Eötvös Loránd Tudományegyetem
Természettudományi Kar

Elek Tibor Attila

Optimalizálási módszerek órarend készítéséhez

BSc Elemző Matematikus Szakdolgozat

Témavezető:

Bérczi-Kovács Erika

Operációkutatási Tanszék

Budapest, 2014
Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondani témavezetőmnek, Bérczi-Kovács Erikának a rengeteg segítségért és a folyamatosan lelkesítésért, valamint köszönöm azt a végtesen türelmet és jókedvet, amit a szakdolgozat elkészítése során tőle kaptam.

Továbbá hálás vagyok Bodó Alexandrának a LaTeX-ben nyújtotott számtalan segítségért, és persze családomnak és barátaimnak az állandó biztatásért.
<table>
<thead>
<tr>
<th>Tartalomjegyzék</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1. Bevezetés</strong></td>
</tr>
<tr>
<td>1.1. Egészértékű lineáris programozás</td>
</tr>
<tr>
<td>1.2. Ütemezéselmélet</td>
</tr>
<tr>
<td><strong>2. Kerettanterv</strong></td>
</tr>
<tr>
<td>2.1. Feltételek</td>
</tr>
<tr>
<td>2.2. Megoldási módszer</td>
</tr>
<tr>
<td>2.3. Modell vizsgálata</td>
</tr>
<tr>
<td><strong>3. Órarendkészítés</strong></td>
</tr>
<tr>
<td>3.1. Ütemezéssel modellezhető feladatok</td>
</tr>
<tr>
<td>3.1.1. Osztályszintű modell</td>
</tr>
<tr>
<td>3.1.2. Előre lefoglalt időpontok</td>
</tr>
<tr>
<td>3.1.3. Azonos órák probléma</td>
</tr>
<tr>
<td>3.2. Lineáris modell</td>
</tr>
<tr>
<td>3.2.1. Paraméterek</td>
</tr>
<tr>
<td>3.2.2. Változók</td>
</tr>
<tr>
<td>3.2.3. Feltételek</td>
</tr>
<tr>
<td><strong>4. Implementálás</strong></td>
</tr>
<tr>
<td>4.1. Deklarálás</td>
</tr>
<tr>
<td>4.1.1. Paraméterek definiálása</td>
</tr>
<tr>
<td>4.1.2. Több paraméteres táblák</td>
</tr>
<tr>
<td>4.1.3. Beolvasás</td>
</tr>
<tr>
<td>4.1.4. Változók</td>
</tr>
</tbody>
</table>
4.2. Feltételek és célfüggvény

4.2.1. Feltételek

4.2.2. Célfüggvény

4.3. Kiíratás és kimenet

4.3.1. Kiíratás

4.3.2. Kimenet
1. fejezet

Bevezetés

Életünk napi szinten határozzák meg táblázatok, erre az egyik legmindennapibb példa az órarend. Egy órarend tervezésénél rengeteg változót és feltételt figyelembe kell vennünk, például a kötelező órák számát, a tanárok munkaidejét, stb. Minél többet alkalmazunk, annál bonyolultabb és nagyobb számítási erőforrást igényel az órarend megtervezése. Több matematikai módszer létezik már a feladat modelllezésére. Célom ezek vizsgálata és hatékony módszerek keresése egyszerűbb esetekben ütemezésmélet, összetettebb feladatoknál egészértékű lineáris programozás segítségével. Először egy valós feladat, az órarendek gerincét képező kerettanterv matematikai modelllezését mutatom be a GMPL modelllezési nyelvet használó GUSEK IP solverben[1]. E feladat alapját a 14/2013 (IV.05) NGM rendeletben kiadott szakkértsési kerettantervek határozzák meg[3].

Ezt követően az órarendkészítés valósághű modelllezésével foglalkozom. Gyakorlati megvalósítását C++ nyelven, LEMON-ban készített saját programon ismertetem.

1.1. Egészértékű lineáris programozás

A lineáris programozás az optimalizálás egy eszköze, melynek számos alkalmazása ismert.

1.1.1. Definíció. Az olyan feltételes szélsőérték-feladatot, amelyben a feltételek lineáris egyenletek és egyenlőtlenségek, és egy lineáris függvény szélsőértékét keressük, lineáris programozási (LP) feladatnak nevezzük.
A \in \mathbb{R}^{m \times n}, \quad b \in \mathbb{R}^m, \quad Ax \leq b \quad \text{max } x \rightarrow x \in \mathbb{R}^n \quad \text{LP} \\
\rightarrow x \in \mathbb{Z}^n \quad \text{IP}

Egészértékű lineáris programozási (IP) feladatról beszélünk, ha a változóink csak egészértékűk lehetnek. Az olyan változókat, amelyek csak 0 vagy 1 értéket vehetnek fel, bináris változóknak nevezzük.

Egyszerű példa egy ilyen feladatra:
Adott két változó $x_1 > 0$, $x_2 < 10$ és egy $z$ egészértékű paraméter. A feladatunk egy célfüggvény maximumát megtalálnunk a következő feltételek mellett:

\[
\begin{align*}
x_1 - 5 & \leq x_2 \\
0 & \leq 2x_1 + x_2 \leq 25 \\
x_1 + x_2 & \leq z
\end{align*}
\]

Célfüggvényünk:

\[5x_1 + 3x_2 \rightarrow \text{max}\]

Ehhez és hasonló feladatok megoldásához számos módszer és eszköz létezik. Ilyen eszköz például a GUSEK nevű IP solver, ami a modellezéshez GMPL IP modellezési nyelvet használ.

A GUSEK-ben először a paraméterek és a változók meghatározásával kezdjük. Ezek után érdemes létrehozni a célfüggvényt, majd elé írni a feltételeket, amik alapján történik majd az optimalizálás. Ezt a szakaszt a \texttt{solve}; paranccsal záthatjuk le. A programkód további részében még megadhatjuk a kiíratási paramétereket C++-ban is megtalálható parancsokkal. A végén \texttt{data}; kezdettel megadhatjuk a futtatáshoz szükséges adatokat. Az előző példafeladat így néz ki GUSEK-ben:

\begin{verbatim}
var x1,>=0;
var x2,<=10;
param z, integer;
\end{verbatim}
Deklaráltuk a változókat és korlátaiakat, illetve a paramétert. Ha szükséges meghatározhatjuk ezek típusait. Itt a $z$ paraméterünk egész típusú.

\[
\text{s.t. felt1: } x_1 - 5 \leq x_2; \\
\text{s.t. felt2: } 0 \leq 2 \cdot x_1 + x_2 \leq 25; \\
\text{s.t. felt3: } x_1 + x_2 \leq z; 
\]

Az „s.t.” a „such that” angol kifejezésre utal. Ezek után írjuk a feltételeket, melyeknek nevét is adhatunk:

\[
\text{maximize obj: } 5 \cdot x_1 + 3 \cdot x_2; \\
solve; \\
printf "\n"; \\
printf "Célfüggvény értéke: "; \\
printf \text{obj}; \text{printf } "\n"; \\
printf "x_1 változó értéke: "; \\
printf x_1; \text{printf } "\n"; \\
printf "x_2 változó értéke: "; \\
printf x_2; \text{printf } "\n"; \\
data; \\
\text{param } z := 16; \\
end; 
\]

A data részben értéket adtunk a $z$ paraméternek, így az (1.3)-as feltétel a megadott érték szerint teljesül.

Célfüggvényt maximalizálhatunk és minimalizálhatunk. Ebben az esetben maximalizálnunk kellett. Futtatás után a maximum értékét az obj elemben tárolja, amelyre a kiírás során tudunk hivatkozni. A program kimeneteként a következőt kapjuk:

Célfüggvény értéke: 66 \\
x_1 változó értéke: 9 \\
x_2 változó értéke: 7
1.2. Ütemezéselmélet

Az ütemezési feladatok célja olyan időbeosztás meghatározása, amely a különféle feltételek figyelembe vételével optimális, vagy ahhoz közeli [2]. Ezekben munkákat (jobs) ütemezünk gépeken (machines) bizonyos feltételek mellett, hogy a célfüggvényt (objective function) optimalizáljuk. Általános szabály, hogy egy időben egy gépen egyszerre csak egy munka elvégzése történhet, és egy munka egy időben csak egy gépen futhat. A gépek, a munkák és a célfüggvény is többfélek lehetnek, ezért az ütemezési feladatokat egyedi részre tagolva írjuk fel.

<table>
<thead>
<tr>
<th>Gépek</th>
<th>Munkák</th>
<th>Célfüggvény</th>
</tr>
</thead>
</table>

A felírás első felében a gépek számát és típusát határozzuk meg. Ezt követően a munkák speciális tulajdonságait adjuk meg. Végül az ütemezés célja szerint írjuk a célfüggvényt:

\[ P_5|p_j \in \{1\}| \sum C_j \quad (1.5) \]

(1.5) egy olyan ütemezési feladatot határozz meg, amely szerint 5 párhuzamos \((P)\) gépre kell, azonosan 1 megmunkálási idejű munkákat ütemezünk úgy, hogy a befejezési idők összegét minimalizáljuk.
2. fejezet

Kerettanterv

A kerettantervek a szakmai képzéseket gerincét képezik. Meghatározzák, hogy egy bizonyos képzési évben mely tantárgyat és mekkora heti óraszámban tanítanak. Számos feltételt tartalmazhatnak, így egészen nagy erőforrást igénylő feladattá válhat az elkészítésük.

2.1. Feltételek


A képzés minden évben 36 hétből áll. Az összóraszámóból levonásra kerül egy szabadság. Ez az, hogy a hetek óraszámait meg tudjuk határoznia. Ezt az eltérést a megoldás során minimalizáljuk, így lehet nulla is, tehát a szabadság értéke pontos lesz. A képzéshez tartozik nyári gyakorlat, de ez nem befolyásolja az év közbeni tanrendet, csupán az összóraszámóból kell levonni. Ezt a feladat legelején megteszük.
2.2. Megoldási módszer

Az operáció kutatásban megismert IP modellezéssel megoldhatjuk a feladatot, a GUSEK IP solver használatával.

Paraméterek:

param hetek, integer, > 0; //Hetek száma

param kepzesoraszam, integer, > 0; //Üsszóraszám a képzés során

param nyarigyak, integer, >0; //Nyári gyakorlat óraszáma

param szabadsav, >=0, <1; //Szabadsáv

param szabadsavelt, >=0, <1; //Eltérés a szabadsávtól

- Eltérés a szabadsávtól: mennyire lehet eltérni a szabadsávra megadott százaléktól. Erre azért van szükség, hogy egészértékű megoldást kapjunk.

param elm, >=0, <=1; //Arány

- Elméleti és gyakorlati tárgyak arányának meghatározására használjuk.

param n, integer, > 0; //Tárgyak száma
set T := 1..n; //Tárgyak halmaza

param y, integer, >0; //Évek száma
set Ev := 1..y; //Évek halmaza

param tipus{T} integer, >0, <4; //Típus

- Típus: egy óra lehet elmélet, gyakorlat vagy vegyes. Az órákat típusok szerint csoportokba soroljuk (1, 2, 3).

param epules{T} integer, >0, <4; //Egyénasra épülés

- Egyénasra épülés: megadjuk, hogy egy tárgyra épül valami, vagy előfeltétele egy másik tárgy, vagy egyik sem (1, 2, 3).
param felsokorlat{i in T}, integer; //Felsőkorlát
param alsokorlat{i in T}, integer; //Alsókorlát

- Felső- és Alsókorlát, amikkel meghatározhatjuk, hogy egy tárgyból minimum és maximum hány óra lehet egy héten.

Ezek a paraméterek mind egészértékek és pozitívak, kivéve az elméleti-gyakorlati arányt, a szabadsávot, és az eltérést a szabadsávotól, hiszen ezeket százalékosan adjuk meg.

Változók:

var osszora, integer, = kepzesoraszam-nyarigyak;

- A képzés óraszáma a nyári gyakorlat nélkül.

var elt, >=0, <=0.03;

- Arányok eltérése: az input paraméterként megadott elméleti-gyakorlati aránytól való eltérést adja meg.

var vegyesek, integer >=0;

- Vegyesek összege: a vegyes, azaz 3-as típusú órák számának az összege.

var elméletek, integer >=0;

- Elméletek összóraszám: az elméleti, azaz 1-es típusú órák összege.

var osszhetioraszam, integer;

- Valós heti összóraszám: annak az összege, hogy évenként egy héten hány órát tervezünk tény- legesen a képzés ideje alatt.

var x{i in T, j in Ev}, integer, >=0;

- Tárgyak évé szerinti óraszámt az i, j változója: megadja, hogy az i. tárgyból a j. évben hány óra legyen egy héten.

var z{i in T, j in Ev}, binary;

- z[i, j]: bináris változó, ha az i. tárgyból a j. képzési évben van óra, akkor az értéke 1, különben 0.
var delta >=0;

- Delta: ezzel a változóval küszöböljük ki, hogy ne történhessen meg, hogy egy tárgyból nagyon magas az óraszám és a többiből meg alacsony, tehát az óraszámok közötti különbséget csökkentjük vele.

A feltételeket a következőképp fogalmaztuk meg:

s.t. oraszam1: osszhetioraszam*hetek >= (osszora-((szabadsav+szabadsavelt) *osszora));

s.t. oraszam2: osszhetioraszam*hetek <= (osszora-((szabadsav-szabadsavelt) *osszora));

Megadjuk feltételként a szabadsáv eltérésének intervallumát, és egyben kiszámloljuk a valós heti ósszorászámot a tanítási heteknek megfelelően. Ez a változó a célfúggvényben nagy súlytal sze-
repel, így törekszünk arra, hogy a lehető legtöbb óra legyen megtartva.

s.t. zfelt1{t in T}:sum{j in Ev} z[i,j] =1;

A bináris változókból álló táblázatban adjuk meg, hogy a sor összeg 1, azaz egy tárgyat csak egyszer tartanak meg az évek alatt.

s.t. zfelt2{j in Ev}:sum{i in T} z[i,j] <=6;

Minden évre megadjuk, hogy az aktuális évhez tartozó oszlop összeg a bináris táblázatban legfel-
jebb 6 legyen, azaz egy évben ennyi tárgynál nem lehet többet tartani.

s.t. psi{j in Ev}: sum{i in T} x[i,j] <= osszhetioraszam/2-1;

Megszorítást adunk arra, hogy egy évben ne legyen túl sok óra a többihez képest.

s.t. zfelt3{i in T, j in Ev}:x[i,j] <= z[i,j]*felsokorlat[i];

s.t. zfelt4{i in T, j in Ev}:z[i,j]*alsokorlat[i]<= x[i,j];

Összekapcsoljuk a két táblázatot, z[i,j] megadja, hogy mikor tartsuk az órát, x[i,j] pedig hogy mennyit tartsunk belőle.

s.t. zfelt5{i in T}: z[i,1] >= (if epules[i]=1 then 1);
Ha egy tárgy előfeltéte más tárgynak, akkor úgy határozzunk, hogy az a képzés elején, az első évben legyen. Figyeljünk kell, mert ha 6-nál több ilyen tárgyunk van, akkor emelnünk kell a tárgyak számának felső korlátját.

s.t. zfelt6\{i \in T\}: \text{if } \text{euples}[i]=2 \text{ then } z[i,1]=0;

Ha a tárgynak van előfeltétele, akkor semmiképp sem rakjuk a képzés elejére.

s.t. veg: \text{sum}\{i \in T, \; j \in \text{Ev}\} \; x[i,j] = \text{osszhetioraszan};

Így határozzuk meg a tárgyak óraszámainak összességét.

s.t. elmgyak1: \text{vegyesek} = \text{sum}\{i \in T, \; j \in \text{Ev}\} \; \text{if } \text{tipus}[i]=3 \text{ then } x[i,j];

s.t. elmgyak2: \text{elmeletek} = \text{sum}\{i \in T, \; j \in \text{Ev}\} \; \text{if } \text{tipus}[i]=1 \text{ then } x[i,j];

Kiszámoljuk, hogy mennyi a vegyes és az elméleti órák óraszáma, a gyakorlati óraszám ebből triviálisan következik.

s.t. elmgyak3: \text{elmeletek}<=\text{(osszhetioraszan-vegyesek)}.Sequential(\text{elm+elt});

s.t. elmgyak4: \text{elmeletek}>=\text{(osszhetioraszan-vegyesek)}.Sequential(\text{elm-elt});

Itt megadunk egy intervallumot, hogy az elméleti óraszám mennyivel térhet el a paraméterként megadott aránytól. Ez a célfigyelvényben 0-nál kisebb súlyal szerepel, hogy a minél kisebb legyen.

s.t. difference\{i \in T, \; j \in \text{Ev}, \; k \in T, \; l \in \text{Ev}\}: x[i, j] <= x[k,l] + \delta;

Megadjuk, hogy a táblázatban két érték legfeljebb deltával térhet el, a delta a célfigyelvényben 0-nál kisebb súlyal fog szerepelni, így lesz minél egyenletesebben szétosztva a valódi heti óraszám.

maximize obj:10000*\text{osszhetioraszan}– 100 * \delta - \text{elt};

Végül maximalizáljuk a célfigyelvénnyt. A delta változónak −100-as szorzót adunk, így minimalizálni fog, és nagyobb súlytal számít. A heti összórászámokat maximalizáljuk, ennek az együtthatója kellően nagy, ezzel adjuk meg, hogy az a legfontosabb, hogy minél több óra legyen. Az eltérést az elmélet-gyakorlat arányától minél kisebbnek szeretnénk, hogy az inputban megadott értéktől minél kisebb legyen a különbség.

Kiíratáshoz egy sablonra van szükségünk, amelyben meg tudjuk jeleníteni, hogy valamelyik tanítási évben egy tárgyat hány órában kell tanítani. Ennek eleget tesz a következő kiíratási kód-sorozat:

13
printf "\n";
printf "Tárgy | 1e/2gy/3v | \
8\n| Év1 | Év2 | Év3 |\n\n"
printf{in T} "%10d %6d %9d || %2d | %3d | %3d |\n\n", i,tipus[i], tipus2[i],
  x[i,1], x[i,2], x[i,3];
printf "----------------------------------------------\n"
printf "Total: %10g %5g %6g %5g %5g %10g\n", sum{i in T, j in Ev} x[i,j],
  osszora, sum{i in T} x[i,1], sum{i in T} x[i,2],sum{i in T} x[i,3], elt;
printf "\n";
Kimenetként egy táblázatot fogunk kapni, melynek oszlopaiban az évek, soraiban pedig a tárgyak
jelennek meg.

| Tárgy | 1e/2gy/3v | \
8\n| Év1 | Év2 | Év3 |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>11</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>12</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

Total: 60 2340 24 14 22

Model has been successfully processed
>Exit code: 0 Time: 2.820

Ez a kimenet az alábbi paraméterek eredménye:

param hetek := 36;
param kepzesoraszam := 2640;
param szabadsav := 0.09;
param szabadsavelt := 0.03;
param n := 12;
param tipus :=
param epules :=
param y := 3;
param nyarigyak := 300;
param elm := 0.3;
param felsokorlat := 20;
2.3. Modell vizsgálata


<table>
<thead>
<tr>
<th>Tárgyak száma</th>
<th>Idő (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 tárgy</td>
<td>0,326</td>
</tr>
<tr>
<td>11 tárgy</td>
<td>0,335</td>
</tr>
<tr>
<td>12 tárgy</td>
<td>3,829</td>
</tr>
<tr>
<td>13 tárgy</td>
<td>11,362</td>
</tr>
<tr>
<td>14 tárgy</td>
<td>127,433</td>
</tr>
</tbody>
</table>

2.1. táblázat. Futási idők a tárgyak számának megfelelően

A súlyok nagysága is komoly hatással van a gyorsaságra. Ha nagy súlyt adunk a delta változónak, lassul, ha kis súlyt adunk a osszhetioraszam változónak akkor gyorsul a futási sebesség. A mérések során a célfüggvényben a delta 100, az osszhetioraszam 10 000 súlyal szerepel.

<table>
<thead>
<tr>
<th>delta változó</th>
<th>Idő (s)</th>
<th>osszhetioraszam változó</th>
<th>Idő (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 · delta</td>
<td>2,29</td>
<td>1 · osszhetioraszam</td>
<td>30,145</td>
</tr>
<tr>
<td>10 · delta</td>
<td>2,202</td>
<td>10 · osszhetioraszam</td>
<td>19,12</td>
</tr>
<tr>
<td>100 · delta</td>
<td>3,813</td>
<td>100 · osszhetioraszam</td>
<td>12,681</td>
</tr>
<tr>
<td>1 000 · delta</td>
<td>5,353</td>
<td>1 000 · osszhetioraszam</td>
<td>6,991</td>
</tr>
<tr>
<td>10 000 · delta</td>
<td>7,638</td>
<td>10 000 · osszhetioraszam</td>
<td>3,9</td>
</tr>
<tr>
<td>100 000 · delta</td>
<td>25,898</td>
<td>100 000 · osszhetioraszam</td>
<td>5,684</td>
</tr>
<tr>
<td>1 000 000 · delta</td>
<td>24,572</td>
<td>1 000 000 · osszhetioraszam</td>
<td>3,707</td>
</tr>
<tr>
<td>10 000 000 · delta</td>
<td>33,098</td>
<td>10 000 000 · osszhetioraszam</td>
<td>5,678</td>
</tr>
</tbody>
</table>

2.2. táblázat. Futási idők súlyoknak változtatásával (a másik két változó súlya változatlan)
Ezek változtatása a célkitűzést is változtatják, így előfordulhat, hogy számunkra kevésbé kedvező eredményt kapunk.

A leghatékonyabban úgy tudjuk a programot gyorsítani, ha meghatározzuk, hogy egy tárgyból maximálisan mennyi lehet egy héten, azaz a \( \text{delta} \) változónak adunk egy felső korlátot. Ezt azonban komolyan át kell gondolni, ugyanis előfordulhat, hogy a korlát megadása után nem lesz megoldása a feladatnak.

<table>
<thead>
<tr>
<th>Felsőkorlát</th>
<th>Idő (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Nincs megoldás</td>
</tr>
<tr>
<td>7</td>
<td>0,211</td>
</tr>
<tr>
<td>8</td>
<td>0,322</td>
</tr>
<tr>
<td>9</td>
<td>0,325</td>
</tr>
<tr>
<td>10</td>
<td>0,335</td>
</tr>
<tr>
<td>11</td>
<td>0,437</td>
</tr>
<tr>
<td>12</td>
<td>2,402</td>
</tr>
<tr>
<td>13</td>
<td>2,08</td>
</tr>
<tr>
<td>14</td>
<td>1,209</td>
</tr>
<tr>
<td>15</td>
<td>4,592</td>
</tr>
<tr>
<td>16</td>
<td>2,41</td>
</tr>
<tr>
<td>17</td>
<td>5,134</td>
</tr>
<tr>
<td>18</td>
<td>5,028</td>
</tr>
</tbody>
</table>

2.3. táblázat. Futási idő a felső korlátnak megfelelően
3. fejezet

Órarendkészítés


3.1. Ütemezéssel modellezhető feladatok

Az általános iskolák alsó tagozatosainak az órarendjei egészen egyszerűen akár egy papírlapon is könnyen elkészíthetők. Általában minden osztálynak 1 tanítója van, és minden óra 1 teremben zajlik. Célunk úgy beosztani az órákat, hogy a leghosszabb tanítási nap minél rövidebb legyen.

3.1.1. Osztályszíntű modell

Ha adott $n$ tanítási nap és $k$ heti óraszám, és osztályszíntű a modell, akkor az ütemezéselméleti megfelelője egy $n$ párhuzamos gépes feladat, melyben az átlagos átfutási időt minimalizáljuk. Az $n$ párhuzamos gépet megfeleltetjük a napoknak, hiszen valamelyik napon tanítani kell egy adott órát, tehát $n$ nap alatt el kell végezni minden munkát. Külön feltételnek vesszük, hogy minden munka elvégzési ideje egyenlő és nagyobb 0-nál, mivel minden óra időtartama azonosan 1 egység. Elegendő egy olyan beosztást tekinteni, melyben minden munka vagy óra valamelyik $[t-1,t]$ időintervallumhoz van hozzárendelve, ahol $t$ pozitív egész. Optimálisnak azt mondjuk, amikor a tanítás minden nap a lehető legkorábban befejeződik. Így heti 5 tanítási nap esetén:

$$P5|p_j \in \{1\}| \sum C_j$$ (3.1)
A feladat megoldása nagyon egyszerű. Bár mely algoritmus, amely állási idő, azaz lyukas óra, nélkül osztja be a munkákat jó ütemezést ad. A LEKIN nevű oktatási célú programban, könnyen vizualizálni is lehet az eredményt [4]. A feladat típusa, a gépek száma és a munkák paraméterei megadása után választhatunk algoritmust, ami alapján ütemezni szeretnénk a problémát. Futtatás után Gantt-diagramként láthatjuk az eredményt, és számos információt kinyerhetünk az ütemezésről. Több algoritmust is futtathatunk egymás után, majd ezeket az általános célfüggvények szerint össze is tudjuk hasonlítani. Hogy ne legyen túl egyszerű a modell, készítsünk olyan órarendet, amelyben hetente kétszer a magyar órák egymás után jönnek, és amelyben a hatékony-ság érdekében nincs matematika óra a 4. óra után. A feladatunk így a következő ütemezésnek felel meg:

\[ P5|p_j \in \{1, 2\}, d_j| \sum C_j \] (3.2)

A LEKIN-ben erre a feladatra megfelelő algoritmus az EDD, amely határidők szerint nemsőkkenő sorra rakja a munkákat, azaz órákat.

A képen látható egy órarendszerű ütemezés. A futtatott EDD algoritmus jó eredményt adott. A szigorúbb határidő nélküli órák az algoritmus szempontjából mind azonos tulajdonságokkal bírnak, ezért azok mohón, megadási sorrendben lettek rendezve.

3.1.2. Előre lefoglalt időpontok

Abban az esetben, amikor már az órarendben egyes óráknak már fix időpontja van (pl.: testnevelés), még továbbra is felírható a feladat az ütemezéselmélet alapján, azonban ki kell egészülnie
néhány feltétellel. A fix órák meghatározásához használhatjuk a rendelkezésre állási és határidőket. Így pontosan meg tudjuk adni, hogy egy munka mikortól végezhető, és meddig kell elvégezni, tehát meghatározhatjuk, hogy egy óra mikortól kezdődhet, és mikorra kell befejeződnie. Ezzel pontosan meg tudunk határozni egy időintervallumot. Természetesen ehhez fel kell tennünk, hogy nincs egy nél több munka azonos intervallumra kiosztva. A szabadon ütemezhető órák rendelkezésre állási ideje minden esetben 0, a határidejük pedig \([k/n]\).

\[
P|r_j, d_j, p_j \in \{1\}| \sum C_j
\]  

(3.3)

3.1.3. Azonos órák probléma

Ha nem szeretnénk, hogy egy tantárgy órái azonos napra essenek, használhatunk párhuzamos gépek helyett független gépeket. Ezeknek az az előnyük, hogy minden gép-munka párra külön megmunkálási időket adhatunk meg. Így \(p_{i,j}\) a \(j\) munka megmunkálási ideje az \(i\) gépen. Ezzel megadhatjuk, hogy mely napokon legyenek vagy ne legyenek tartva az egyes órák. A feladat továbbra is az maradt, hogy mely óra melyik napon legyen megtartva és mikor.

\[
R|r_j, d_j, p_{i,j} \in \{0, 1\}| \sum C_j
\]  

(3.4)

Ez az a pont, amikor már be kell láttnunk, hogy a ütemezésemélet adta eszközök már nem elegendőek, hiszen egy jó ütemezés érdékelően már a feladat megadásánál el kell végeznünk a munka nagyrészét. Előre meg kell határoznunk, hogy egy nap milyen órák legyenek. Át kell térnünk más modellre.
3.2. Lineáris modell

Az ütemezéselméletben megemlített feladatok megoldást adhatnak egy egyszerűbb órarendkészítés problémára, de sok oldalról korlátoztak az alkalmazhatóságuk. Célunk egy valóságú modell létrehozása, így egy bizonyos ponton túl már ezek nem felelnek meg a modellezés követelményeinek.

Órarendkészítésnél, a speciális esetek (pl.: egyszerű alsó tagozatos órarend) kivételével, szükséges, hogy minden órarend egyszerre készüljön el. Egy tanintézményben mindig véges, a tanításhoz szükséges, erőforrás áll rendelkezésünkre. Így minden órarend befolyásolhatja a többi órarendet. Ilyen véges erőforrás például a tanárok. Az, hogy egy adott tanár mikor tanít, befolyásolja, hogy milyen tárgyakat tud tanítani, kik és hányan tudják azokat a tárgyakat tanítani, hol lehet azokat tanítani, a tanár mikor ér rá tanítani, stb. Fontos, hogy a befolyásoló tényezők is hatnak egymásra, tehát a célunk egy összetett rendszer optimalizálása.

Célkitűzés:

Akkor lesz a modell valóságú, a triviális feltételeken túl (nincs egy héten 7-nél több tanítási nap), ha:

- megadható, hogy egy tanár milyen tantárgyakat mely osztályoknak taníthat (az általunk vizsgált példában valósághűen minden tanár 2 tárgyat taníthat)
- megadható, hogy egy adott tanár mikor eléhető
- egy tárgyat egy osztálynak csak egy tanár tanít
- az osztályoknak minden kötelező óra adott óraszámban tanítva van

3.2.1. Paraméterek

Sok paraméterrel és változóval kell dolgoznunk egy ilyen modell érdekében, tehát sok adatot kell kezelnünk egyszerre. Ezek az adatok különfélek lehetnek, egészen az egyszerű egyjegyű paraméterektől a sok paraméteres táblákig. Utóbbira elsősorban programozási szempontból van szükség. Szinte minden megoldható lenne nélkülük, de a feladatok elvégzése körülményesebb, és átláthatatlanabb lenne. Továbbá a modellünk nem lenne elég rugalmas a valós alkalmazáshoz.
Egyszerű paraméterek:

<table>
<thead>
<tr>
<th>Paraméter</th>
<th>Típus</th>
<th>Jelölés</th>
</tr>
</thead>
<tbody>
<tr>
<td>tanítás napok száma</td>
<td>egész szám</td>
<td>napok</td>
</tr>
<tr>
<td>egy napon lehetséges órák száma</td>
<td>egész szám</td>
<td>orak</td>
</tr>
<tr>
<td>tantárgyak száma</td>
<td>egész szám</td>
<td>targvszam</td>
</tr>
<tr>
<td>osztályok száma</td>
<td>egész szám</td>
<td>osztalszam</td>
</tr>
<tr>
<td>tanárok száma</td>
<td>egész szám</td>
<td>tanarszam</td>
</tr>
</tbody>
</table>

3.1. táblázat. Előre megadott egyszerű paraméterek

Több paraméteres táblák:

<table>
<thead>
<tr>
<th>Paraméter</th>
<th>Indexek</th>
<th>Jelölés</th>
</tr>
</thead>
<tbody>
<tr>
<td>tárgyak adatai</td>
<td>osztályok száma, tantárgyak száma</td>
<td>Kotelezo</td>
</tr>
<tr>
<td>tanárok elérhetősége</td>
<td>tanárok száma, egy napon lehetséges órák száma, tanítás napok száma</td>
<td>Tanarel</td>
</tr>
<tr>
<td>tanár-osztály hozzárendelés</td>
<td>tanárok száma, tantárgyak száma, osztályok száma</td>
<td>Hrend</td>
</tr>
</tbody>
</table>

3.2. táblázat. Több indexű paraméterek

Mindegyik több paraméteres tábla bináris, tehát minden pont értéke 0 vagy 1. Az előre megadott paraméterek szerepelhetnek a programkódban és külön fájiban is.
### 3.2.2. Változók

<table>
<thead>
<tr>
<th>Változó</th>
<th>Indexek</th>
<th>Jelölés</th>
</tr>
</thead>
<tbody>
<tr>
<td>tárgyak órarendjei osztályonként</td>
<td>osztályok száma, tantárgyak száma, egy napon lehetséges órák száma, tanítás napok száma</td>
<td>Targy</td>
</tr>
<tr>
<td>osztályok órarendjei</td>
<td>osztályok száma, egy napon lehetséges órák száma, tanítás napok száma</td>
<td>Orarendbin</td>
</tr>
<tr>
<td>tanár-osztály hozzárendelés</td>
<td>tanárok száma, tantárgyak száma, osztályok száma</td>
<td>HRend2</td>
</tr>
<tr>
<td>tanárok órarendjei</td>
<td>tanárok száma, tantárgyak száma, egy napon lehetséges órák száma, tanítás napok száma</td>
<td>Tanarora</td>
</tr>
<tr>
<td>teremfoglalások</td>
<td>termek száma, egy napon lehetséges órák száma, tanítás napok száma</td>
<td>Teremel</td>
</tr>
</tbody>
</table>

3.3. táblázat. Változók (minden változó bináris)

### 3.2.3. Feltételek

\[
\forall j \in \text{Tanarszam}, \forall i \in \text{Osztalyszam}, \forall t \in \text{Targyszam} : \\
\text{HRend}_{j,i,t} \geq \text{HRend2}_{j,i,t} \\
\tag{3.5}
\]

A \text{HRend} olyan adott táblázatok, amelyekben meghatározott, hogy egy tanár egy bizonyos osztályt mely tárgyakból taníthatja. A \text{HRend2} változóban létrejön, hogy megoldás esetén egy tanár egy osztályt mely tárgyakból tanít. A feltétel alapján egy tanár csak olyan osztály tárgyaihoz lehet hozzárendelve, amelyeket taníthat is.
\[ \forall i \in \text{Osztalyszam}, \forall t \in \text{Targyszam} : \sum_{j \in \text{Tanarszam}} H\text{Rend}_{2,j,i,t} = 1 \quad (3.6) \]

E feltétel szerint minden osztály minden tárgyához kell, hogy legyen pontosan egy tanár rendelve.

\[ \forall i \in \text{Osztalyszam}, \forall t \in \text{Targyszam} : \sum_{o \in \text{Orak}} \sum_{n \in \text{Napok}} T\text{argy}_{i,t,o,n} = \text{Kotelezo}_{i,t} \quad (3.7) \]

A \text{Targy} olyan táblázatok, amelyek egy osztály egy tárgyának az órarendjei. A \text{Kotelezo} tartalmazza, hogy egy osztálynak egy tárgyát pontosan melyik órászámban kell tanítani. Az osztályok tárgyait a megadott órászámban kell tanítani, tehát az osztály tárgyainak az órarendjeiben, annyi egyesek kell lennie, amennyi a tárgy kötelező órászáma.

\[ \forall i \in \text{Osztalyszam}, \forall o \in \text{Orak}, \forall n \in \text{Napok} : \sum_{t \in \text{Targyszam}} T\text{argy}_{i,t,o,n} \leq \text{Orarendbin}_{i,o,n} \quad (3.8) \]

Az \text{Orarendbin}-ben bináris táblázatokban vannak az osztályok órarendjei. Egy osztály tárgyainak órarendjeinek az összege, az osztály órarendjét adják. Minden időpontban, ha van egy tárgynak órája, akkor az osztálynak lesz órája, ha nincs, akkor nem lesz. Ezzel a feltétellel meghatározzuk azt is, hogy egy osztálynak ne legyen több tárgya azonos időpontban.

\[ \forall j \in \text{Tanarszam}, \forall o \in \text{Orak}, \forall n \in \text{Napok} : \sum_{t \in \text{Targyszam}} T\text{anarora}_{j,t,o,n} \leq T\text{anarel}_{j,o,n} \quad (3.9) \]

A \text{Tanarel}-lel előre megadott táblák határozzák meg, hogy egy bizonyos tanár mikor tarthat órát, a \text{Tanarora}-ban pedig pontosan tároljuk, hogy egy tanár mikor, melyik osztálynak és milyen tárgyból tart órát. Egyszerűen csak akkor tart órát, ha tarthat is.

\[ \forall j \in \text{Tanarszam}, \forall i \in \text{Osztalyszam}, \forall t \in \text{Targyszam}, \forall o \in \text{Orak}, \forall n \in \text{Napok} : \] \[ H\text{Rend}_{2,j,i,t} + T\text{argy}_{i,t,o,n} \leq 1 + T\text{anarora}_{j,t,o,n} \quad (3.10) \]

A feltétel alapján, egy időpontban egy osztály tárgyából csak akkor lehet órát tartani, ha a tárgyhoz hozzárendelt tanár ráér, tehát tarthat órát.
\[ \forall t \in Targszam, \forall o \in Orak, \forall n \in Napok: \]
\[ \sum_{j \in Tanarszam} \text{Tanarora}_{j,t,o,n} = \sum_{i \in Osztalyszam} \text{Targy}_{i,t,o,n} \quad (3.11) \]

Megbízonyosodunk róla, hogy ha a tanár hozzá van rendelve egy osztály tárgyához, akkor minden egyik órát az osztály azon tárgyából ő tartja.

A (3.2) elején megfogalmazott célkitűzéseket a modellünk így teljesíti:

- megadható, hogy egy tanár milyen tantárgyakat mely osztályoknak taníthat: Ezt a HRent több indexű paraméterrel és a (3.6) feltétellel biztosítjuk.
- megadható, hogy egy adott tanár mikor elérhető: A Tanarel több indexű paraméterben meghatározott elérhetőséget a (3.11) és (3.12) feltételekkel használjuk.
- egy tárgyat egy osztálynak csak egy tanár tanít: (3.7) az osztály-tárgy-tanár hozzárendelést így szabályozza.
- az osztályoknak minden kötelező órát adott óraszámban tanítják: (3.8) feltétel az órarend alapjában teljesíti a követelményt.

Egy olyan iskolazintős órarend a célunk, ami elősegíti, hogy minden osztálynak minden nap a lehető leghamarabb befejeződően a tanítás. Ennek megfelelően a célfüggvényünk

\[ \forall i \in Osztalyszam: \]
\[ \sum_{o \in Orak} \sum_{n \in Napok} (\text{Orarendbin}_{i,o,n} \cdot o^2) \rightarrow \min \quad (3.12) \]

Az Orarendbin\((k,i,j)\) változó táblázatok értékeit úgy választjuk 0 és 1 közül, hogy a négyzetes sorindexssel vett szorzatok összege minél kisebb legyen.
4. fejezet

Implementálás

Erré a feladatra gyakorlati szempontokból már nem a GUSEK optimalizáló programot (keret- tanterv feladat) fogjuk használni, hanem C++-ban írjuk meg a programot, a LEMON(Library for Efficient Modeling and Optimization in Networks) nyílt forrású C++ template library segítségével, amely lehetővé teszi számunkra solving használatát [5].

4.1. Deklarálás

4.1.1. Paraméterek definiálása

#define targyszam 6
#define orak 8
#define napok 5
#define tanarszam 6
#define osztalyszam 4

4.1.2. Több paraméteres táblák

vector<vector<int>> Kotelezo; //Kotelezo[][] megadott
Kotelezo.resize(targyszam);
for (int i = 0; i < targyszam; ++i) {
    Kotelezo[i].resize(osztalyszam);
}

vector<vector<vector<bool>> > > HRend; //HRend[][][] *megadott
HRend.resize(tanarszam);
for (int i = 0; i < tanarszam; ++i) {
    HRend[i].resize(targyszam);
for (int j = 0; j < targyszam; ++j)
    HRend[i][j].resize(osztalyszam); }

vector<vector<vector<bool> > > Tanarel; //Tanarel[][][] *megadott
Tanarel.resize(tanarszam);
for (int i = 0; i < tanarszam; ++i) {
    Tanarel[i].resize(orak);
for (int j = 0; j < orak; ++j)
    Tanarel[i][j].resize(napok); }

4.1.3. Beolvásás

ifstream HRendel;
string HRendeles;
HRendel.open ("Hozzarendeles.txt",ifstream::in);
if (HRendel.is_open())
{
for (int i = 0; i < tanarszam; ++i) {
for (int j = 0; j < targyszam; ++j) {
for (int k = 0; k < osztalyszam; ++k){
    getline (HRendel,HRendeles);
    if (HRendeles=="1")
        HRend[i][j][k]=1;
    else HRend[i][j][k]=0;}}}
else cout << "A Hozzarendeles.txt file megnyitása sikertelen volt.";

ifstream Tanareler;
string Tanareleres;
Tanareler.open ("Tanarelerhetoseg.txt",ifstream::in);
if (Tanareler.is_open())
{
for (int i = 0; i < tanarszam; ++i) {
for (int j = 0; j < orak; ++j) {

26
for (int k = 0; k < napok; ++k){
    getline (Tanareler,Tanareleres);
    if (Tanareleres=="1")
        Tanarel[i][j][k]=1;
    else Tanarel[i][j][k]=0;}}
}
else cout << "A Tanarelerhetoseg.txt file megnyitasa sikertelen volt."

ifstream Kotelezotargyak;
string Kotelezok;
Kotelezotargyak.open ("Kotelezok.txt",ifstream::in);
if (Kotelezotargyak.is_open())
{
    for (int j = 0; j < osztalyszam; ++j) {
        for (int i = 0; i < targyszam; ++i) {
            getline (Kotelezotargyak,Kotelezok);
            Kotelezo[i][j]=atoi(Kotelezok.c_str());}
    }
}
else cout << "A Kotelezok.txt file megnyitasa sikertelen volt.";

4.1.4. Változók

vector<vector<vector<Mip::Col> > > HRend2; //HRend2[][][]
HRend2.resize(tanarszam);
for (int i = 0; i < tanarszam; ++i) {
    HRend2[i].resize(targyszam);
    for (int j = 0; j < targyszam; ++j) {
        HRend2[i][j].resize(osztalyszam);
    }
}
for (int i = 0; i < tanarszam; ++i) {
    for (int j = 0; j < targyszam; ++j) {
        mip.addColSet(HRend2[i][j]);
        for (int k = 0; k < osztalyszam; ++k) {
            mip.colType(HRend2[i][j][k], Mip::INTEGER);
            mip.colLowerBound(HRend2[i][j][k], 0);
            mip.colUpperBound(HRend2[i][j][k], 1);}}
}
vector<vector<vector<vector<Mip::Col> > > > Targy;  //Targy[][][][]
Targy.resize(osztalyszam);
for (int k = 0; k < osztalyszam; ++k) {
    Targy[k].resize(targyszam);
    for (int i = 0; i < targyszam; ++i) {
        Targy[k][i].resize(orak);
        for (int j = 0; j < orak; ++j)
            Targy[k][i][j].resize(napok);
    }
}
for (int i = 0; i < osztalyszam; ++i) {
    for (int j = 0; j < targyszam; ++j) {
        for (int k = 0; k < orak; ++k) {
            mip.addColSet(Targy[i][j][k]);
            for (int l = 0; l < napok; ++l) {
                mip.colType(Targy[i][j][k][l], Mip::INTEGER);
                mip.colLowerBound(Targy[i][j][k][l], 0);
                mip.colUpperBound(Targy[i][j][k][l], 1);
            }
        }
    }
}

vector<vector<vector<vector<Mip::Col> > > > Orarendbin;  //Orarendbin[][][][]
Orarendbin.resize(osztalyszam);
for (int i = 0; i < osztalyszam; ++i) {
    Orarendbin[i].resize(orak);
    for (int j = 0; j < orak; ++j) {
        Orarendbin[i][j].resize(napok);
    }
}
for (int i = 0; i < osztalyszam; ++i) {
    for (int j = 0; j < orak; ++j) {
        mip.addColSet(Orarendbin[i][j]);
        for (int k = 0; k < napok; ++k) {
            mip.colType(Orarendbin[i][j][k], Mip::INTEGER);
            mip.colLowerBound(Orarendbin[i][j][k], 0);
            mip.colUpperBound(Orarendbin[i][j][k], 1);
        }
    }
}
vector<vector<vector<vector<Mip::Col> > > > Tanarora;//Tanarora[][][][]
Tanarora.resize(tanarszam);
for (int i = 0; i < tanarszam; ++i) {
    Tanarora[i].resize(targyszam);
    for (int j = 0; j < targyszam; ++j) {
        Tanarora[i][j].resize(orak);
        for (int k = 0; k < orak; ++k) {
            Tanarora[i][j][k].resize(napok);
            }
        }
    }
}

for (int i = 0; i < tanarszam; ++i) {
    for (int j = 0; j < targyszam; ++j) {
        for (int k = 0; k < orak; ++k) {
            mip.addColSet(Tanarora[i][j][k]);
            for (int l = 0; l < napok; ++l) {
                mip.colType(Tanarora[i][j][k][l], Mip::INTEGER);
                mip.colLowerBound(Tanarora[i][j][k][l], 0);
                mip.colUpperBound(Tanarora[i][j][k][l], 1);
            }
        }
    }
}

4.2. Feltételek és célfüggvény
4.2.1. Feltételek

for (int i = 0; i < tanarszam; ++i) {
    for (int j = 0; j < targyszam; ++j) {
        for (int k = 0; k < osztalyszam; ++k) {
            mip.addRow(HRend[i][j][k] >= HRend2[i][j][k][k]);
        }
    }
}

Mip::Expr e3[osztalyszam][targyszam];
for (int i = 0; i < osztalyszam; ++i) {
    for (int j = 0; j < targyszam; ++j) {
        for (int k = 0; k < tanarszam; ++k) {
            e3[i][j][k] = HRend2[k][j][i];
        }
    }
    Mip::Constr b = e3[i][j] == 1;
    mip.addRow(b);
}
Mip::Expr e8[osztalyszam][targyszam];
for (int i = 0; i < osztalyszam; ++i) {
    for (int j = 0; j < targyszam; ++j) {
        for (int k = 0; k < orak; ++k) {
            for (int m = 0; m < napok; ++m) {
                e8[i][j]+=Targy[i][j][k][m];
            }
        }
        Mip::Constr e = e8[i][j]==Kotelezo[i][j][j];
        mip.addRow(e);
    }
}

Mip::Expr e10[osztalyszam][orak][napok];
for (int i = 0; i < osztalyszam; ++i) {
    for (int j = 0; j < orak; ++j) {
        for (int k = 0; k < napok; ++k) {
            for (int m = 0; m < targyszam; ++m) {
                e10[i][j][k]+=Targy[i][m][j][k];
            }
        }
        Mip::Constr g = e10[i][j][k]<=Orarendbin[i][j][k];
        mip.addRow(g);
    }
}

Mip::Expr e13[tanarszam][orak][napok];
for (int i = 0; i < tanarszam; ++i) {
    for (int j = 0; j < orak; ++j) {
        for (int k = 0; k < napok; ++k) {
            for (int m = 0; m < targyszam; ++m) {
                e13[i][j][k]+=Tanarora[i][m][j][k];
            }
        }
        Mip::Constr v = e13[i][j][k]<=Tanarel[i][j][k];
        mip.addRow(v);
    }
}

Mip::Expr e14[tanarszam][osztalyszam][targyszam][orak][napok],
e15[tanarszam][osztalyszam][targyszam][orak][napok];
for (int i = 0; i < tanarszam; ++i) {
    for (int j = 0; j < osztalyszam; ++j) {
        //felt9
    }
}
for (int k = 0; k < targyszam; ++k){
for (int m = 0; m < orak; ++m) {
for (int n = 0; n < napok; ++n){
e14[i][j][k][m][n]=HRend2[i][k][j]+Targy[j][k][m][n];
e15[i][j][k][m][n]=1+Tanarora[i][k][m][n];
Mip::Constr p= e14[i][j][k][m][n]<=e15[i][j][k][m][n];
mip.addRow(p);}}}}

Mip::Expr e16[targyszam][orak][napok],e17[targyszam][orak][napok];
for (int i = 0; i < targyszam; ++i) { //felt14
for (int j = 0; j < orak; ++j) {
for (int k = 0; k < napok; ++k){
for (int l = 0; l < osztalyszam; ++l){
e16[i][j][k]+=Targy[l][i][j][k];}
for (int m = 0; m < tanarszam; ++m){
e17[i][j][k]+=Tanarora[m][i][j][k];}
Mip::Constr u= e16[i][j][k]==e17[i][j][k];
mip.addRow(u);}}}}

4.2.2. Célfüggvény

Mip::Expr cel;
for (int i = 0; i < osztalyszam; ++i) { //célfgyv
for (int j = 0; j < orak; ++j) {
for (int k = 0; k < napok; ++k){
    cel+=Orarendbin[i][j][k]*(j+1)*(j+1);}}}
mip.min();
mip.obj(cel);
mip.solve();
4.3. Kiíratás és kimenet

4.3.1. Kiíratás

```c
for (int i = 0; i < osztalyszam; ++i) {
    printf ("\n");
    printf (" Hétfő | Kedd | Szerda | Csütörtök | Péntek |\n");
    printf ("-_________________________________________");
    for (int k = 0; k < orak; ++k){
        printf ("\n");
        for (int l = 0; l < napok; ++l){
            for (int j = 0; j < targyszam; ++j){
                if (mip.sol(Targy[i][j][k][l])==1){
                    if (j==0){
                        cout << " Magyar ";
                    }
                    else {
                        if (mip.sol(Orarendbin[i][k][l])==0){
                            if (j==0) {
                                cout << "------------";}}}}
            }
        }
    }
}```
**4.3.2. Kimenet**

<table>
<thead>
<tr>
<th>1. osztály</th>
<th>Hető</th>
<th>Kedd</th>
<th>Szerda</th>
<th>Csütörtök</th>
<th>Péntek</th>
</tr>
</thead>
<tbody>
<tr>
<td>Testnevelés</td>
<td>Magyar</td>
<td>Történelem</td>
<td>Idegen nyelv</td>
<td>Magyar</td>
<td>Testnevelés</td>
</tr>
<tr>
<td>Magyar</td>
<td>Történelem</td>
<td>Matematika</td>
<td>Matematika</td>
<td>Magyar</td>
<td>Történelem</td>
</tr>
<tr>
<td>Idegen nyelv</td>
<td>Testnevelés</td>
<td>Magyar</td>
<td>Testnevelés</td>
<td>Magyar</td>
<td>Idegen nyelv</td>
</tr>
<tr>
<td>Informatika</td>
<td>Informatika</td>
<td>Informatika</td>
<td>Informatika</td>
<td>Informatika</td>
<td>Informatika</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2. osztály</th>
<th>Hető</th>
<th>Kedd</th>
<th>Szerda</th>
<th>Csütörtök</th>
<th>Péntek</th>
</tr>
</thead>
<tbody>
<tr>
<td>Magyar</td>
<td>Történelem</td>
<td>Matematika</td>
<td>Matematika</td>
<td>Magyar</td>
<td>Testnevelés</td>
</tr>
<tr>
<td>Testnevelés</td>
<td>Idegen nyelv</td>
<td>Magyar</td>
<td>Testnevelés</td>
<td>Idegen nyelv</td>
<td>Magyar</td>
</tr>
<tr>
<td>Idegen nyelv</td>
<td>Testnevelés</td>
<td>Magyar</td>
<td>Idegen nyelv</td>
<td>Magyar</td>
<td>Idegen nyelv</td>
</tr>
<tr>
<td>Informatika</td>
<td>Informatika</td>
<td>Informatika</td>
<td>Informatika</td>
<td>Informatika</td>
<td>Informatika</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3. osztály</th>
<th>Hető</th>
<th>Kedd</th>
<th>Szerda</th>
<th>Csütörtök</th>
<th>Péntek</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idegen nyelv</td>
<td>Testnevelés</td>
<td>Informatika</td>
<td>Történelem</td>
<td>Testnevelés</td>
<td></td>
</tr>
<tr>
<td>Testnevelés</td>
<td>Idegen nyelv</td>
<td>Magyar</td>
<td>Testnevelés</td>
<td>Idegen nyelv</td>
<td>Magyar</td>
</tr>
<tr>
<td>Idegen nyelv</td>
<td>Testnevelés</td>
<td>Magyar</td>
<td>Informatika</td>
<td>Történelem</td>
<td></td>
</tr>
<tr>
<td>Matematika</td>
<td>Informatika</td>
<td>Történelem</td>
<td>Magyar</td>
<td>Informatika</td>
<td></td>
</tr>
<tr>
<td>Informatika</td>
<td>Informatika</td>
<td>Történelem</td>
<td>Magyar</td>
<td>Informatika</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4. osztály</th>
<th>Hető</th>
<th>Kedd</th>
<th>Szerda</th>
<th>Csütörtök</th>
<th>Péntek</th>
</tr>
</thead>
<tbody>
<tr>
<td>Testnevelés</td>
<td>Idegen nyelv</td>
<td>Történelem</td>
<td>Testnevelés</td>
<td>Idegen nyelv</td>
<td>Magyar</td>
</tr>
<tr>
<td>Idegen nyelv</td>
<td>Történelem</td>
<td>Testnevelés</td>
<td>Magyar</td>
<td>Matematika</td>
<td>Idegen nyelv</td>
</tr>
<tr>
<td>Történelem</td>
<td>Testnevelés</td>
<td>Magyar</td>
<td>Matematika</td>
<td>Idegen nyelv</td>
<td>Magyar</td>
</tr>
<tr>
<td>Idegen nyelv</td>
<td>Testnevelés</td>
<td>Magyar</td>
<td>Matematika</td>
<td>Idegen nyelv</td>
<td>Magyar</td>
</tr>
<tr>
<td>Informatika</td>
<td>Informatika</td>
<td>Magyar</td>
<td>Idegen nyelv</td>
<td>Informatika</td>
<td>Informatika</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>5. osztály</th>
<th>Hető</th>
<th>Kedd</th>
<th>Szerda</th>
<th>Csütörtök</th>
<th>Péntek</th>
</tr>
</thead>
<tbody>
<tr>
<td>Magyar</td>
<td>Testnevelés</td>
<td>Történelem</td>
<td>Matematika</td>
<td>Magyar</td>
<td>Testnevelés</td>
</tr>
<tr>
<td>Testnevelés</td>
<td>Idegen nyelv</td>
<td>Magyar</td>
<td>Testnevelés</td>
<td>Idegen nyelv</td>
<td>Magyar</td>
</tr>
<tr>
<td>Idegen nyelv</td>
<td>Testnevelés</td>
<td>Magyar</td>
<td>Testnevelés</td>
<td>Magyar</td>
<td>Idegen nyelv</td>
</tr>
<tr>
<td>Informatika</td>
<td>Informatika</td>
<td>Informatika</td>
<td>Informatika</td>
<td>Informatika</td>
<td>Informatika</td>
</tr>
</tbody>
</table>
Irodalomjegyzék


[4] Lekin, *Scheduling system developed at the Stern School of Business, NYU*,
    (http://community.stern.nyu.edu/om/software/lekin/)

    (http://lemon.cs.elte.hu/trac/lemon)