol[j2]}:
    utolsojarat[j,b] <= (1 - hozzarendel[j2,b]);
```

Cél

A cél ugyanaz, mint a Modell 1A-ban megfogalmazott. A célfüggvényben összeszámoltam az összes járatok közötti távolságot, valamint azokat az utakat is, amíg a busz a depóból az első járata kezdési helyére ér, illetve az utolsó járata után a depóba visszaér.

```
minimize Koztestav:
sum {b in Buszok, j1 in Jaratok, j2 in Jaratok}
tav[hova[j1],honnan[j2]] * atmenet[b,j1,j2]
+
sum {b in Buszok, j in Jaratok} elsojarat[j,b] * tav[depo[b],honnan[j]]
+
sum {b in Buszok, j in Jaratok} utolsojarat[j,b] * tav[hova[j],depo[b]];
```

Feladatosztály 3 – Járatok közötti mozgás minimalizálása töltöttség figyelembevételével

Az előzőekben még csak az általános buszokkal kapcsolatos alapokat írtam bele a modelljeimbe (például a járatok buszokhoz való kötelező hozzárendelése, depók bevezetése), azonban jelen osztálytól kezdődően már lépésenként elkezdtem az elektromos buszok tulajdonságait is figyelembe venni. Ilyen attribútum például az a távolság amennyit egy töltéssel meg tud tenni. A buszok fogyasztásával illetve töltésével jelen esetben még nem foglalkoztam, azok már másik feladatosztályokhoz tartoznak. Célként még ebben a feladatosztályban is az átmeneti járatok távolságainak minimalizálását választottam.

Bemeneti adatok

A második feladatosztály továbbfejlesztéseként az összes ottani bemeneti adat itt is szerepel, viszont ehhez a modellhez hozzáadtam két újdonságot is.

Ha egy járatról beszélünk, akkor, az nem feltétlen csak egy két pont közötti távolság lehet, hanem egy ennél hosszabb, kacskaringósabb út is. Ennek a számszerűsítésére hoztam létre a `Jaratok` halmazhoz tartozó `tav2` paramétert, ami egy járat két végpontja közötti valós hosszának kilométeradatát tartalmazza.

Mivel már elektromos buszokról beszélünk, így figyelembe kell venni azt is, hogy minden busznak van egy olyan határértéke, hogy hány kilométert tud menni egyetlen töltéssel. Ennek kapcsán szükség van erre az adatra is, ami ezt a buszokra vonatkozó számot jelzi. is Jelen esetben ez a `maxtav` nevű paraméter.

Modell 3

Változók

Többek között az előző modellbeli négy bináris változót használtam ebben a modellben is, de ezeken felül kellett még egy változó is, ami azt az értéket jelenti, hogy egy busz a járatok elvégzése közben hány kilométert tett meg. Ez a változó a modellben az `osszhasznalat{Buszok}` nevet kapta.

Korlátozások

Alapul szolgáltak a már Modell 1A és Modell 2-ben felépített korlátozások. Azok a járatok elvégzését, az összeférhetetlen járatok kiküszöbölését és a közvetlen átmenetek jelzését biztosították valamint meghatározták a buszok első és utolsó járatait.

Figyelembe kellett venni azonban azt is, hogy egy busz csak annyi kilométert tehet meg amennyit a töltöttsége enged, ezért szükség volt további két korlátozásra. Először is a megtett utakat úgy lehetett kiszámolni, hogy összeadtam a járatok valós hosszát, az átmeneti kilométereket és a depóból illetve a depóba jutás távolság adatait (`OsszhasznalatKiszamitas`), majd pedig felírtam azt a korlátozást, ami kimondja, hogy ez a szám nem lehet nagyobb, mint az egy töltéssel megtehető kilométerek száma (`MaxHasznalat`).

```
s.t. OsszhasznalatKiszamitas {b in Buszok}:
  sum {j1 in Jaratok, j2 in Jaratok}
    tav[hova[j1],honnan[j2]] * atmenet[b,j1,j2]
  +
  sum {j in Jaratok}elsojarat[j,b] * tav[depo[b],honnan[j]]
  +
  sum {j in Jaratok}utolsojarat[j,b] * tav[hova[j],depo[b]]
  +
  sum {j in Jaratok} hozzarendel[j,b] * tav2[j]
  = osszhasznalat[b];
```

```
s.t. MaxHasznalat  {b in Buszok}:
  osszhasznalat[b] <= maxtav[b];
```

Cél

A modell célja továbbra is a köztes kilométerek minimalizálása.

Feladatosztály 4 – Buszok összes energiafogyasztásának minimalizálása a fogyasztás megjelenésével

A harmadik feladatosztályban bevezetésre került töltöttség kiegészítéseként most már a buszok energiafelhasználásával is foglalkoztam, de ebben az esetben még nem vettem figyelembe az újratöltések szükségességét. A fogyasztás megjelenésével ettől az osztálytól kezdve a cél is megváltozott. Az új cél a buszok összes energiafogyasztásának minimalizálása lett.

Bemeneti adatok

Az előzőekben felhasznált bemeneti adatokat ez az osztály is átvette, azonban az ott megismert `maxtav` paraméter helyett a továbbiakban egy `maxtoltes` nevű paraméter fog szerepelni a modellekben. A `maxtoltes` paraméter értéke a buszok kW-ban akkumulátor kapacitását jelenti. Ezt az osztályt a buszoknak az energiahasználatával bővítettem tovább, ezért az adatok között egy ezt leíró paraméterre is szükség volt. Ez a paraméter a `fogyasztas`, ami a buszoknak a kWh-ban mért energiafogyasztását mutatja meg.

Modell 4

Változók

Modell 3 továbbfejlesztéseként az ottani változókra ebben a modellben is szükségem volt. Újdonságként szerepel az `osszfogyasztas{Buszok}` változó, ami csak nemnegatív értéket vehet fel, mivel annak az összes kilowattóra mennyiségnek a tárolására szolgál, hogy az adott busz a járatai elvégzése során összesen mennyi energiát használt fel.

Korlátozások

A már megismert korlátozásokból kivettem a `MaxHasznalat` nevűt, és helyette felírtam két energiafogyasztáshoz kapcsolódó korlátozást. A `FogyasztasKiszamitas` segítségével

meghatároztam a buszok összes energiafelhasználását. Ezt az értéket az `osszhasznalat` és a `fogyasztas` szorzatából számítottam ki.

```
s.t. FogyasztasKiszamitas  {b in Buszok}:
  osszfogyasztas[b] = osszhasznalat[b] * fogyasztas[b];
```

A `MaxFogyasztas` nevű korlátozás szerint a busz nem fogyaszthat több energiát a járatok elvégzése során, mint amennyi az akkumulátor kapacitása.

```
s.t. MaxFogyasztas  {b in Buszok}:
  osszfogyasztas[b] <= maxtoltes[b];
```

Cél

Ettől a modelltől kezdődően már nem a köztes távolságok kilométereinek minimalizálása a cél, hanem a buszok összes energiafelhasználásának minimalizálása.

```
minimize Buszok_osszes_fogyasztasa: sum {b in Buszok} osszfogyasztas[b];
```

Feladatosztály 5 - Buszok összes energiafogyasztásának minimalizálása az újratöltés lehetőségével

Ebben az osztályban már figyelembe vettem (a töltöttség és a fogyasztás mellett) azt is, hogy az elektromos buszok nem csak addig működhetnek, amíg van az akkumulátorukban energia, hanem azt az energiát újra is lehet tölteni. Ennek eredményeképpen a járatok elvégzése között amennyiben időnként a buszoknak töltésre volt szükségük, akkor azt megtehették. Azonban nem állhattak meg akármikor, akárhol és akármennyi ideig, hanem csak megadott időintervallumokban, a megadott helyek valamelyikén. Azokon a helyeken és időszeletekben viszont akár nulláról is feltölthették az akkumulátorukat a maximális kapacitás értékére. Ezen időintervallumok kijelölése a feladatosztály modelljeiben eltérő módon történt. Az ötödik feladatosztályban is a buszok összes fogyasztásának minimalizálása volt a cél.

Bemeneti adatok

A töltéseket úgy vezettem be a modellekbe, mint speciális járatokat, amiknek van egymással megegyező kiindulási és érkezési helye, kezdési és befejezési ideje (időtartama), viszont a menetrendszerinti járatokra jellemző távolságadat esetükben nulla.

Az ötödik feladatosztályhoz tartozó modellek mindegyikénél használtam az összes menetrendszerinti járatot és töltőjáratot is magába foglaló `MindenJarat` halmazt. Modell 5v2-ben a csak menetrendszerinti utasokat is szállító `Jaratok` halmaz is szerepelt a bemeneti adatok között.

Modell 5-ben csak manuálisan vettem fel a `Toltojaratok` halmazba a buszok töltésére szolgáló töltőjáratokat. A `Toltojaratok` halmaz része a `MindenJarat` halmaznak, ezért rendelkezik a járatokra jellemző paraméterekkel is (`honnan`, `hova`, `mikortol`, `meddig`, `tav2`). Értelem szerűen a járatok valós kilométerben mért hosszát jelölő `tav2` paraméter értéke minden töltőjárat esetében 0, illetve a járat kezdési és befejezési helyét kifejező `honnan` és `hova` paraméterek egymással megegyeztek.

Modell 5v2-ben a töltőjáratok bevezetését máshogy oldottam meg.

Modell 5

Változók

Az előzőekben megismert változókhoz képest két új változó került bevezetésre a töltések figyelembevétele miatt. A `toltottsege{j in MindenJarat, b in Buszok}` és `toltottsegu {j in MindenJarat, b in Buszok}` változók azt a nemnegatív értéket tárolják, hogy mennyi az adott busz kilowattban számított töltöttsége az adott járat elvégzése előtt illetve után. Természetesen ez a szám csak kisebb vagy egyenlő lehet, mint a maximális töltöttségi szintet leíró `maxtoltes` paraméter.

Korlátozások

Az `JaratokElvegzese` alapkorlátozást, ami már a Modell 1 óta része a modelleknek, ebben a modellben két részre osztottam a `Toltojaratok` halmaz megjelenése miatt. A `JaratokElvegzese1` nevű korlátozás alapján a menetrend szerinti járatokhoz mindenképpen hozzá kell rendelni egy buszt. Mivel a `Jaratok` halmaz a `Toltojaratok`

halmazt is tartalmazza, ezért csak a halmaz töltőjáratokon kívüli elemeire vonatkozik a korlátozás.

```
s.t. JaratokElvegzese1 {j in (MindenJarat diff Toltojaratok)}:
  sum {b in Buszok} hozzarendel[j,b] = 1;
```

Annak ellenére, hogy az összes menetrend szerinti járatot el kell végezni, nem biztos, hogy minden töltőjáratra szükség van. Egy töltőjárat által nyújtott töltési lehetőséget viszont csak legfeljebb egy busz használhat. Ezt fejeztem ki a `JaratokElvegzese2` nevű korlátozással.

```
s.t. JaratokElvegzese2 {j2 in Toltojaratok}:
  sum {b in Buszok} hozzarendel[j2,b] <= 1;
```

Mivel ebben a feladatosztályban sor került a töltések bevezetésére, így már szükségessé vált néhány töltöttségi szinthez kapcsolódó korlátozás is. Ezek mindegyike „nagy M” korlátozás, tehát az `M` paraméter ezekben is megjelent. Az ütemezés elején a depóból kell a busznak legalább annyi energiájának lenni, hogy eljusson az első járat kezdési helyére, és az ütemezés végén legalább annyi energiája kell, hogy legyen, hogy visszaérjen a depóba. Ezért írtam fel a `DepobolToltottseg` és a `DepobaToltottseg` nevű úgynevezett „nagy M” korlátozásokat. Ezekhez a korlátozásokhoz szükséges volt felvenni egy kellően nagy értékű M paramétert.

```
s.t. DepobolToltottseg {b in Buszok, j in MindenJarat}:
  toltottsege[j,b]
  <=
  maxtoltes[b] - tav[depo[b],honnan_minden[j]] * fogyasztas[b]
  + M * (1 - elsojarat[j,b]);
```

```
s.t. DepobaToltottseg {b in Buszok, j in MindenJarat}:
  toltottsegu[j,b]
  >=
  tav[hova_minden[j],depo[b]] * fogyasztas[b]
  - M * (1 - utolsojarat[j,b]);
```

Ha egy b busz, egy j1 járat után elvégzi a j2 járatot is, akkor a j2 járat előtti töltöttségi szint a következőképp alakul: a j1 járat utáni töltöttségi szintből ki kell vonni a járatok közötti távolság és a busz fogyasztásának szorzatát. Ez az előtte állapot beállító korlátozás a `JaratElottiToltottseg`.

```
s.t. JaratElottiToltottseg
  {b in Buszok, j1 in MindenJarat, j2 in MindenJarat}:
    toltottsege[j2,b] <= toltottsegu[j1,b]
      -
      tav[hova[j1],honnan[j2]] * fogyasztas[b]
      +
      M * (1 - atmenet[b,j1,j2]);
```

Amennyiben egy b busz elvégezte a j menetrendi járatot, akkor a járat utáni töltöttségét a `JaratUtaniToltottseg1` korlátozás fejezi ki.

```
s.t. JaratUtaniToltottseg1
  {b in Buszok, j in MindenJarat diff Toltojaratok}:
    toltottsegu[j,b] <= toltottsege[j,b]
      -
      tav2[j]*fogyasztas[b]
      +
      M * (1 - hozzarendel[j,b]);
```

Ha egy b busz egy töltőjáratot végzett el, akkor a töltőjárat utáni `toltottsegu` változó értéke a busznak megfelelő `maxtoltes` paraméter értékét veszi fel, tehát a busz teljesen feltöltött akkumulátorral folytatja tovább a járatai végrehajtását. Az előbb említettekre írtam fel a `JaratUtaniToltottseg2` nevű korlátozást.

```
s.t. JaratUtaniToltottseg2 {b in Buszok, tj in Toltojaratok}:
  toltottsegu[tj,b] >= maxtoltes[b] - M * (1-hozzarendel[tj,b]);
```

Cél

A modell célja ugyanaz, mint Modell 4-ben. A buszok összes energiafogyasztásának minimalizálása.

Modell 5v2

Változók

Ebben a modellben az előzőhöz képest eltérő módon vettem fel a töltőjáratokat. Már nem egy bemeneti adathalmazként kezeltem őket, hanem a modell által generáltattam. Egy `Toltohelyek` nevű halmazt jelöltem ki azon helyeknek, ahol a buszoknak lehetőségük van az akkumulátorukat feltölteni. Kijelöltem az időhorizontot az `IH` paraméterrel, ami jelen esetben a legkésőbbi befejezési idővel rendelkező járat vége. Egy `idoszelet` nevű paraméterrel osztottam fel az időhorizontot. Az előzőek segítségével határoztam meg a `Toltojaratok` halmazt, melynek elemei úgy néznek ki, hogy egy t töltőhely, egy alulvonás és a megfelelő időszelete az idősíknak. (Például egy A töltőhelyen a harmadik időszelet töltőjáratának jelölése: A_3).

```
set Toltojaratok := setof {t in Toltohelyek, s in 1..(IH/idoszelet)}
 t & '_' & s;
```

Ebben a modellben a `MindenJarat` nevű halmazt a menetrendszerinti járatokat tartalmazó `Jaratok` és a töltőjáratokat tartalmazó `Toltojaratok` halmazok uniójának halmazaként használtam. Viszont így arra, is szükségem volt, hogy legeneráljam a `Toltojaratok` `Jaratok`-hoz kapcsolódó paramétereit. Először is felvettem egy `t_hely` és egy `t_szam` nevű paramétert a töltőjáratok helyének és időszeletének jelölésére. Ezután pedig létrehoztam az összes járatra jellemző `_minden` és a csak a töltőjáratokhoz tartozó `_tolto` végződésű paramétereket. Ezek értelem szerűen a járatokhoz a `honnan`, `hova`, `mikortol`, `meddig`, `tav2`, míg a töltőjáratokhoz a `_tolto` végűeket rendelik.

```
param honnan_tolto {t in Toltojaratok} symbolic:= t_hely[t];
param honnan_minden{m in MindenJarat} symbolic :=
  if m in Jaratok
    then honnan[m]
  else honnan_tolto[m];
```

```
param hova_tolto {t in Toltojaratok} symbolic:= t_hely[t];
param hova_minden{m in MindenJarat} symbolic:=
  if m in Jaratok
    then hova[m]
  else hova_tolto[m];
```

```
param mikortol_tolto {t in Toltojaratok}:= (t_szam[t]-1)*idoszelet;
param mikortol_minden{m in MindenJarat} :=
  if m in Jaratok
    then mikortol[m]
  else mikortol_tolto[m];
```

```
param meddig_tolto {t in Toltojaratok}:= (t_szam[t])*idoszelet;
param meddig_minden{m in MindenJarat} :=
  if m in Jaratok
    then meddig[m]
  else meddig_tolto[m];
```

```
param tav2_tolto {t in Toltojaratok}:= tav[t_hely[t],t_hely[t]];
param tav2_minden{m in MindenJarat} :=
  if m in Jaratok
    then tav2[m]
  else tav2_tolto[m];
```

Korlátozások

Ebben a modellben nem történt új korlátozás felvétele, csupán a paraméterezettségük változhatott. A csak menetrendszerinti járatokra vonatkozó (`JaratokElvegzese1` és `JaratUtaniToltottseg1`) korlátozások a `Jaratok` halmaz elemeire lettek érvényesek, a csak töltőjáratokra vonatkozó (`JaratokElvegzese2` és `JaratUtaniToltottseg2`) korlátozások pedig a `Toltojaratok` halmaz elemeire. Minden más járatokhoz köthető korlátozás a `MindenJarat` halmazt használta továbbra is paraméterül.

Cél

A célfüggvény megegyezik a Modell 5-ben felírttal.

Modell 5v3

Ez a modell csupán annyiban különbözik Modell 5v2-től, hogy tartalmaz két a modell gyorsítását célzó korlátozást. Ezek az `UgyanolyanParameteruBuszok1` és `UgyanolyanParameteruBuszok2` nevet kapták.

```
s.t. UgyanolyanParameteruBuszok1
  {b1 in Buszok, b2 in Buszok:
    depo[b1] = depo[b2]
    &&
    maxtoltes[b1] = maxtoltes[b2]
    &&
    fogyasztas[b1] = fogyasztas[b2]
    &&
    b1 < b2}:
  osszfogyasztas[b1] >= osszfogyasztas[b2];
```

```
s.t. UgyanolyanParameteruBuszok2
  {b1 in Buszok, b2 in Buszok:
    depo[b1] = depo[b2]
    &&
    maxtoltes[b1] = maxtoltes[b2]
    &&
    fogyasztas[b1] = fogyasztas[b2]
    &&
    b1 < b2}:
  sum{j in MindenJarat} elsojarat[j,b1] * mikortol_minden[j]
  <=
  sum{j in MindenJarat} elsojarat[j,b2] * mikortol_minden[j];
```