# Modellek

Az előbbiekben ismertetett probléma megoldását több irányból közelítettem meg. Felírtam két modellt, amelyek közül a későbbiekben már csak az egyiket fejlesztettem tovább.

# Modell 1

## Bemeneti adatok

A főbb bemeneti adatok a buszok, a buszok járatai, valamint a járatok kezdő és végpontjául szolgáló helyek. Ezeket az adatokat halmazokba rendezzük, és különböző kiegészítő paramétereket definiálunk 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 távolságot mutatja meg kilométerben megadva. Az *ido* nevű paraméter a két helyiség közötti átlagos átjutási időt jelzi percben számolva.

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 kezdési időpontja (percben), *meddig* - a járat befejezésének időpontja (percben), *jaratszam* – a buszjáratok száma.

Azokat a járatpárokat, amiket nem tud egy busz elvégezni, mivel ütközés lépne fel a végrehajtás és köztes utazások összes időtartamában, 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 j1 és j2 járatot, és j2 járat kezdési időpontja előbb van, mint ahogy egy busz az előző j1 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.

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

## Változók

A modell két bináris változót használ. Azt, hogy melyik járatot ténylegesen melyik busz végzi el, a *hozzarendel{Jaratok,Buszok}* változó értéke jelzi. 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 fejezzük ki. Ha egy b busz elvégez egy j1 járatot és utána közvetlenül a j2 járatot is, akkor az *atmenet[b,j1,j2]* változó értéke 1 lesz, minden más esetben 0.

## 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 azonos buszhoz. Ennek érdekében jött létre az *OsszeferhetetlenJaratok* nevű korlátozás. A *kulonbozobusz* halmazból vett járatpárt nem lehet egyszerre az adott buszhoz rendelni, legfeljebb az egyiket.

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

Amennyiben két járat hozzá van rendelve egy buszhoz és nincs a két járat között más köztes járat, akkor ez azt jeleni, hogy a busz az egyik járatból a másikba közvetlenül átmegy. Ilyen esetben az *atmenet* változó értéke 1 lesz, egyébként 0. 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];
```

Ez a korlátozás két másikkal tovább élesíthető. (*AtmenetKorlatozas_elesito1*, *AtmenetKorlatozas_elesito2*).

## Cél

A cél a buszjáratok közötti áthaladások távolságainak minimalizálása. A célfüggvény felírásakor szummázni kell a járatok közötti átmenetek kilométeradatait.

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

# Modell 1_2

## Bemeneti adatok

Az előző modellhez hasonlóan ez a változat is használja a *Helyek*, *Jaratok*, *Buszok* halmazokat. A *Helyek* halmaz itt is rendelkezik a *tav* és az *ido* paraméterekkel, a *Jaratok* halmaz a *honnan*, *hova*, *mikortol*, *meddig* nevű paraméterekkel, valamint a *jaratszam* és a *buszszam* adják meg az elvégzendő járatok és a rendelkezésre álló buszok számát.

Újdonságként 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.

## Változók

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;


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

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

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.

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

```
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];
```

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 nagy táblázatnyi korlátozást generál, ami nagyban lassítja a modell végrehajtását, ezért két új korlátozás is bevezetésre került. Ezek a táblázat sorainak és oszlopainak egybevonásával segítik a modell gyorsítását (*KoztesKmBeallitas2*, *KoztesKmBeallitas3*).

### Cél

A cél itt is az átmeneti kilométerek minimalizálása, amit ennek a modellnek a célfüggvényében a *koztesutazas* nevű változó segítségével fejezhetünk ki.

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

Mivel Modell 1_2 még a gyorsító korlátozások ellenére is sokkal lassabban működött, mint Modell 1, így az először bemutatottat fejlesztettem tovább.

## Modell 2

### Bemeneti adatok

Ez a modell az előbbiekben bemutatott két eset közül a Modell 1-et viszi tovább, így többek között az ott megadott bemeneti adatokat is használja. A *Helyek* halmazt a *tav* és *ido*, a *Buszok* halmazt a *buszszam*, valamint a *Jaratok* halmazt a *honnan*, *hova*, *mikortol*, *meddig* és *jaratszam*, paraméterekkel. A járatpárok időbeni ütközésének elkerülését a *kulonbozobusz* halmaz használata teszi lehetővé.

Minden egyes busz rendelkezik egy olyan hellyel (a *Helyek* halmazból) ahonnan a járatok előtt elindul és ahova a járatok elvégzése után visszatér. Ezt a helyet a modellben a buszokhoz rendelt *depo* nevű paraméter jelzi.

Szükséges volt továbbá egy *M* paraméter felvétele, amit a modell az úgynevezett ’nagy M’ korlátozásai használnak.

### Változók

Modell 1-hez képest kettővel több, azaz négy bináris változóval dolgozunk Modell 2-ben. A buszok járatokhoz való hozzácsatolását továbbra is a *hozzarendel{Jaratok,Buszok}*

változó 1 értéke jelzi, valamint az *atmenet{Buszok,Jaratok,Jaratok}* nevű változó használható az egymás után elvégzendő járatok kifejezésére.

A két új bináris változó, amit bevezettem az *elsojarat{Jaratok,Buszok}* és *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. Ebben a modellben a depóból induló illetve ide érkező járatot is vehetjük első illetve utolsó járatnak.

## Korlátozások

Az alapmodellhez 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. Ezeket mutatom be a következőkben.

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 járat, ami a másiknál később kezdődik, és a buszhoz van rendelve, nem lehet egy busz első járata, valamint egy olyan járat, ami a másiknál korábban kezdődik, és a buszhoz van rendelve, nem lehet egy busz utolsó járata. Ezeket a feltételeket úgy illesztettem bele a modellbe, hogy két úgynevezett ’nagy M’ 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] <= 0 + M * (1- hozzarendel[j2,b]);
s.t. KorabbiNemUtolso
  {b in Buszok, j in Jaratok,j2 in Jaratok: mikortol[j]<mikortol[j2]}:
    utolsojarat[j,b] <= 0 + M * (1- hozzarendel[j2,b]);
```

## Cél

A cél ugyanaz, mint a Modell 1-ben megfogalmazott. A célfüggvényben összeszámoljuk 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]];
```

# Modell 3

Az előzőekben még csak az általános buszokkal kapcsolatos alapokat írtam bele a modelljeimbe, azonban jelen modelltől kezdődően már lépésenként elkezdtem az elektromos buszok tulajdonságait is figyelembe venni.

## Bemeneti adatok

Modell 2 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 felvettem egy *maxtoltes* nevű paramétert, ami ezt a buszokra vonatkozó adatot jelzi. A buszok fogyasztásával illetve töltésével jelen modellben még nem foglalkozunk, de majd a későbbiekben lesz arról is szó.

## 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 Modell1 és Modell2-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 összeadtuk 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] <= maxtoltes[b];
```

## Cél

A modell célja továbbra is a köztes kilométerek minimalizálása.