FIALA TIBOR

WinQSB

Kezelési segédlet

az

Operációkutatás tantárgyhoz

Lektorálta: Temesi József

BKÁE Operációkutatás Tanszék

Budapest, 2003. június

Tartalomjegyzék

Bevezetés			6
1.	Általános ismeretek		7
2.	Néhány fontosabb modul ismertetése		10
	2.1	Decision Analysis	11
	2.2	Linear and Integer Programming	12
	2.3	Network Modeling	13
	2.4	PERT_CPM	14
	2.5	Queuing Analysis	16
3.	Operációkutatási feladatok megoldása WinQSB-vel		18
	3.1	Lineáris programozás	18
	3.2	Szállítási feladat	20
	3.3	Hozzárendelési feladat	22
	3.4	Egészértékű programozás	24
	3.5	Hálózati modellek	28
		3.5.1. Legrövidebb út	28
		3.5.2. Minimális feszítő fa	29
		3.5.3. Maximális folyam	31
		3.5.4. Kritikus út módszer (CPM)	32
		3.5.5. PERT módszer	34
	3.6	Döntésanalízis	36
		3.6.1. Játékelmélet	36
		3.6.2. Payoff elemzés (Döntési szabályok)	37
		3.6.3. Bayes elemzés	39
		3.6.4. Döntési fa	40
	3.7	Sorbanállási feladat	43
Irodalomjegyzék			46

Bevezetés

A QSB a Quantitative Systems for Business (szabad fordításban: Kvantitatív módszerek a gazdaságban) kifejezés rövidítése. Ennek a programcsomagnak a Windows operációs rendszerre elkészített, 1997-ben véglegesített 1.00 változatával foglalkozunk. Nem a verziószám, sőt nem is az operációs rendszer fontos, hanem az egyes feladatok előkészítése, az algoritmusok futtatása és a végeredmények értelmezése. A kezelési segédlet elsősorban a BKÁE-en Operációkutatás c. tantárgyat tanuló hallgatók számára készült, ezért ennek a tárgynak a tematikája lesz a tárgyalás vezérfonala. Természetesen hasznos lehet a segédlet bárki számára, aki operációkutatást tanul, tanít, vagy a programcsomagban érintett gazdasági számításokkal foglalkozik.

A programcsomag 19 modulból áll, s ezáltal 19 témakört érint. Az első részben a programcsomag kezelésével kapcsolatos általános ismeretekkel foglalkozunk. A második részben 5 olyan modult tárgyalunk, melyek az Operációkutatás c. tantárgy számára fontosak. Mindegyikben ismertetünk egy-egy, a csomaghoz tartozó megoldott mintafeladatot. A 3. részben az Operációkutatás c. tantárgy tematikája szerint haladunk. Megmutatjuk, hogy hogyan lehet új feladatokat a WinQSB program segítségével előkészíteni illetve megoldani, és milyen segítséget nyújt a program a végeredmény értékeléséhez. A feladatokat kevés kivételtől eltekintve Winston [2003] könyvéből választottuk.

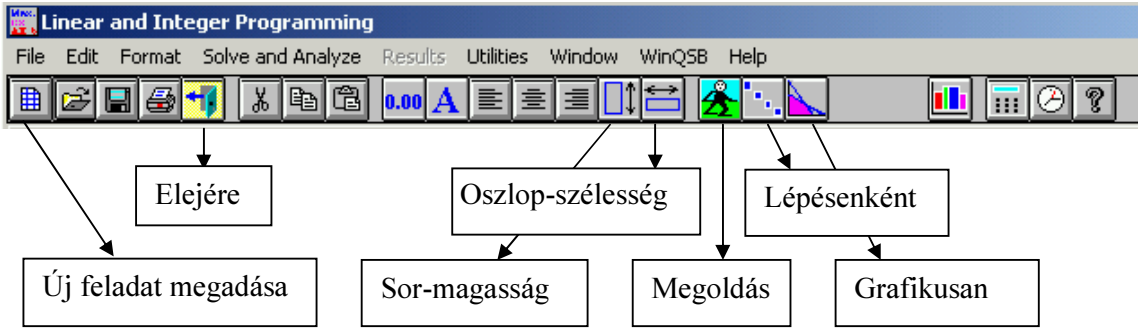
Köszönettel tartozom Temesi Józsefnek a kezdeményezésért, a kézirat alapos áttanulmányozásáért és értékes javaslataiért. Személyes közreműködéssel illetve hasznos megjegyzésekkel hozzájárultak a munkához: Dér Zsuzsanna, Fiala Balázs, Fiala Péter, Fiala Zoltán, Forgó Ferenc, Jeszenői Krisztina, Kádas Sándor, Solymosi Tamás és még sokan mások.

Budapest, 2003. június 15.

1. Általános ismeretek

A programcsomag elindítása után először a leggyakrabban használt modulok listája jelentkezik, néhány másodperc múlva pedig egy 19 soros lista jelenik meg, amelyik a programcsomag valamennyi modulját tartalmazza. Ebből (vagy az előzőből) egy kattintással kiválaszthatjuk azt, amelyikkel dolgozni akarunk. (A 2. részben ismertetjük, hogy az Operációkutatás tantárgy egy adott témaköréhez melyik modul nyújt segítséget.) A különböző modulok gyakorlatilag függetlenek egymástól, mindegyiknek saját menüje, eszközsora, súgója van, s mindegyikhez saját file-kiterjesztés tartozik. Például a *Lineáris programozás* modult (amely tartalmazza az egészértékű programozást is) a **Linear and Integer Programming** paranccsal indíthatjuk, és az alábbi menüvel illetve eszközsorral jelentkezik:

A **Help** menüpont rendkívül sok értékes információt tartalmaz. Például az **About LP-ILP** ágban egy 13 sorból álló lista sorolja, hogy milyen szolgáltatásokat nyújt ez a modul. A baloldali ikonnal új feladat modelljének felírását kezdeményezhetjük. Ezzel a 3. részben foglalkozunk részletesen. A második ikonnal, vagy a *File Load Problem* paranccsal meglévő modellt lehet betölteni, s ezután a következő menüvel illetve eszközsorral találkozunk:



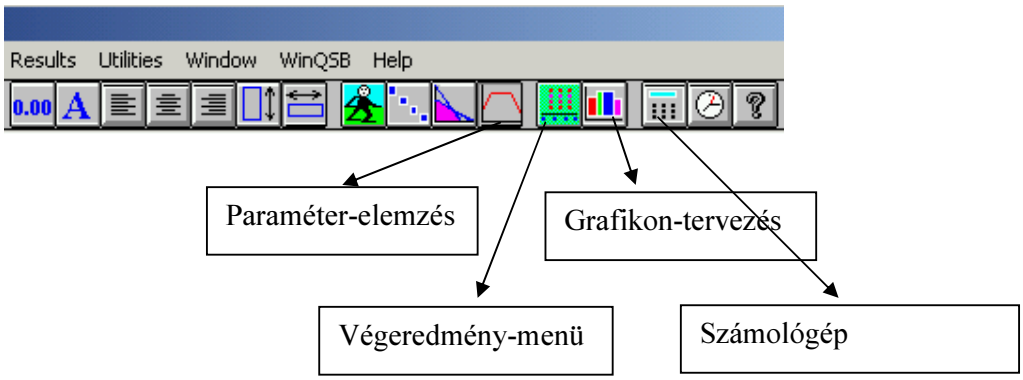
Külön megjelöltük azokat az ikonokat, melyek Word-ben vagy Excelben illetve más Windows alkalmazásokban nem szerepelnek. A balról első ikonnal lehet **új feladat felírását** kezdeményezni, ezt a 3. részben fogjuk használni. Ezután három jól ismert ikon következik: létező modell **betöltése**, **elmentése**, illetve **nyomtatás**. A következő ikonnal a megoldás során menetközben mindig visszamehetünk az adott feladat **elejére**, ahol fel vannak írva az adatok, de még nem kezdődött el a megoldás. Ezután 3+2+3 közismert ikon látható a következő bontásban: **kivágás, másolás, beillesztés**; *szám- és betűformázás*; **balra-, középre-** illetve **jobbraigazítás**. A következő két ikonnal táblázatok (pl. szimplex tábla) **sormagasság**át és **oszlopszélesség**ét lehet állítani. A *síbajnok* ikon azt jelenti, hogy a program *megállás nélkül* végzi el a számításokat, és így jut el a feladat **megoldás**ához. Ezzel szemben a szomszédos **lépésenként** ikonra történő kattintás esetén az algoritmus egyes iterációs lépései után a program megáll, megtekinthetők a részeredmények, és a *Next*

Iteration (vagy valami hasonló) paranccsal megy csak tovább. A következő ikon használata esetén a program **grafikusan** oldja meg a feladatot. Ezután 4 jólismert ikon következik: **grafikon** tervezése valamelyik táblázat kijelölt adataiból (vagy a felhasználó által begépelt adatokból), **zsebszámológép** használata, idő illetve **súgó**.

A súgó (**Help**) minden lényeges információt tartalmaz az angolul olvasni tudó felhasználó számára. Ezen belül rendkívül hasznos a **Glossary**, amelyik ismerteti az összes fontos definíciót, módszert illetve algoritmust. Például, ha nem emlékszik a felhasználó arra, hogy mit jelent az *árnyékár*, vagyis a *shadow price*, akkor a *Glossary Shadow price* pontjában a következő leírást találja: *The shadow price of a constraint is the marginal change of the objective function when the right hand side value of that constraint increases by one unit.* Tehát egy korlátozó feltétel árnyékára nem más, mint a célfüggvény marginális megváltozása, amikor az adott korlátozó feltétel jobboldalát egy egységgel növeljük.

Említettük, hogy minden modulnak saját kiterjesztése van. Például a lineáris programozási feladatokat .LPP kiterjesztéssel tárolja a QSB program, hálózati feladat esetén .NET a kiterjesztés, döntésanalízisben .DAA. Valójában mindegyik ilyen file közönséges text-file, tehát bármilyen text-editorral (pl. Notepad (jegyzettömb), Word, stb.) megnyitható, szerkeszthető.

Az induló menüből a **Solve and Analyze** parancsra hívjuk fel a figyelmet. Ezzel lehet egy menetben (Solve the Problem) vagy lépésenként (Solve and Display Steps) vagy pedig grafikusan (Graphic Method) **megoldani** a feladatot. Amikor eljutottunk a megoldáshoz, általában újabb ikonok is jelentkeznek, például:



Ezek közül a **Végeredmény-menü** ikonnak ugyanaz a hatása, mint a menüben a **Results** pontnak; több oldalról megtekinthetjük a végredményt. Például lineáris programozás esetén a következő lehetőségek vannak:

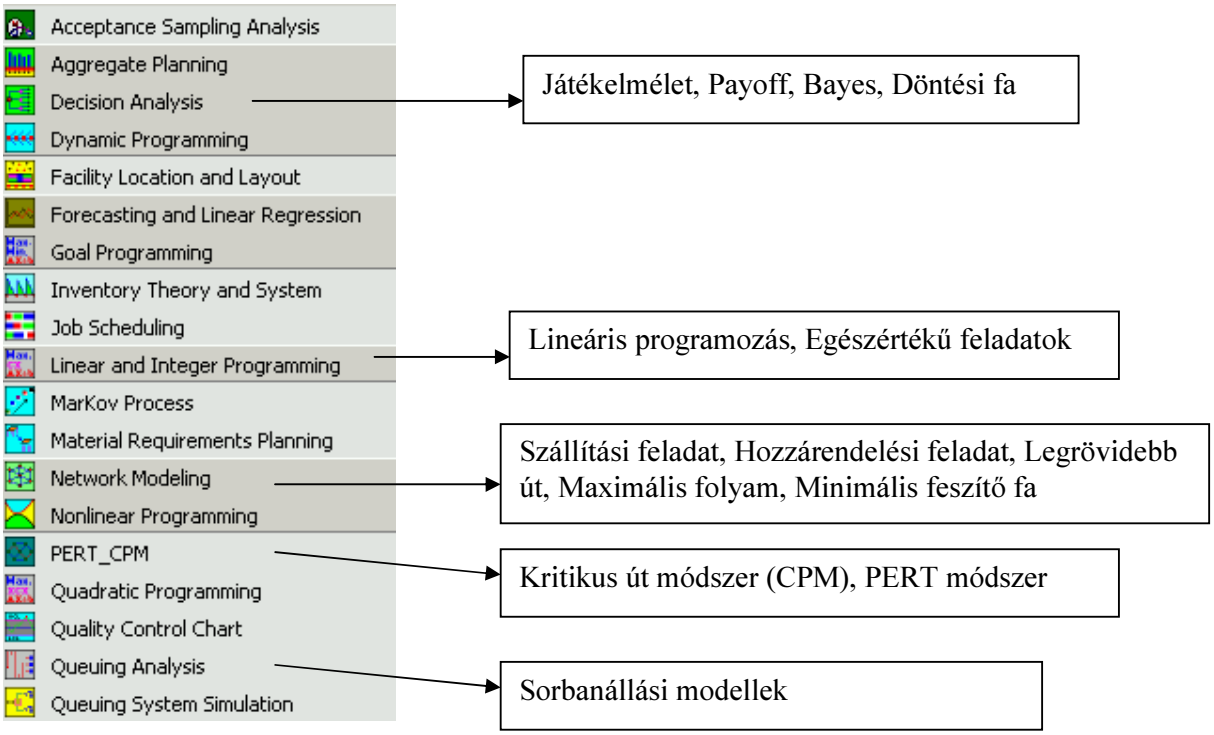
Solution Summary	Az optimális megoldás összegzése
Constraint Summary	A korlátozó feltételek elemzése
Sensitivity Analysis for OBJ	Érzékenységvizsgálat a célfüggvénnyel kapcsolatban
Sensitivity Analysis for RHS	Érzékenységvizsgálat a korlátokkal kapcsolatban
Combined Report	Az előző négy pont egy táblázatban
Perform Parametric Analysis	Paraméter-elemzés kezdeményezése

Final Simplex Table	Végső szimplex-tábla
Show Run Time and Iteration	Futási idő és iterációk száma

Ezek bármelyikét elmenthetjük a Save paranccsal text-file formátumban (a program .txt kiterjesztést javasol). Ha a végeredménnyel kapcsolatos információ grafikus (pl. grafikus módszerrel (Graphic Method) történő megoldás esetén, vagy különböző gráfokkal leírható feladatokban), akkor a grafikus eredményt BMP típusú grafikus formátumban menti el a program.

2. Néhány fontosabb modul ismertetése

Ebben a részben azokat a modulokat ismertetjük, melyek az Operációkutatás tantárgy tematikája szempontjából a legfontosabbak. Mindegyiknél bemutatunk egy a programcsomaghoz tartozó kész modellt. Mindenekelőtt megmutatjuk, hogy az Operációkutatás tantárgy különböző témaköreihez melyik modul nyújt segítséget.



Látható, hogy öt olyan modul van, amelyik a tantárgy szempontjából fontos. Ezért most ismertetni fogjuk az ehhez az öt modulhoz tartozó speciális ismereteket. Mindegyik esetben egy olyan modellen keresztül történik az ismertetés, amely a programcsomaghoz tartozik.

2.1 Decision Analysis

Ennek a modulnak a segítségével *Bayes vizsgálat*ot (azaz különböző valószínűségek kiszámítását) és *Payoff elemzés*t (döntési szabályok vizsgálatát) végezhetünk, ezenkívül *játékelméleti* és *döntési fa* jellegű feladatokat oldhatunk meg. Mindegyik feladatot **.DAA** kiterjesztésű file-ban tároljuk. Példaként töltsük be a BAYESIAN.DAA modellt! Itt egy cég termékei iránti keresletről van szó, amely lehet magas, közepes illetve alacsony.

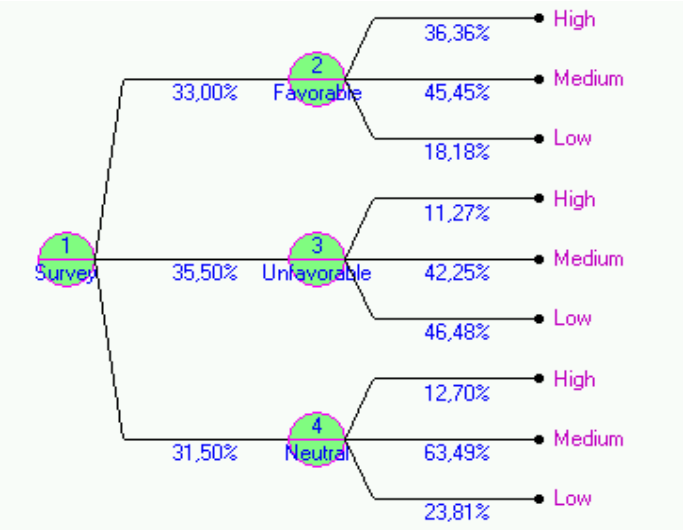
Outcome \ State	High	Medium	Low
Prior Probability	0.20	0.50	0.30
Favorable	0.60	0.30	0.20
Unfavorable	0.20	0.30	0.55
Neutral	0.20	0.40	0.25

Az egyes állapotok a priori valószínűségeit tartalmazza az első sor. Ez alatt a 3x3-as mátrix egy tanácsadó cég előrejelzéseinek feltételes valószínűségeit tartalmazza. Tehát magas kereslet esetén 60% valószínűséggel volt az előrejelzés pozitív, 20% valószínűséggel negatív és 20% valószínűséggel volt semleges. Alacsony kereslet esetén 20% valószínűséggel volt pozitív a jóslat, 55% valószínűséggel volt negatív és 25% valószínűséggel volt semleges. A **Solve and Analyse Solve the Problem** paranccsal számíthatjuk ki az ún. *a-posteriori* (tehát fordított feltételes ld. Help Glossary) valószínűségeket. Arra a kérdésre keresünk választ, hogy pozitív (negatív illetve semleges) előrejelzés esetén mekkora valószínűséggel lesz a kereslet nagy, közepes, illetve kicsi. Az eredményeket a valószínűségszámítás Bayes-tételével számítja ki a program:

Indicator\State	High	Medium	Low
Favorable	0,3636	0,4545	0,1818
Unfavorable	0,1127	0,4225	0,4648
Neutral	0,1270	0,6349	0,2381

Például pozitív előrejelzés esetén 36.36% a magas kereslet valószínűsége, 45.45% a közepesé, 18.18% az alacsonyé. Ugyanakkor például negatív előrejelzése esetén 46.48% valószínűséggel lesz a kereslet alacsony. A modul specialitása egy új ikon.

Döntési fa gráfjának felrajzolása. (Ha ezt választjuk, akkor először egy technikai jellegű ablak jelenik meg, ahol a fa méreteit és feliratait állíthatjuk be. OK parancsra készül el az ábra.) Ezzel ugyanazt a hatást érjük el, mintha a **Results** menüből a **Show Decision Tree Graph** parancsot hívnánk.



Itt 33.00% az Outcome/State táblázat első és második sorának skaláris szorzata, 35.5% az első és harmadik sor, 31.5% pedig az első és negyedik sor skaláris szorzata. Tehát arról van szó, hogy *általában* milyen valószínűséggel pozitív, negatív illetve semleges a cég előrejelzése. A további 9 valószínűség (melyek a döntési fa leveleihez tartoznak) az *Indicator /State* táblázat ismeretében nem szorul magyarázatra.

2.2 Linear and Integer Programming

Ennek a modulnak a segítségével *folytonos, egészértékű illetve vegyes lineáris* programozási feladatokat lehet megoldani. Ezeket a feladatokat **.LPP kiterjesztésű** file-okban tároljuk. Kétváltozós feladat esetén választhatunk a *grafikus* illetve az *algebrai* módszer között. Ha minden változó folytonos, akkor a modul a *szimplex módszer*t használja, egészértékű és vegyes feladatok esetén pedig a *Branch-And-Bound módszer*t. A modul specialitása a **Format** menüpontban található **Switch to Dual Form** parancs, melynek segítségével azonnal megkapjuk az LP feladat **duálisát**.

Példaként töltsük be a programcsomaghoz tartozó LP.LPP modellt!

Variable -->	X1	X2	Direction	R. H. S.
Maximize	50	60		
C1	2	3	<=	180
C2	3	2	<=	150
LowerBound	0	0		
UpperBound	M	M		
VariableType	Continuous	Continuous		

Ez az
$(50x_1+60x_2) \rightarrow max$

$(2x_1+3x_2) <= 180$
$(3x_1+2x_2) <= 150$
$0<=x_1, x_2$
feladat megfogalmazása QSB-vel.

Ha a **Solve and Analyze** menüpontból a **Solve the Problem** parancsot választjuk, akkor a program meghatározza az optimális megoldást. A **Results** menüpontban a **Solution Summary** összegzi a végeredményt:

Decision Variable	Solution Value	Unit Cost or Profit C(j)	Total Contribution	Reduced Cost	Basis Status
X1	18,0000	50,0000	900,0000	0	basic
X2	48,0000	60,0000	2 880,0000	0	basic
Objective	Function	(Max.) =	3 780,0000		

Az optimális megoldás:
$x_1=18$, $x_2=48$,

a célfüggvény maximuma 3780.

Az is látható, hogy mindkét változó bázisváltozó.

Egyenként megtekinthetjük a szimplex módszer lépéseit, ha visszatérünk a feladat elejére (az Elejére ikonnal, ld. 1. rész), és a **Solve and Analyze** menüpontból a **Solve and Display Steps** parancsot választjuk (vagy a *Lépésenként* ikonra kattintunk). Az indulótáblában két maradék-változó van a bázisban, ezek C1, illetve C2. A célfüggvényben mindketten 0 együtthatóval szerepelnek. A szükkeresztmetszet szabály által meghatározott generáló-elem (pivot-elem) inverz módban van feltüntetve (1. sor, 2. cella). Tehát x_2 bekerül a bázisba, c1 pedig kilép a bázisból. Az eredmény a **Simplex Iteration Next Iteration** parancs után látható.

		X1	X2	Slack_C1	Slack_C2		
Basis	C(j)	50,0000	60,0000	0	0	R. H. S.	Ratio
Slack_C1	0	2,0000	3,0000	1,0000	0	180,0000	60,0000
Slack_C2	0	3,0000	2,0000	0	1,0000	150,0000	75,0000
	C(j)-Z(j)	50,0000	60,0000	0	0	0	

		X1	X2	Slack_C1	Slack_C2		
Basis	C(j)	50,0000	60,0000	0	0	R. H. S.	Ratio
X2	60,0000	0,6667	1,0000	0,3333	0	60,0000	90,0000
Slack_C2	0	1,6667	0	-0,6667	1,0000	30,0000	18,0000
	C(j)-Z(j)	10,0000	0	-20,0000	0	3 600,0000	

A célfüggvény értéke most 3600, s az új generáló elem a 2. sor 1. cellája.

A további lépések ismertetésétől eltekintünk. Megemlítjük még, hogy ha visszatérünk a feladat elejére, és a **Solve and Analyze** menüpontból a **Graphic Method** parancsot választjuk., akkor a program grafikusan oldja meg a feladatot, és miután megadtuk, hogy x_2 és x_2 közül melyik legyen a vízszintes, melyik a függőleges tengelyen, a program ábrázolja a grafikus megoldást.

2.3 Network Modeling

Ennek a modulnak a segítségével oldhatjuk meg a *szállítási feladatot*, a *hozzárendelési feladatot*, valamint számos *hálózati modell* problémát, mint például *legrövidebb út*, *maximális folyam* és *minimális feszítő fa* feladatok. Ezeket a feladatokat **.NET kiterjesztésű** file-okban tároljuk. A modul specialitása a **Format** menüpontban található **Switch to Graphic model** parancs, és ennek párja a **Switch to Matrix Form** parancs, melyek segítségével váltani lehet a **gráf**- illetve **mátrix**-reprezentáció között. A **Solve and Analyse** menüpontban általában választhatunk az **induló-megoldás** előállítására alkalmazott módszerek között (*Select Initial Solution Method*). Az egyes módszereket a Help Glossary ismerteti. **Lépésenkénti megoldás** esetén (*Solve and Display Steps*) eldönthetjük, hogy **gráf** formájában (*Network*) **vagy mátrix**-alakban (*Tableau*) akarjuk látni az algoritmus lépéseit. - Példaként töltsük be a programcsomaghoz tartozó ASSIMENT.NET modellt! (A G és N betű azért hiányzik, mert évekkel ezelőtt egy file neve nem lehetett 8 karakternél hosszabb.) Ez egy hozzárendelési feladat 4 munkással és 4 munkával.

From \ To	A	B	C	D
John	3	6	7	10
Peter	5	6	3	8
Toshi	2	8	4	16
Rudy	8	6	5	9

A táblázat azt mutatja, hogy melyik munkás melyik munkát menynyiért hajlandó elvégezni. Mindegyik munkát el kell végezni úgy, hogy egy munkás csak egy munkát végezhet, és az összköltséget kell minimalizálni.

Ha a **Solve and Analyze** menüpontból a **Solve the Problem** parancsot választjuk (vagy a síbajnok ikonra kattintunk), akkor a program meghatározza az optimális megoldást. Itt

From	To	Assignment	Unit Cost	Total Cost
John	B	1	6	6
Peter	C	1	3	3
Toshi	A	1	2	2
Rudy	D	1	9	9
Total	Objective	Function	Value =	20

látható, hogy melyik munkás melyik munkát kapta (pl. John a B-t, Peter a C-t, stb.), és a minimális összköltség értéke 20.

Ugyanakkor az eszközsorban egy új ikon jelentkezik:

 Grafikus megoldás

Ha erre kattintunk, ugyanazt az eredményt érjük el, mintha a **Results** menüből a **Graphic Solution** parancsot választanánk; gráf formájában látjuk a hozzárendelést. Visszatérve a

feladat elejére (használjuk az *Elejére* ikont) a **Solve and Analyze** menüpontból válasszuk a **Solve and Display Steps-Tableau** parancsot! Ezzel lépésenként oldjuk meg a feladatot, mégpedig mátrix alakban.

	A	B	C	D
John	0	2	4	3
Peter	2	2	0	1
Toshi	0	5	2	10
Rudy	3	0	0	0

A magyar módszer első iterációs lépését látjuk (redukált táblázatban). A 6 db nullát három vonal lefedi, tehát legfeljebb három független 0 található. Az algoritmus addig folytatódik, míg a független nullák száma el nem éri a 4-et.

Az *Iteration Next Iteration* paranccsal ez már a második lépésnél bekövetkezik.

	A	B	C	D
John	0	0	2	1
Peter	4	2	0	1
Toshi	0	3	0	8
Rudy	5	0	0	0

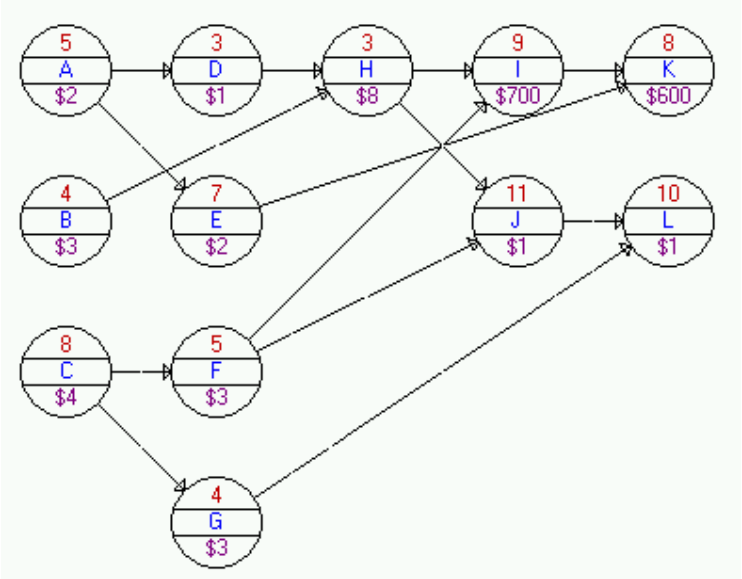
John - B
Peter - C
Toshi - A
Rudy - D
az optimális megoldás.

2.4 PERT_CPM

Ennek a modulnak a segítségével az ún. PERT (Problem Evaluation and Review Technique) módszert illetve a kritkus út módszert (CPM=Critical Path Method) alkalmazhatjuk többek között projekt-ütemezési feladatok megoldására. A feladatokat **.CPM kiterjesztésű** file-okban tároljuk. A modul specialitása a **Format** menüpontban található **Switch to Graphic model** parancs, és ennek párja a **Switch to Matrix Form** parancs, melyek segítségével váltani lehet a **gráf**- illetve **mátrix**-reprezentáció között. A **Solve and Analyse** menüpontban választhatunk, hogy normál időkkel (Normal Time) vagy erőltetett menetben érvényes munkaidőkkel (Crash Time) számolunk. Példaként töltsük be a programcsomaghoz tartozó CPM.CPM modellt! Ez egy projekt-ütemezési feladat 12 tevékenységgel. A táblázat azt mutatja, hogy melyik tevékenységet mely tevékenységek előzik meg közvetlenül (**Immediate Predecessor**), és melyik tevékenységhez mekkora munkaidő illetve mekkora költség tartozik.

Activity Name	Immediate Predecessor (list number/name, separated by ',')	Normal Time	Crash Time	Normal Cost	Crash Cost
A		5	3	$2 000	$2 500
B		4	4	$3 000	$3 000
C		8	7	$4 000	$5 000
D	A	3	2	$1 200	$1 500
E	A	7	5	$2 000	$3 000
F	C	5	5	$3 000	$3 000
G	C	4	3	$3 000	$3 700
H	B,D	3	3	$8 000	$8 000
I	F,H	9	6	$700	$1 600
J	F,H	11	7	$1 500	$2 000
K	E,I	8	6	$600	$1 500
L	G,J	10	9	$1 000	$1 050

A Format Switch to Graphic Model paranccsal megtekint-hetjük a feladat gráfját.



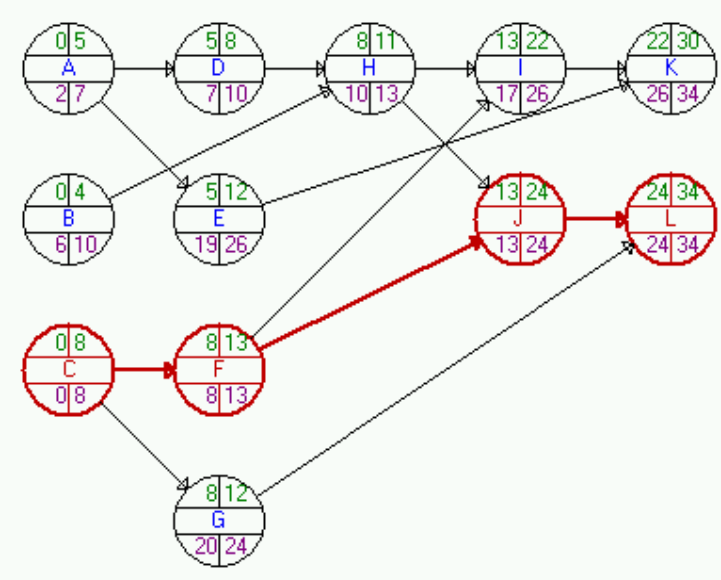
Ezen jól felismerhetők a precedencia előírások, a tevékenységek betűjelei, a munkaidők és a költségek.

Ha a **Solve and Analyze** menüpontból a **Solve Critical Path using Normal Time** parancsot választjuk, akkor a program meghatározza az optimális megoldást. A táblázatban a **Yes** szó jelzi a kritikus út éleit (C-F-J-L).

Ugyanakkor új ikonok is jelentkeznek:

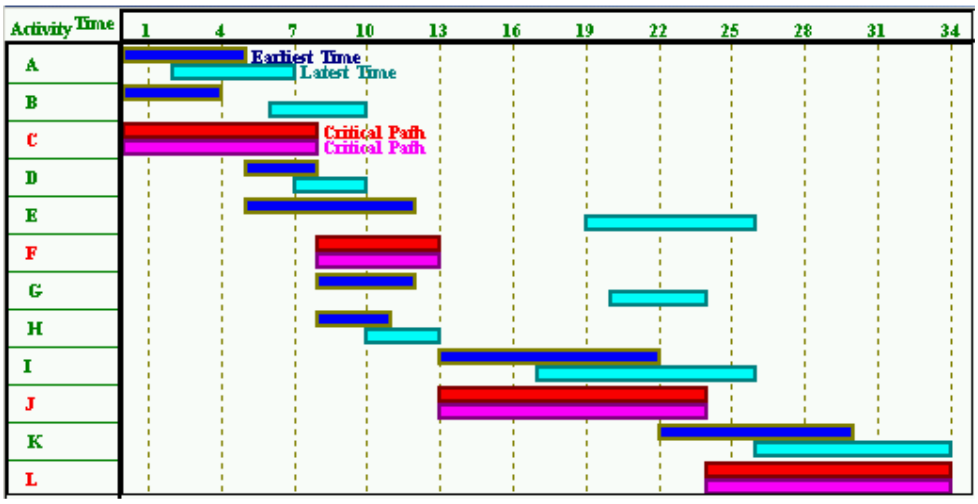
 Graphic Activity Analysis és  Gantt diagram

Az elsőre kattintva ugyanazt az eredményt érjük el, mintha a **Results** menüből a **Graphic Activity Analysis** parancsot választanánk.



Egy kör felső részén a legkorábbi elkezdés illetve befejezés időpontját, az alsó részen pedig a legkésőbbi időpontokat látjuk. A kritikus út élein a legkorábbi és a legkésőbbi adatok azonosak.
A projekt megvalósításához minimálisan 34 időegységre van szükség.

Ha a *Gantt diagram* ikonra kattintunk, vagy a **Results** menüből a **Gantt Chart** parancsot választjuk, akkor ugyanennek az eredménynek az ún. Gantt diagramját kapjuk meg, melynek értelmezése egyszerű feladat.



Sötétkék szín jelzi a legkorábbi, világoskék pedig a legkésőbbi végrehajtás időontervallumát. A kritikus út tevékenységei esetén a kettő azonos.

2.5 Queuing Analysis

Ennek a modulnak a segítségével sorbanállási feladatokat lehet megoldani.A feladatokat **.QAA kiterjesztésű** file-okban tároljuk. A modul specialitása a **Format** menüpontban található **Switch to General Queuing Format** parancs, és ennek párja a **Switch to Simple M/M Format** parancs, melyek segítségével váltani lehet az általános sorbanállási feladatok illetve a speciális **M/M/s** típusú sorbanállási feladatok között. Az utóbbiak megoldására explicit képleteket használ a program (a részletek a **Help Glossary Queuing System Classification** pontjában találhatók), az általános típusra viszont közelítő módszereket, vagy szimulációt.
Példaként töltsük be a programcsomaghoz tartozó QUEUE1.QAA modellt! Ez egy M/M/1 típusú sorbanállási feladat, tehát 1 szerver van, amelyik egy óra alatt átlagosan 3 ügyfelet szolgál ki, (μ=3) és óránként átlagosan 2 ügyfél érkezik (λ=2). A Queue Capacity mezőbe gépeljünk be 6-ot! Ez azt jelenti, hogy egyszerre legfeljebb hatan állhatnak sorban, ezenkívül 1 fő lehet kiszolgálás alatt, tehát a rendszer hossza 7. A **Solve and Analyze Solve the Performance** paranccsal megkapjuk a rendszer legfontosabb átlagos jellemzőit.

Data Description	ENTRY
Number of servers	1
Service rate (per server per hour)	3
Customer arrival rate (per hour)	2
Queue capacity (maximum waiting space)	6
Customer population	

Performance Measure	Result
System: M/M/1/7	From Formula
Customer arrival rate (lambda) per hour =	2,0000
Service rate per server (mu) per hour =	3,0000
Overall system effective arrival rate per hour =	1,9594
Overall system effective service rate per hour =	1,9594
Overall system utilization =	65,3132 %
Average number of customers in the system (L) =	1,6752
Average number of customers in the queue (Lq) =	1,0220
Average number of customers in the queue for a busy system (Lb) =	1,5648
Average time customer spends in the system (W) =	0,8549 hours
Average time customer spends in the queue (Wq) =	0,5216 hours
Average time customer spends in the queue for a busy system (Wb) =	0,7986 hours
The probability that all servers are idle (Po) =	34,6868 %
The probability an arriving customer waits (Pw) or system is busy (Pb) =	65,3132 %
Average number of customers being balked per hour =	0,0406

Megfigyelhető, hogy a rendszer M/M/1/7 típusú, ahol 7 a rendszer hossza. Egy ügyfél átlagosan 0.5216 órát tölt a sorban, további 0.3333 órát a pultnál, tehát átlagosan 0.8549 órát tölt a rendszerben. Annak a valószínűsége, hogy az ügyfél még a sorba sem tud beállni (hiszen a sor megtelt, hatan állnak ott), 0.0406. A költségekre vonatkozó sorokkal most nem foglalkozunk

03 13:09:41 n	Estimated Probability of n Customers in the System
0	0,3469
1	0,2312
2	0,1542
3	0,1028
4	0,0685
5	0,0457
6	0,0305
7	0,0203

Figyelemreméltó a **Results Probability Summary** menüpont, ahol minden `n`-re megtekinthetjük annak valószínűségét, hogy éppen `n` ügyfél van a rendszerben, azaz `n-1` ügyfél áll a sorban.Például annak a valószínűsége, hogy senki sem áll sorban, 0.3469+0.2312=0.5781.

3. Operációkutatási feladatok megoldása WinQSB-vel

3.1 Lineáris programozás

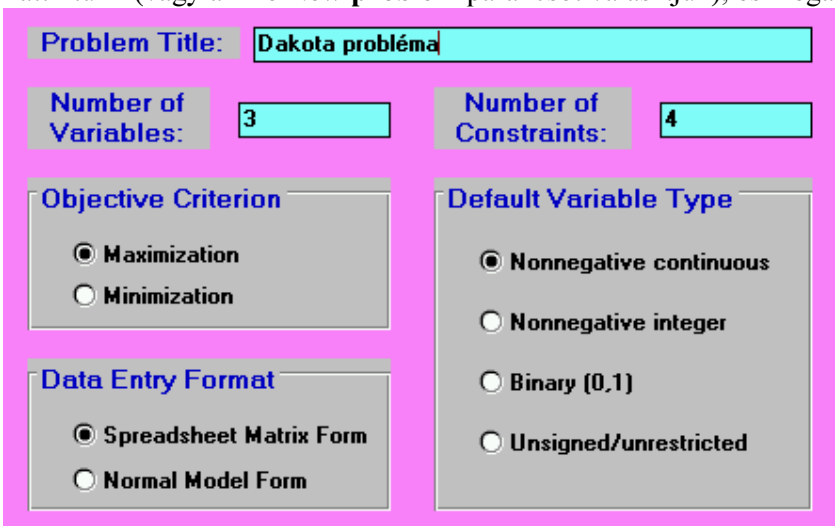
3.1.1 Első példaként tekintsük az alábbi normál-feladatot!

$(60x_1+30x_2+20x_3) = z \rightarrow max$

$$\begin{aligned} 8x_1+\ 6x_2+\ x_3 &<= 48 \\ 4x_1+\ 2x_2+1.5x_3 &<= 20 \\ 2x_1+1.5x_2+0.5x_3 &<= 8 \\ x_2 &<= 5 \end{aligned}$$

$0<=x_1,\ x_2,\ x_3$

A **Linear and Integer Programming** modul elindítása után az *Új feladat megadása* ikonra kattintunk (vagy a **File New problem** parancsot választjuk), és megadjuk az alapadatokat:



A változók száma 3, 4 korlátozó feltétel van, maximalizálni akarunk, és nemnegatív, folytonos változóink vannak.

A számadatokat táblázatos formátumban (spreadsheet Matrix Form) érdemes bevinni.

Az OK gombra kattintva begépelhetjük az adatokat (a program tizedespontot használ):

Variable -->	X1	X2	X3	Direction	R. H. S.
Maximize	60	30	20		
C1	8	6	1	<=	48
C2	4	2	1.5	<=	20
C3	2	1.5	0.5	<=	8
C4	0	1	0	<=	5
LowerBound	0	0	0		
UpperBound	M	M	M		
VariableType	Continuous	Continuous	Continuous		

Mindegyik változó nemnegatív, tehát alsó korlátja 0, felső korlátja pedig $+\infty$. A korlátozó feltételek illetve a változók típusát szükség esetén dupla kattintással módosíthatjuk.

A **síbajnok ikon**ra kattintva (vagy a *Solve and Analyze, Solve the Problem* paranccsal) azonnal megkapjuk az optimális megoldást. A **Results** menüpontból érdemes a **Solution Summary** pontot választani:

Decision Variable	Solution Value	Unit Cost or Profit C(j)	Total Contribution	Reduced Cost	Basis Status
X1	2,0000	60,0000	120,0000	0	basic
X2	0	30,0000	0	-5,0000	at bound
X3	8,0000	20,0000	160,0000	0	basic
Objective	Function	(Max.) =	280,0000		

A célfüggvény maximuma 280, $x_1=2$ és $x_3=8$ az optimális bázisváltozók értéke, $x_2=0$ nincs a bázisban.

A Results menü többi elemét az 1. részben érintettük. Megemlítjük még, hogy lépésenként is megoldhattuk volna a feladatot, hogyha a síbajnok ikon helyett a *Lépésenként* ikont (vagy a *Solve and Analyze Display Steps* parancsot és az *Iteration* menüpontot) választottuk volna.Visszatérve a feladat elejére (az *Elejére* ikonnal) a **Format Switch to Dual Form** paranccsal megkapjuk a duális feladatot, amit ugyanúgy oldhatunk meg, mint az eredeti primál feladatot. $y_1=0$, $y_2=10$, $y_3=10$ és $y_4=0$ a duál-optimális megoldás, a célfüggvény optimuma **(ez most minimum)** természetesen most is 280.

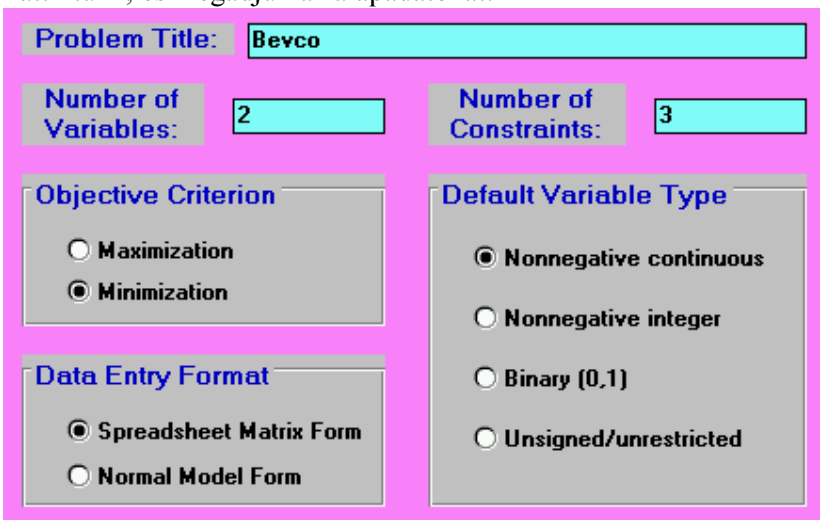
3.1.2 Tekintsük most az alábbi LP feladatot, amelyik >= és = típusú korlátozó feltételt is tartalmaz:

$$(2x_1+3x_2) = z \rightarrow \min$$

$$0.5x_1+0.25x_2 <= 4$$
$$x_1+3x_2 >= 20$$
$$x_1+x_2 = 10$$

$$0<=x_1,\ x_2$$

A **Linear and Integer Programming** modul elindítása után az *Új feladat megadása* ikonra kattintunk, és megadjuk az alapadatokat:



Két folytonos, nemnegatív változónk van 3 korlátozó feltétellel.
A célfüggvényt minimalizálni kell.

OK

A korlátozó feltételek típusát (és szükség esetén a változók típusát is), a számadatok begépelése közben dupla kattintással állíthatjuk.

Variable -->	X1	X2	Direction	R. H. S.
Minimize	2	3		
C1	0.5	0.25	<=	4
C2	1	3	>=	20
C3	1	1	=	10
LowerBound	0	0		
UpperBound	M	M		
VariableType	Continuous	Continuous		

A Direction oszlopban dupla kattintással lehet kiválasztani az egyenlőtlenség típusát, illetve az egyenlőséget.

Megemlítjük, hogy az **Edit, Variables Names** paranccsal megadhatjuk a változók nevét, az **Edit, Constraint Names** paranccsal a korlátozó feltételek neveit, az **Edit, Objection Function Criterion** paranccsal választhatunk a minimalizálás illetve maximalizálás között, új változót illetve korlátot lehet beilleszteni az **Edit, Insert a Variable** illetve **Edit, Insert a Constraint** parancsokkal..
A megoldást ugyanúgy kapjuk meg, mint a 3.1.1 Példa esetén (*síbajnok ikon, Results, Solution Summary*)

Decision Variable	Solution Value	Unit Cost or Profit C(j)	Total Contribution	Reduced Cost	Basis Status
X1	5,0000	2,0000	10,0000	0	basic
X2	5,0000	3,0000	15,0000	0	basic
Objective	Function	(Min.) =	25,0000		

A célfüggvény minimuma 25, $x_1=5$ és $x_3=5$ mindketten bázisváltozók.

3.1.1 mintájára ezt a feladatot is megoldhattuk volna lépésenként. Visszatérve a feladat elejére (az *Elejére* ikonnal) a **Format Switch to Dual Form** paranccsal megkapjuk a duális feladatot, ahol y_1 nempozitív, y_2 nemnegatív, y_3 pedig kötetlen előjelű. A duál feladatot ugyanúgy oldhatjuk meg, mint az eredeti primál feladatot. $y_1=0$, $y_2=0.5$, $y_3=1.5$ a duál-optimális megoldás, a célfüggvény optimuma **(ez most maximum)** természetesen most is 25.

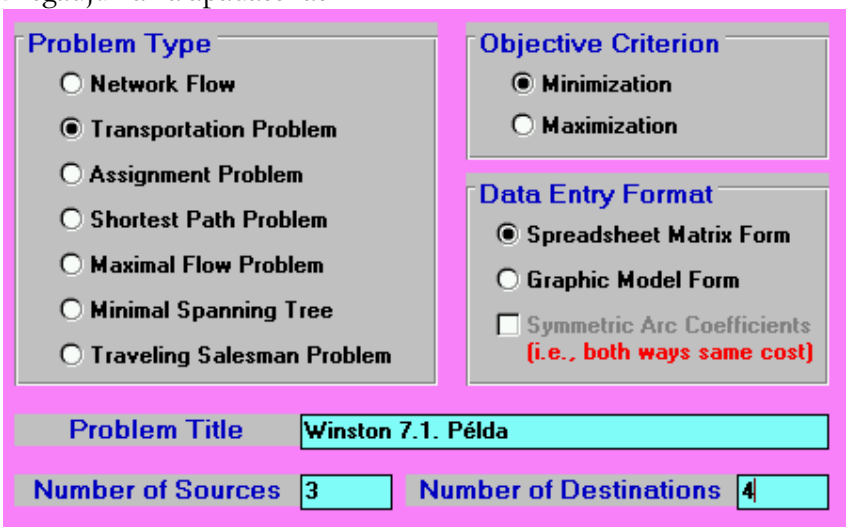
3.2 Szállítási feladat

Tekintsük a következő szállítási feladatot!

	V1	V2	V3	V4	Kapacitás
E1	8	6	10	9	**35**
E2	9	12	13	7	**50**
E3	14	9	16	5	**40**
Igény	**45**	**20**	**30**	**30**	

Itt E1, E2 és E3 három erőművet jelöl, V1, V2, V3 és V4 pedig városok. Az egyes erőművek kapacitásai, és a városok energia-igényei adottak. Figyelemreméltó, hogy a kapacitások összege és az igények összege azonos. (A program megengedi, hogy a kapacitások összege nagyobb, vagy kisebb legyen az igények összegénél. Az első esetben megadja a *felhasználatlan kapacitás* (Unused_Suppply), a második esetben pedig a *kielégítetlen*

kereslet (Unfilled_Demand) mértékét.) Ezenkívül adottak az egységnyi szállítási költségek. Például egységnyi energia szállításának költsége a 3-as erőműből a 2-es városba 9, a 2-es erőműből a 3-as városba viszont 13. Olyan szállítási tervet kell készíteni, amelyik megmondja, hogy melyik erőműből melyik városba mennyit szállítsunk úgy, hogy mindegyik város igénye pontosan ki legyen elégítve, egyik erőmű kapacitását se lépjük túl (tehát mindegyiket teljes mértékben használjuk ki), és az összköltség minimális legyen.
A **Network Modeling** modul elindítása után az *Új feladat megadása* ikonra kattintunk, és megadjuk az alapadatokat:



Szállítási feladatot oldunk, 3 forrásunk (erőmű) és 4 rendeltetési helyünk van, a célfüggvényt pedig minimalizálni akarjuk. Az adatokat mátrix formában célszerű bevinni.

OK.

Ezután begépeljük a kapacitásokat, az igényeket, és az egységnyi szállítási költségeket.

From \	V1	V2	V3	V4	Supply
E1	8	6	10	9	35
E2	9	12	13	7	50
E3	14	9	16	5	40
Demand	45	20	30	30	

A források és rendeltetési helyek neveit az **Edit, Node Names** paranccsal lehet megváltoztatni. Az *oszlop-szélesség ikonnal* beállíthatjuk a kívánatos méretet.

A feladatot lépésenként is meg lehet oldani, válasszuk azonban most az azonnali megoldást (a *síbajnok ikon*ra kattintunk, *Results, Solution Table – Nonzero Only*).

From	To	Shipment	Unit Cost	Total
E1	V2	10	6	60
E1	V3	25	10	250
E2	V1	45	9	405
E2	V3	5	13	65
E3	V2	10	9	90
E3	V4	30	5	150
Total	Objective	Function	Value =	1020

A 3x4 szállítási lehetőségből 6=3+4-1 van kihasználva. Az első erőmű 10 egységet szállít a 2-es városba, és 25 egységet a 3-asba. A 2-es erőmű 45 egységet szállít az 1-es városba, és 5 egységet a 3-asba, és így tovább. A minimális szállítási költség értéke 1020.

Megemlítjük még, hogy ha az adatok begépelése után (a *Lépésenként* ikonnal, vagy a *Solve and Analyze, Solve and Display Steps – Tableau* paranccsal) a lépésenkénti megoldást választjuk, akkor végignézhetjük a huroktranszformációs módszer egyes állomásait. A **Solve and Analyze, Select Initial Solution Method** paranccsal megválaszthatjuk az induló lehetséges megoldás előállítására használt módszert. Ha például ezek közül nem ismeri az olvasó a Russel féle módszert (*Russel's Approximation Method*), akkor a Help Glossary ad eligazítást.

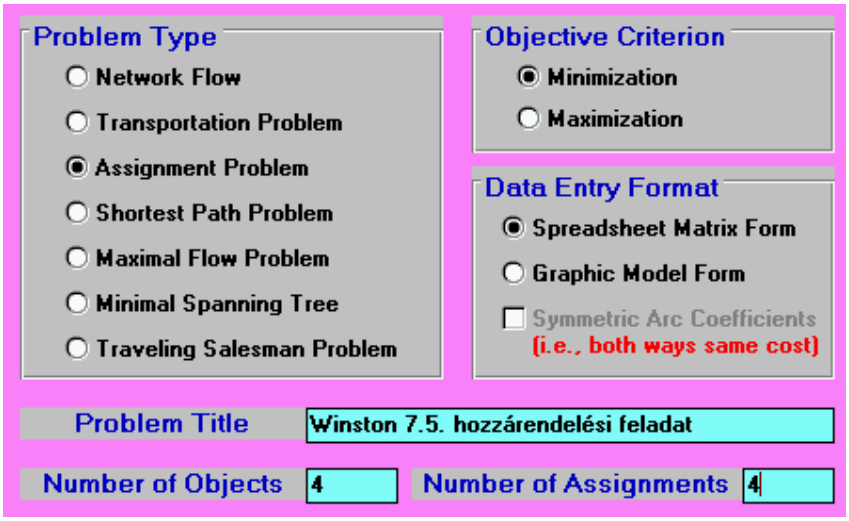
3.3 Hozzárendelési feladat

Tekintsük az alábbi hozzárendelési feladatot!

	M1	**M2**	**M3**	**M4**
G1	14	5	8	7
G2	2	12	6	5
G3	7	8	3	9
G4	2	4	6	10

Itt G1, G2, G3 és G4 négy gépet jelöl, M1, M2, M3 és M4 pedig négy munkát. A táblázat azt mutatja, hogy melyik munkát melyik géppel mennyi idő alatt lehet elvégezni. Például a 3-as munkát az 1-es géppel 8, a 4-es géppel viszont 6 időegység alatt lehet elvégezni. Egy géphez csak egy munkát rendelhetünk, ugyanakkor mindegyik munkát el kell végezni. Az a kérdés, hogy melyik munkát melyik géphez rendeljük úgy, hogy a szükséges munkaidők összege minimális legyen.

A **Network Modeling** modul elindítása után az *Új feladat megadása* ikonra kattintunk, és megadjuk az alapadatokat:



Hozzárendelési feladatot oldunk meg, 4 gép és 4 munka adott, a célfüggvényt pedig minimalizálni akarjuk. Az adatokat mátrix formában célszerű bevinni.

OK.

Ezután begépeljük az egyes munkák elvégzéséhez szükséges idő-adatokat.

From \ To	M1	M2	M3	M4
G1	14	5	8	7
G2	2	12	6	5
G3	7	8	3	9
G4	2	4	6	10

A gépek és munkák neveit az **Edit, Node Names** paranccsal lehet megváltoztatni. Az *oszlop-szélesség ikonnal* beállíthatjuk a kívánatos méretet.

Válasszuk most a a *Lépésenként* ikonnal, vagy a *Solve and Analyze, Solve and Display Steps - Tableau* paranccsal a lépésenkénti megoldást! A program az ún. magyar módszert alkalmazza.

	M1	M2	M3	M4
G1	9	0	3	0
G2	0	10	4	1
G3	4	5	0	4
G4	0	2	4	6

Minden sorból levontuk a legkisebb elemet, ezután azokból az oszlopokból, ahol még nem volt nulla, szintén levontuk a legkisebb elemet. Így 5 db nullát kaptunk, és ezeket három vonallal le tudjuk fedni. A le nem fedett elemek minimuma 1. Ezt kivonjuk a le nem fedett elemekből, és hozzáadjuk a duplán lefedett elemekhez. (Valójában kivonunk 1-et a 2. és 4. sorból, és hozzáadunk 1-et az első oszlophoz.) A következő lépést az *Iteration, Next Iteration* paranccsal kapjuk meg.

	M1	M2	M3	M4
G1	10	0	3	0
G2	0	9	3	0
G3	5	5	0	4
G4	0	1	3	5

Most már csak négy vonallal tudjuk lefedni a nullákat, tehát van 4 db független 0, melyek megadják az optimális hozzárendelést:

```
G1:    M2
G2:    M1
G3:    M3
G4:    M1
```

A célfüggvény minimuma 5+5+3+2 = 15.
A *Results, Graphic Solution* paranccsal *grafikusan* is megtekinthetjük az eredményt.

3.4 Egészértékű programozás

Ennek a szakasznak a tanulmányozása előtt érdemes megismerkedni a 3.1 szakasszal (Lineáris programozás), hiszen itt is lineáris feladatokat oldunk meg. A lényeges újdonság az, hogy a változók egy része (vagy valamennyi) egészértékű kell, hogy legyen.

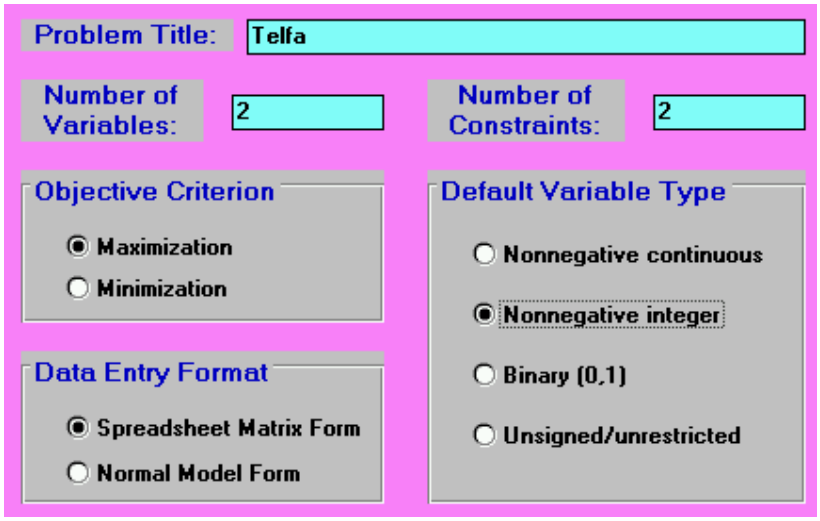
3.4.1 Első példaként tekintsük az alábbi tiszta egészértékű feladatot!

$(8x_1+5x_2) = z \rightarrow \max$

$x_1 + x_2 <= 6$
$9x_1 + 5x_2 <= 45$

$0<=x_1,\ x_2$
$x_1,\ x_2$ egész

A **Linear and Integer Programming** modul elindítása után az *Új feladat megadása* ikonra kattintunk, és megadjuk az alapadatokat:



2 változónk van 2 feltétellel, a változók típusa nemnegatív egész. A célfüggvényt maximalizálni kell.OK
Begépeljük az együtthatókat:

Variable -->	X1	X2	Direction	R. H. S.
Maximize	8	5		
C1	1	1	<=	6
C2	9	5	<=	45
LowerBound	0	0		
UpperBound	M	M		
VariableType	Integer	Integer		

Szükség esetén dupla kattintással változtathatnánk a változók illetve a korlátozó feltételek típusát.

Természetesen azonnal megkaphatjuk a végeredményt a síbajnok ikon segítségével. Ehelyett azonban válasszuk most a lépésenkénti megoldást, hogy tanulmányozhassuk a korlátozás és szétválasztás (vagyis Branch-and-Bound) módszer részeredményeit!

A **Lépésenként** ikonra kattintva (vagy a *Solve and Analyze, Solve and Display Steps* paranccsal) megkapjuk az **első** részeredményt, vagyis a **folytonos** feladat optimális megoldását:

Decision Variable	Lower Bound	Upper Bound	Solution Value	Variable Type	Status
X1	0	M	3,7500	Integer	No
X2	0	M	2,2500	Integer	No
Current	OBJ(Maximize)	= 41,2500	>= ZL =	-M	Non-integer

A célfüggvény értéke 41.25, $x_1=3.75$ és $x_2=2.25$
Mivel x_1 nem egész, ezért

szétválasztjuk a lehetséges megoldások halmazát két részre: az egyikben $x_1>=4$, a másikban pedig $x_1<=3$ lesz előírva. A program következő lépése az első halmazon optimalizál (és csak később tér rá a másodikra). **Branch-and-Bound Iteration, Next Iteration** paranccsal megkapjuk a **második** részredményt:

Decision Variable	Lower Bound	Upper Bound	Solution Value	Variable Type	Status
X1	4,0000	M	4,0000	Integer	Yes
X2	0	M	1,8000	Integer	No
Current	OBJ(Maximize)	= 41,0000	>= ZL =	-M	Non-integer

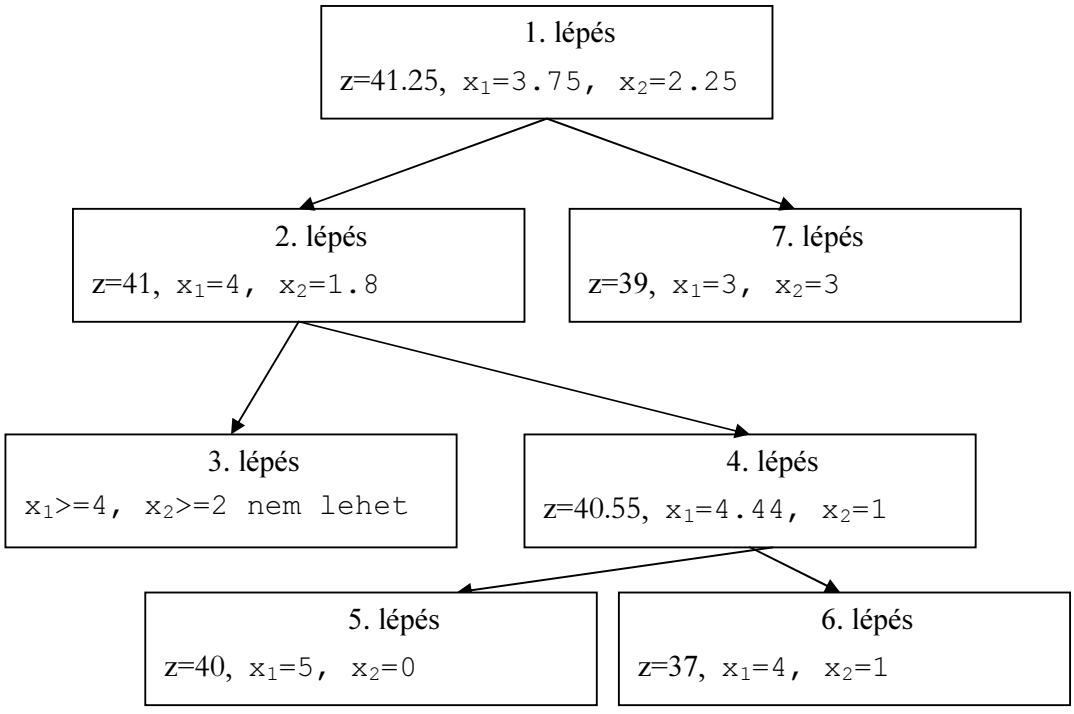
A célfüggvény értéke 41, $x_1=4$ és $x_2=1.8$

Status:**No** jelzi, hogy x_2 nem egész, ezért a lehetséges halmaz $x_1>=4$ részhalmazát két újabb részhalmazra bontjuk: az egyikben **$x_1>=4$, és $x_2>=2$,** a másikban pedig **$x_1>=4$, és $x_2<=1$** lesz előírva. A program következő lépése az első rész-részhalmazon optimalizál (és csak később tér rá a másodikra). **Branch-and-Bound Iteration, Next Iteration** paranccsal megkapjuk a **harmadik** részeredményt:

Decision Variable	Lower Bound	Upper Bound	Solution Value	Variable Type	Status
X1	4,0000	M		Integer	
X2	2,0000	M		Integer	
This	node	is	infeasible	!!!!!!	

Ez a rész-részhalmaz üres, $x_1>=4$, és $x_2>=2$ egyszerre nem teljesülhet.

A program ezután az **$x_1>=4$, és $x_2<=1$** rész-részhalmazon optimalizál, és így tovább. Eredményként részhalmazok és részeredmények egy fa-struktúrájú családjához jutunk. Mindegyiknél feltüntetjük, hogy hányadik részeredményről van szó (sorszám), mekkora a célfüggvényérték, mi az optimális megoldás (ha van), illetve azt, hogy nincs lehetséges megoldás.



Ahol mindkét változó értéke egész, ott nem folytatódik a fa, tehát valamilyen lehetséges megoldáshoz jutottunk. Ebben az esetben a részfeladat maximális célfüggvényértéke az eredeti feladat optimális célfüggvényértékének nyilvánvalóan alsó korlátja (innen van a *Bound* elnevezés). Ezeknek az alsó korlátoknak a maximumát hívjuk *aktuális alsó korlát*nak. Ahol az optimális megoldás valamelyik koordinátája nem egész, ott kettéágazik (innen van a *Branch* elnevezés) a fa, *feltéve, hogy a részfeladat maximum-értéke nem kisebb az aktuális alsó korlátnál.*
A lehetséges megoldásokhoz tartozó legnagyobb célfüggvényértékkel rendelkező csúcs(ok)ban található(ak) az optimális megoldás(ok). A mi esetünkben ez az 5. lépéshez tartozó megoldás. Az algoritmust gyorsan befejezhetjük a **Branch-and-Bound Iteration, Nonstop to Finish** paranccsal. **Results, Solution Summary** választással az alábbi **végeredményt** kapjuk.

Decision Variable	Solution Value	Unit Cost or Profit C(j)	Total Contribution	Reduced Cost	Basis Status
X1	5,0000	8,0000	40,0000	-1,0000	at bound
X2	0	5,0000	0	0	basic
Objective	Function	(Max.) =	40,0000		

A célfüggvény maximuma 40, $x_1=5$ és $x_2=0$

Megjegyezzük még, hogy a program megengedi, hogy a feladat elejére visszatérve a Format, Switch to Dual Form paranccsal áttérjünk a duális feladatra. Egészértékű feladatokra azonban –általában - az erős dualitás-tétel nem érvényes, tehát a duál célfüggvény minimuma nem szükségképpen egyenlő a primál célfüggvény maximumával.

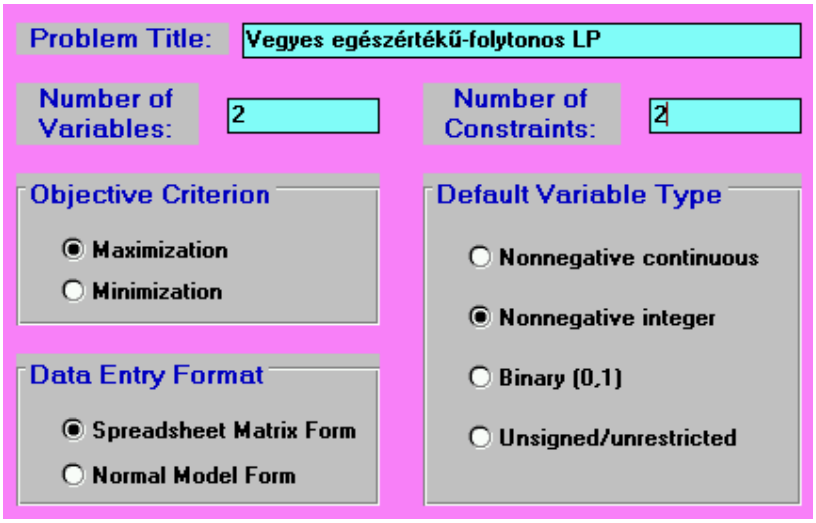
3.4.2 Második példaként oldjuk meg a következő vegyes folytonos-egészértékű feladatot!

$$(2x_1+ x_2) = z \rightarrow max$$

$$5x_1+ 2x_2 <= 8$$
$$x_1+ x_2 <= 3$$

$0<=x_1, x_2;$ x_1 egész, x_2 folytonos.

A **Linear and Integer Programming** modul elindítása után az *Új feladat megadása* ikonra kattintunk, és megadjuk az alapadatokat:



2 változó van és 2 korlátozó feltétel. A változók típusát nemnegatív egésznek választottuk, s csak a számadatok begépelésekor tudjuk x_2 típusát megváltoztatni.

OK

Begépeljük az együtthatókat:

Variable -->	X1	X2	Direction	R. H. S.
Maximize	2	1		
C1	5	2	<=	8
C2	1	1	<=	3
LowerBound	0	0		
UpperBound	M	M		
VariableType	Integer	Continuous		

x_2 típusát többszöri dupla kattintással választottuk ki a **Binary** (bináris), **Unrestricted** (kötetlen előjelű folytonos) illetve **Continuous** (folytonos nemnegatív) kínálatból.

A megoldás lépései ugyanazok, mint a 3.4.1 példánál. Válasszuk most a síbajnok ikonnal az azonnali megoldást (Results, Solution Summary)!

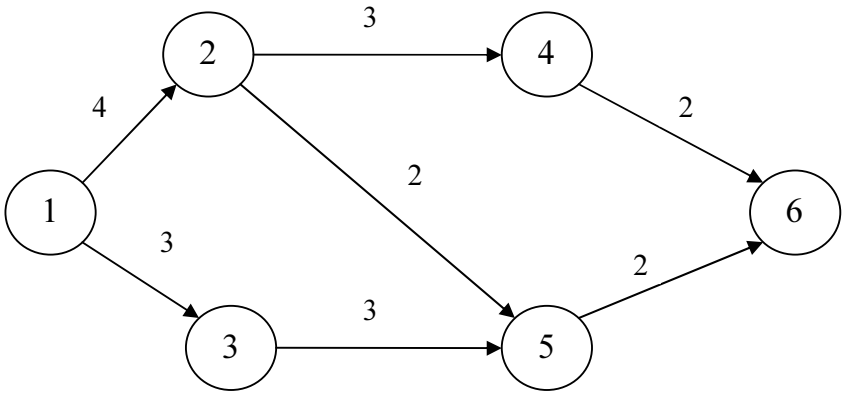
Decision Variable	Solution Value	Unit Cost or Profit C(j)	Total Contribution	Reduced Cost	Basis Status
X1	1,0000	2,0000	2,0000	-0,5000	at bound
X2	1,5000	1,0000	1,5000	0	basic
Objective	Function	(Max.) =	3,5000		

A célfüggvény optimális értéke 3.5, $x_1=1$ és $x_2=1.5$.

3.5 Hálózati modellek

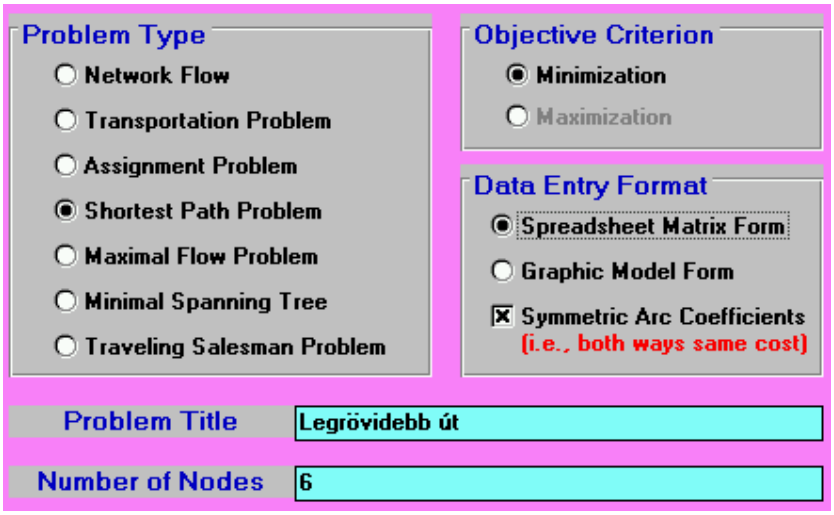
3.5.1 Legrövidebb út

Tekintsük a következő hálózatot!



A gráf pontjait városoknak tekintjük, ezek számozva vannak. Az élek útvonalakat jelölnek, s mindegyikre rá van írva az útvonal hossza. Keressük az 1-es városból a 6-os városba vezető legrövidebb útvonalat!

A **Network Modeling** modul elindítása után az *Új feladat megadása* ikonra kattintunk, és megadjuk az alapadatokat:



Legrövidebb út (Shortest Path) feladattal van dolgunk, 6 csúcsa van a gráfnak. Az adatokat mátrixos formában érdemes bevinni. A távolság-adatok szimmetrikusak, tehát A-ból B-be ugyanolyan hosszú az út, mint B-ből A-ba. Az útvonal(ak) hosszát minimalizálni fogjuk.

OK parancs után megadjuk a gráf adatait. Az Edit Node Names paranccsal megváltoztathatjuk a csúcsok neveit.

From	1	2	3	4	5	6
1		4	3			
2	4			3	2	
3	3				3	
4		3				2
5		2	3			2
6				2	2	

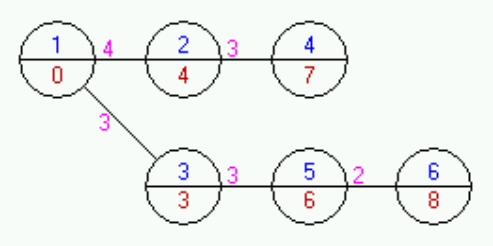
A mátrixnak csak a főátló feletti elemeit kell megadni, a többit automatikusan adja a program. (Törléskor sajnos nem törli a szimmetrikus elemet.) Az 1-esből a 2-esbe vezető út hossza 4, az 1-esből a 3-asba vezető út hossza 3, és így tovább. Érdemes tudni, hogy az **Edit, Add a Node**, illetve **Delete a Node** parancsokkal új csúcsot adhatunk a gráfhoz, illetve törölhetünk egy csúcsot, az **Edit, Problem Type**

paranccsal pedig módosíthatjuk a feladat típusát (például legrövidebb út helyett minimális feszítő fát kereshetünk). A megoldás előtt célszerű elmenteni a modellt. A síbajnok ikonra kattintva megadjuk, hogy melyik csúcstól melyik csúcsig akarunk eljutni, és azt, hogy a végeredmény érdekel (**Solve**), vagy az algoritmus egyes lépései is kíváncsiak vagyunk (*Display Steps*). Nézzük most a végeredményt az 1-es csúcstól a 6-os csúcsig!

From	To	Distance/Cost	Cumulative
1	3	3	3
3	5	3	6
5	6	2	8
From 1	To 6	=	8
From 1	To 2	=	4
From 1	To 3	=	3
From 1	To 4	=	7
From 1	To 5	=	6

Látható, hogy az 1-esből a 6-osba vezető legrövidebb útvonal az 1-3-5-6 útvonal, melynek együttes hossza 8. A program ezenkívül megadja az 1-esből valamennyi csúcsba vezető legrövidebb út hosszát.

A Results, Graphic Solution paranccsal grafikusan is megkapjuk a megoldást, ennek az automatikus síkbeli elrendezése azonban nem mindig szerencsés.

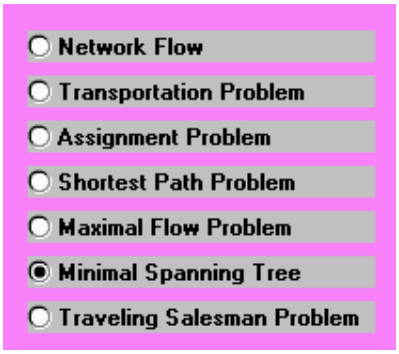


A felső félkörben lévő szám a csúcs sorszáma (színes képernyőn kékkel), alatta az adott csúcsnak az 1-es csúcstól vett távolsága (pirossal) látható.

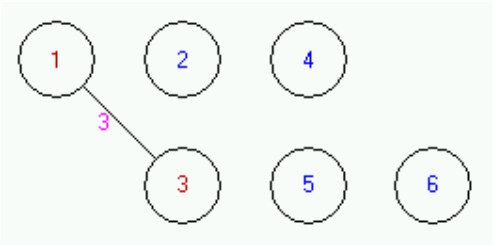
A gráf esetleges szerencsétlen elrendezését a feladat *Elejére* visszamenve az **Edit, Node** illetve **Edit, Arc/Connection/Link** parancsokkal módosíthatjuk.

3.5.2 Minimális feszítő fa

Most is az előző (3.5.1) pontban szereplő hálózatot fogjuk használni, de a feladat más. Feltételezzük, hogy az 1-es csúcsban van egy erőmű, a másik öt csúcs pedig városokat azonosít. Az erőműből mind az öt várost el kell látni elektromos árammal úgy, hogy a vezetékek együttes hossza minimális legyen. A szükséges vezetékek (gráfelméleti értelemben) fát fognak alkotni, amelyik minden csúcsba eljut. Így kapjuk a minimális feszítő fa feladatot.
A **Network Modeling** modul elindítása után két lehetőségünk van. Vagy az *Új feladat megadása* ikonra kattintunk, és megadjuk az alapadatokat ugyanúgy, mint az előző pontban, vagy pedig betöltjük az előző pont modelljét, és az **Edit, Problem Type** menüpontban a minimális feszítő fát (*Minimal Spanning Tree*) választjuk.

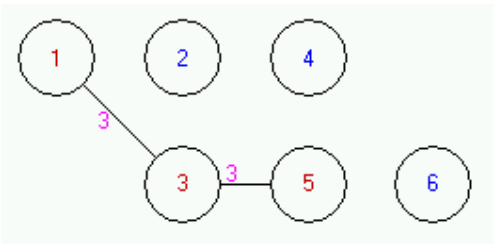


A minimális feszítő fa előállítására szolgáló algoritmus lépésről-lépésre bővíti azt a halmazt, amely már el van látva árammal. A bővítés során mindig egy olyan éllel bővítjük a fát, amelyik egy árammal ellátott és egy árammal még el nem látott csúcsot köt össze, és ezek között a legrövidebb. Az egyes lépéseket grafikusan megtekinthetjük a *Lépésenként* ikonra (ez a síbajnok szomszédja) kattintva, vagy pedig a *Solve and Analyze, Solve and Display Stpes-Network* paranccsal.



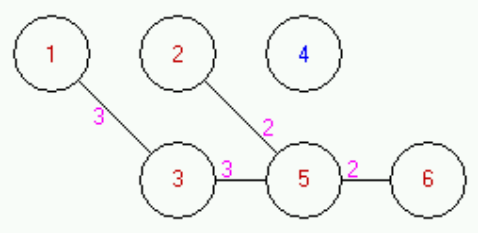
Az első lépésben a 3-as városba vezetjük be az áramot, hiszen ez van legközelebb az erőműhöz. Ezután az a város fog következni, amelyik az árammal már ellátott pontok halmazától a legkisebb távolságra fekszik, tehát az 5-ös.

Ismét a *Lépésenként* ikonra kattintva (vagy az *Iteration, Next Iteration* paranccsal)



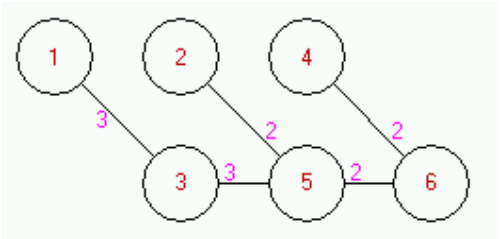
az 5-ös csúcs is bekerül a feszítő fába. Ezután két lehetőség van a bővítésre, a 2-es vagy a 6-os város, hiszen mindkettő 2 távolságra van az (1,3,5) halmaztól.

Kattintsunk kétszer a Lépésenként ikonra, s így hajtsunk végre két lépést!



Már csak a 4-es csúcs hiányzik a fából, és ide a 6-osból fogjuk bevezetni az áramot, hiszen közelebb van a 6-oshoz, mint bármely másik csúcshoz.

A végeredmény:

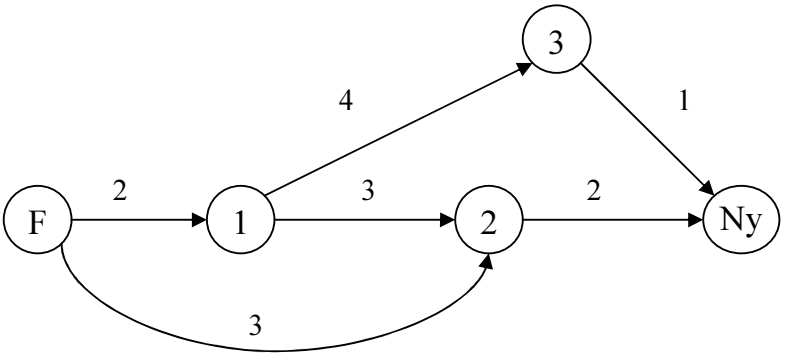


A minimális feszítő fa természetesen 6 csúcsból és 5 élből áll. A fa hossza:

`3+3+2+2+2=12.`

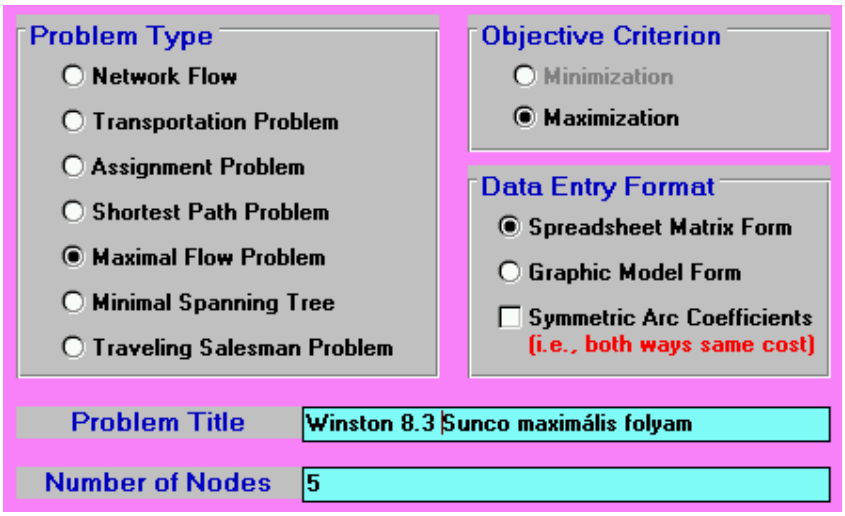
3.5.3 Maximális folyam

Tekintsük a következő hálózatot!



F az ún. *forrás*, **Ny** pedig a *nyelő*. Az élek csővezetékeket jelölnek, melyek egyirányúak, és a kapacitásuk adott. Például az 1-es csúcsból a 2-esbe maximálisan 3 egység olajat lehet vezetni. Az a kérdés, hogy hogyan lehet a forrásból a nyelőbe maximális mennyiségű olajat vezetni.

A **Network Modeling** modul elindítása után az *Új feladat megadása* ikonra kattintunk, és megadjuk az alapadatokat:



Maximális folyam feladatot oldunk meg egy 5 csúcsot tartalmazó hálózaton. Az adatokat mátrix formátumban célszerű bevinni. A mátrix nem szimmetrikus, mert egyirányú élek alkotják a hálózatot.

OK

Ezután mátrix formában megadjuk a hálózat adatait.

From \	Forrás	1	2	3	Nyelő
Forrás		2	3		
1			3	4	
2					2
3					1
Nyelő					

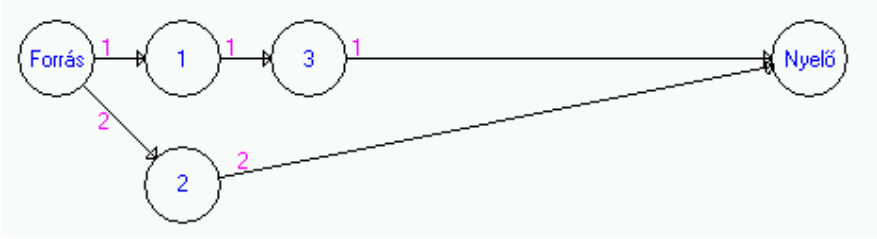
A csúcsok neveit az **Edit, Node Names** paranccsal adhatjuk (változtathatjuk) meg. A mátrix valamennyi eleme a főátló felett van, ami azt jelenti, hogy minden él a nagyobb sorszámú csúcs felé mutat.

A síbajnok ikonra kattintva megadjuk, hogy a forrástól a nyelőig szeretnénk eljutni (Solve), és máris megkapjuk a megoldást.

From	To	Net Flow		From	To	Net Flow
Forrás	1	1	4	2	Nyelő	2
Forrás	2	2	5	3	Nyelő	1
1	3	1				
Net Flow	From	Forrás	To	Nyelő	=	3

A forrásból az 1-es csúcsba 1 egység,
a forrásból a 2-es csúcsba 2 egység,
az 1-esből a 3-asba 1 egység,
a 2-esből a nyelőbe 2 egység,
a 3-asból a nyelőbe 1 egység folyik.

A folyam maximális értéke a forrástól a nyelőig 3 egység.
Könnyebben áttekinthető a megoldás, hogyha a gráf formátumot választjuk. **Results, Graphic Solution** parancs hatására az alábbi ábrát kapjuk.



A maximális folyam értéke a forrásból kimenő (vagy a nyelőbe vezető) élek terheléseinek összege, tehát 3.
Megemlítjük még, hogy ha a feladat típusát nem maximális folyamnak, hanem hálózati folyamnak (Network Flow) választjuk, akkor minimalizálni is lehet a célfüggvényt.

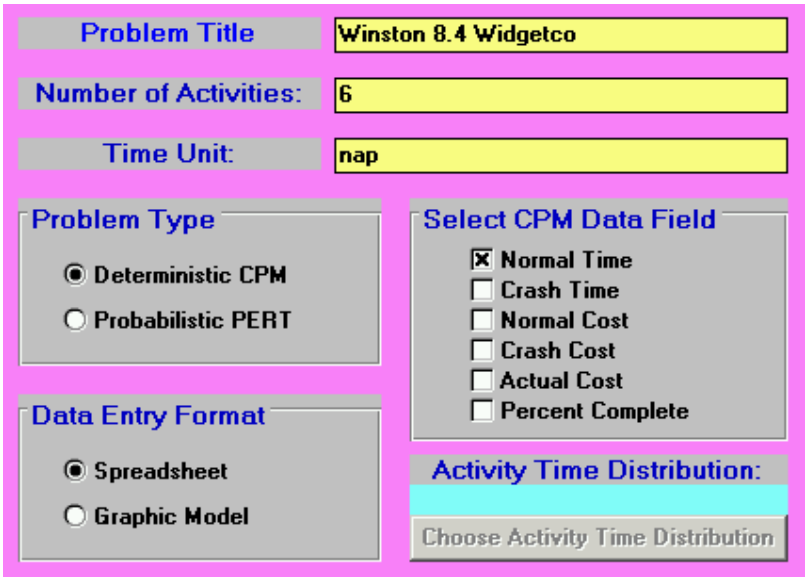
3.5.4 Kritikus út módszer (CPM)

Tekintsük a következő tevékenységekből álló projektet!

Kód	Tevékenység	Előzmények	Időtartam (nap)
A	Munkások kiképzése	-	6
B	Alapanyag beszerzése	-	9
C	1-es termék gyártása	A, B	8
D	2-es termék gyártása	A, B	7
E	2-es termék ellenőrzése	D	10
F	1-es és 2-es termék összeszerelése	C, E	12

Az előzmény azt jelenti, hogy pl. az 1-es termék gyártásának megkezdése előtt be kell fejezni a munkások kiképzését és az alapanyagok beszerzését. Hasonló módon a két termék összeszerelése csak akkor kezdődhet meg, ha az 1-es termék gyártása és a 2-es termék ellenőrzése (természetesen a 2-es termék gyártása után) már befejeződött. Adott, hogy az egyes tevékenységek elvégzéséhez mennyi időre van szükség. Az a kérdés, hogy melyik tevékenységet mikor lehet elkezdeni, ha azt akarjuk, hogy a teljes projekt időtartama minimális legyen.

A **PERT_CPM** modul elindítása után az *Új feladat megadása* ikonra kattintunk, és megadjuk az alapadatokat:



6 tevékenységből áll a projekt. Az időegység egy nap, a tevékenységek időtartama (ellentétben a PERT módszerrel) determinisztikus, és normális körülmények között értendő.
(A Crash Time az erőltetett menetben érvényes időtartamot jelenti.)
Az adatokat táblázatos formában érdemes megadni.
OK

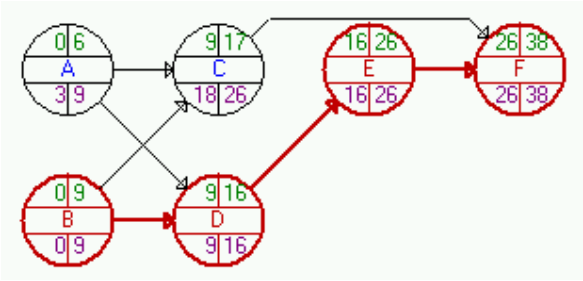
Ezután begépeljük a projektet leíró adatokat, és elmentjük a modellt.

Activity Name	Immediate Predecessor	Normal Time
A		6
B		9
C	A,B	8
D	A,B	7
E	D	10
F	C,E	12

Tevékenység-névként a kódot célszerű használni, különben áttekinthetetlenek lesznek a program által készített gráfok, sőt bonyolult tevékenység-nevek esetén az is előfordulhat, hogy végtelen ciklusba keveredik a program. A *Format, Switch to Graphic Model* paranccsal megkaphatjuk a feladat gráfját, a *Format, Switch to Matrix Form* paranccsal pedig visszatérhetünk a táblázathoz. **Solve and Analyze, Solve Critical Path** parancs (vagy a *síbajnok ikon*) hatására megkapjuk a feladat megoldását.

Activity Name	On Critical Path	Activity Time	Earliest Start	Earliest Finish	Latest Start	Latest Finish	Slack (LS-ES)
A	no	6	0	6	3	9	3
B	Yes	9	0	9	0	9	0
C	no	8	9	17	18	26	9
D	Yes	7	9	16	9	16	0
E	Yes	10	16	26	16	26	0
F	Yes	12	26	38	26	38	0
Project	Completion	Time	=	38	naps		
Number of	Critical	Path(s)	=	1			

Látható, hogy az egyes tevékenységeket mikor lehet legkorábban (**Earliest Start**) illetve legkésőbb (**Latest Start**) elkezdeni, illetve befejezni (**Finish**). A kritkus úthoz tartozó tevékenységeket **Yes** jelöli. Ezeknél a legkorábbi és legkésőbbi adatok azonosak. A kritikus út hossza 38 nap (s a többesszám jele az angol nyelvben, ezt automatikusan kiteszi a program), és csak egy kritikus utat talált a program, mégpedig a B-D-E-F útvonalat. A *Results* menüpontból érdemes megtekinteni a projekt Gantt-diagramját (*Gantt Chart*), vagy például a végeredményhez tartozó gráfot (*Graphic Activity Analysis*), amelyik esetünkben a következő:



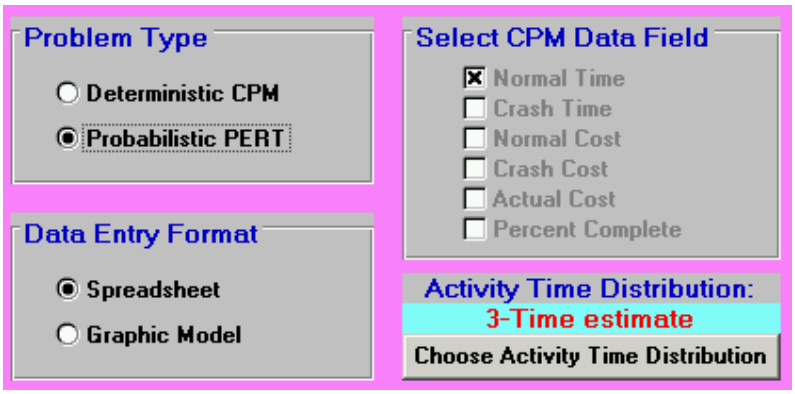
Felül a legkorábbi kezdés illetve befejezés időpontjait, alul pedig a legkésőbbi kezdés és befejezés időpontjait látjuk. A kritikus út színes monitoron piros színű, s a rajta lévő nyilak vastagabbak. A legfontosabb ismérv azonban az, hogy a kritikus úton lévő tevékenységek esetén a legkorábbi illetve legkésőbbi adatok azonosak.

3.5.5 PERT módszer

Ugyanazzal a projekttel foglalkozunk, mint az előző (3.5.4) pontban, pontosabban szólva ugyanazok a tevékenységek, és a megelőző tevékenységek. Ugyanakkor a tevékenységek időtartamai valószínűségi változók, adott eloszlással. A konkrét eredményeket arra az esetre ismertetjük, amikor mindegyik tevékenység időtartama egy adott intervallumon vehet fel értékeket. Az intervallum végpontjait **a**-val és **b**-vel jelöljük, a várható értéket pedig **m**-mel (ez általában **a** és **b** számtani közepe).

Kód	Tevékenység	Előzmények	a	m	b
A	Munkások kiképzése	-	2	6	10
B	Alapanyag beszerzése	-	5	9	13
C	1-es termék gyártása	A, B	3	8	13
D	2-es termék gyártása	A, B	1	7	13
E	2-es termék ellenőrzése	D	8	10	12
F	1-es és 2-es termék összeszerelése	C, E	9	12	15

Legegyszerűbb, ha az előző (CPM) feladat QSB modelljében az *Elejére* ikonra kattintva visszalépünk a feladat elejére (vagy újra betöltjük), és az **Edit, Project Specification** paranccsal megváltoztatjuk a feladat típusát:



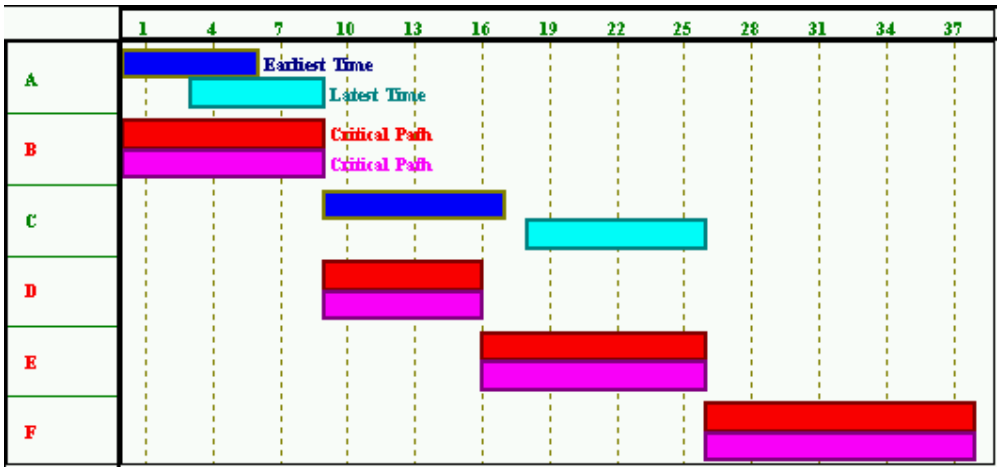
Ezúttal a valószínűségszámítási modellt választjuk, a tevékenységek időtartamának eloszlását 3 paraméterrel jellemezzük. Ezek közül **a** a legkisebb (optimista időtartam), **m** a középső (legvalószí-nűbb időtartam), és **b** a legnagyobb (pesszimista időtartam). Megemlítjük még, hogy a *Choose Activity Time Ditribution* ikonra kattintva 21 különböző eloszlás közül választhatunk.
OK parancsra megjelenik az a táblázat, ahova most csak a hiányzó értékeket (**a** és **b** értékeit) kell begépelni.

Activity Name	Immediate Predecessor	Optimistic time (a)	Most likely time (m)	Pessimistic time (b)
A		2	6	10
B		5	9	13
C	A,B	3	8	13
D	A,B	1	7	13
E	D	8	10	12
F	C,E	9	12	15

A *Format, Switch to Graphic Model* paranccsal megkaphatjuk a feladat gráfját, a *Format, Switch to Matrix Form* paranccsal pedig visszatérhetünk a táblázathoz. **Solve and Analyze, Solve Critical Path** parancs (vagy a *sibajnok ikon*) hatására megkapjuk a feladat megoldását.

Activity Name	On Critical Path	Activity Mean Time	Earliest Start	Earliest Finish	Latest Start	Latest Finish	Slack (LS-ES)	Activity Time Distribution	Standard Deviation
A	no	6	0	6	3	9	3	3-Time estimate	1,3333
B	Yes	9	0	9	0	9	0	3-Time estimate	1,3333
C	no	8	9	17	18	26	9	3-Time estimate	1,6667
D	Yes	7	9	16	9	16	0	3-Time estimate	2
E	Yes	10	16	26	16	26	0	3-Time estimate	0,6667
F	Yes	12	26	38	26	38	0	3-Time estimate	1
Project	Completion	Time	=	38	naps				
Number of	Critical	Path(s)	=	1					

Látható, hogy az egyes tevékenységeket normál időtartam esetén mikor lehet legkorábban (**Earliest Start**) illetve legkésőbb (**Latest Start**) elkezdeni, illetve befejezni (**Finish**). Az utolsó oszlopban a tevékenységi idők szórásait láthatjuk. A kritkus úthoz tartozó tevékenységeket **Yes** jelöli. Ezeknél a legkorábbi és legkésőbbi adatok azonosak. A kritikus út hossza 38 nap és csak egy kritikus utat talált a program. A *Results* menüpontból érdemes megtekinteni a végeredményhez tartozó gráfot (*Graphic Activity Analysis*), vagy pl. projekt Gantt-diagramját (*Gantt Chart*), amelyik esetünkben a következő:



Itt a legkorábbi illetve legkésőbbi végrehajtás időintervallumait láthatjuk, színes monitoron különböző színekkel. A kritikus út színes monitoron piros színű. A legfontosabb ismérv azonban az, hogy a kritikus úton lévő tevékenységek esetén a legkorábbi illetve legkésőbbi adatok azonosak.

Megemlítjük még, hogy a **Results, Perform Probability Analysis** paranccsal megkaphatjuk annak valószínűségét, hogy a projektet T idő alatt végre lehet hajtani. T egy tetszőleges input adat, amit a *Desired Completion Time* in nap kérdés melletti cellába gépelünk be (*Compute Probability*). Például T= 39 esetén a valószínűség 0.6451, T=37 esetén csak 0.3549, és a kritikus út egyik esetben sem változik.

3.6 Döntésanalízis

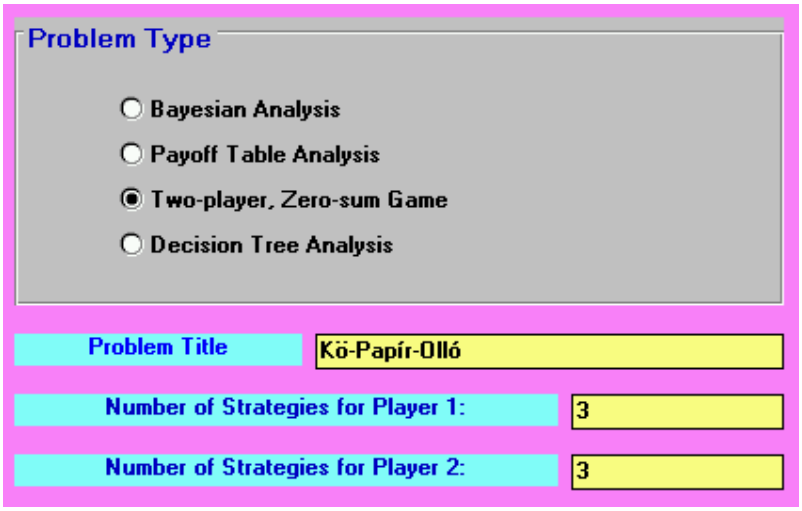
3.6.1 Játékelmélet

Tekintsük a közismert kő-papír-olló játékot! Ketten játszanak úgy, hogy kezükkel a kő, a papír és az olló jele közül valamelyiket felmutatják, mégpedig egyszerre (tehát egyik sem tudja előre, hogy mit mutat fel a másik). A kő üti az ollót (hiszen az olló kicsorbulhat), az olló üti a papírt (mert el tudja vágni a papírt), a papír pedig üti a követ (mert bele lehet csomagolni a követ). Az alábbi táblázat első oszlopa azt mutatja, hogy az 1. játékos mit választott, az első sor pedig a 2. játékos választásait tartalmazza. A táblázatban az 1. játékos nyeresége (illetve a második vesztesége) van feltüntetve.

	Kő	**Papír**	**Olló**
Kő	0	-1	1
Papír	1	0	-1
Olló	-1	1	0

Ha a játékosok a kő-papír játékot játszanák, akkor a számtáblázatnak csak az első két sorát és az első két oszlopát kellene figyelembe vennünk. Ebben a 2x2-es táblázatban azonnal találunk egy *nyeregpontot*, vagyis olyan cellát, amelynek a tartalma a *saját oszlopában maximális*, és a *saját sorában pedig minimális*. Ilyen a 2. sor 2. cellája. Ez azt jelenti, hogy amikor mindkét játékos a papírt választja, akkor mindketten örülhetnek annak, hogy nem a követ választották. Ezt úgy mondjuk, hogy a 2x2-es játéknak egyensúlyi pontja van a *tiszta stratégiák* körében.
Ugyanakkor a 3x3-as játék nyereség-táblájában nem találunk nyeregpontot, tehát nincs a játéknak olyan kimenetele, amikor mindkét játékos örülhet annak, hogy nem egy másik jelet választott. Ezt úgy mondjuk, hogy a *tiszta stratégiák* körében a kő-papír-olló játéknak nincs egyensúlyi pontja. Van viszont *egyensúlyi pont* a kevert stratégiák körében, és ezt ki is számítja a QSB program.

A **Decision Analysis** modul elindítása után az *Új feladat megadása* ikonra kattintunk, és megadjuk az alapadatokat:



Kétszemélyes, zérus-összegű játékról van szó. Mindkét játékosnak 3 stratégiája (választása) van.

OK

Ezután begépeljük a nyereségtábla értékeit, és elmentjük a modellt.

Player1 \ Player2	Kö	Papír	Olló
Kő	0	-1	1
Papír	1	0	-1
Olló	-1	1	0

Az egyes stratégiák neveit az **Edit, Player1 Strategy Name** illetve **Edit, Player2 Strategy Name** parancsokkal lehet megváltoztatni.
A síbajnok ikonra kattintva (vagy *Solve and Analyze, Solve the Problem* paranccsal) megkapjuk a megoldást.

Player	Strategy	Dominance	Elimination Sequence
1	Kő	Not Dominated	
1	Papír	Not Dominated	
1	Olló	Not Dominated	
2	Kö	Not Dominated	
2	Papír	Not Dominated	
2	Olló	Not Dominated	
Player	**Strategy**	**Optimal Probability**	
1	Kő	0,33	
1	Papír	0,33	
1	Olló	0,33	
2	Kö	0,33	
2	Papír	0,33	
2	Olló	0,33	
Expected	Payoff	for Player 1 =	0

Ezek szerint egyik játékosnak sincs *domináns stratégiája*, tehát olyan választása, melynél a másik játékos bármelyik választása mellett legalább akkora lenne a nyeresége, mint egy másik választás esetén. (Tehát a nyereség-táblában egyik sorvektor sem >= egy másik sorvektornál, és egyik oszlopvektor sem <= egy másik oszlopvektornál)
Ugyanakkor mindkét játkosnak van *optimális kevert stratégiája*, tehát létezik *egyensúlyi stratégia-pár*. Ezt úgy tudják megvalósítani, ha hosszútávon 1/3, 1/3, 1/3 valószínűséggel választják mind a három jelet. Ebben az esetben az első játékos nyereségének és a második játékos veszteségének a várható értéke 0. Ezt úgy mondjuk, hogy *a játék értéke 0*.
Megemlítjük még, hogy ha visszatérünk a feladat elejére, akkor a **Solve and Analyze, Perform Zero-sum Game Simulation** parancs (vagy a szimuláció ikon) hatására, a program véletlenszám-generátor segítségével lejátszik a játékból néhány ezer (vagy tízezer) menetet, és kiszámítja a nyereség empírikus várható értékét.

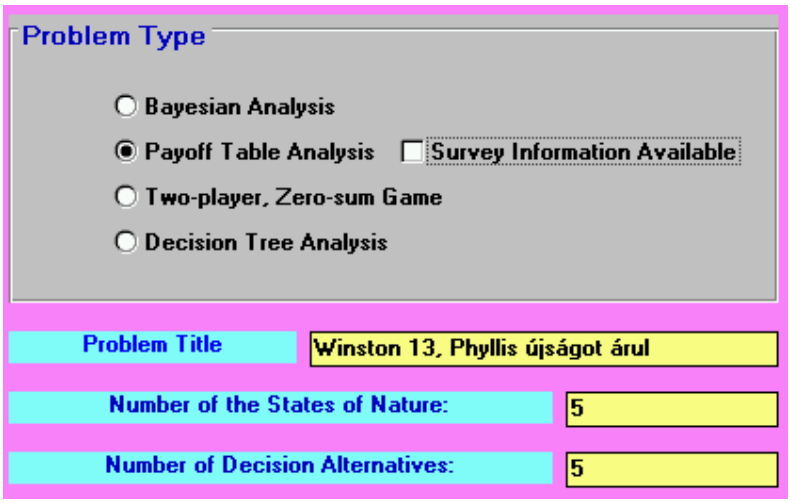
3.6.2 Payoff elemzés (Döntési szabályok alkalmazása)

Tekintsük a következő problémát! Egy újságárus tudja, hogy az eladott újságok száma minden nap 6 és 10 között van, és *mindegyik érték 1/5 valószínűséggel fordul elő*. Az a kérdés, hogy hány újságot rendeljen holnapra, 6-ot, 7-et, 8-at 9-et vagy 10-et. Az újságot 20 centért veszi és 25 centért adja el. Nyereségét az alábbi táblázat mutatja.

kereslet / rendelés	6	7	8	9	10
6	30	30	30	30	30
7	10	35	35	35	35
8	-10	15	40	40	40
9	-30	-5	20	45	45
10	-50	-25	0	25	50

Például, ha 7-et rendelt, és 6 lesz a kereslet, akkor 6-ot el tud adni, s ezáltal 6*5=30 cent nyereségre tesz szert, de 1 újság rajta marad, ami 20 cent felesleges kiadás, ezért a nyeresége csak 10 cent.

A **Decision Analysis** modul elindítása után az Új feladat megadása ikonra kattintunk, és megadjuk az alapadatokat.



Payoff elemzést végzünk, 5 állapot (*State of Nature*) következhet be, és 5 döntési lehetőségünk (*Decision Alternatives*) van.

OK

Ezután megadjuk az előzetes valószínűségeket, és a nyereségtáblázat elemeit.

Decision \ State	6	7	8	9	10
Prior Probability	0.2	0.2	0.2	0.2	0.2
6	30	30	30	30	30
7	10	35	35	35	35
8	-10	15	40	40	40
9	-30	-5	20	45	45
10	-50	-25	0	25	50

Az állapotok neveit az **Edit, State of Nature** paranccsal, a döntési lehetőségek neveit pedig az **Edit, Decision Alternative Name** paranccsal változtathatjuk meg. Ezen a ponton érdemes elmenteni a modellt.

Amikor a megoldás előállítása céljából a síbajnok ikonra kattintunk (vagy a *Solve and Analyze, Solve the Problem* parancsot kiadjuk), megkérdezi a program, hogy a Hurwicz kritériumnál mekkora optimizmussal számoljon. Ez egy 0 és 1 közötti p szám, ami azt jelenti, hogy a célfüggvény egy súlyozott számtani közép, amiben p súllyal szerepel a maximális nyereség, és (1-p) súllyal a minimális nyereség. A megoldás sorait az egyes döntési szabályokra külön-külön kell értelmezni:

05-27-2003 Criterion	Best Decision	Decision Value	
Maximin	6	$30	
Maximax	10	$50	
Hurwicz (p=0,5)	6	$30	
Minimax Regret	6	$20	
Expected Value	6	$30	
Equal Likelihood	6	$30	
Expected Regret	6	$10	
Expected Value	without any	Information =	$30
Expected Value	with Perfect	Information =	$40
Expected Value	of Perfect	Information =	$10

Maximin azt jelenti, hogy a legrosszabb állapot bekövetkezésére számítunk, s az ehhez tartozó nyereséget akarjuk maximalizálni, s ezért 6 újságot rendelünk.

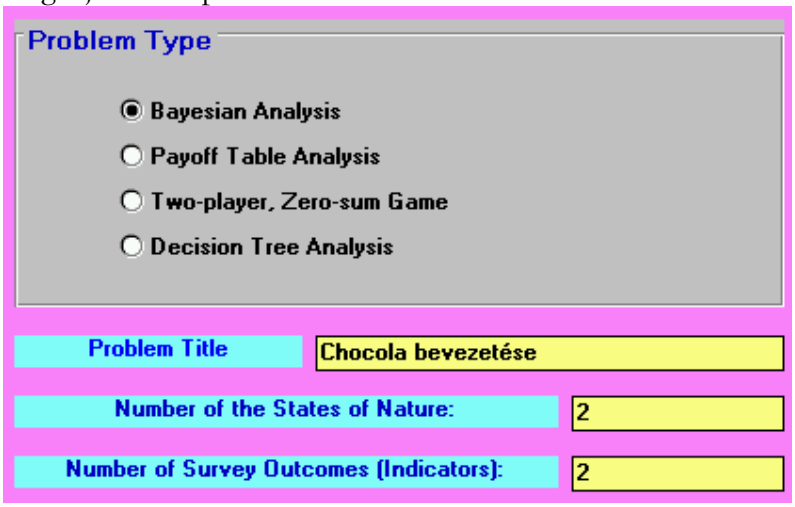
Maximax esetén arra számítunk, hogy a legjobb állapot fog bekövetkezni, ezért rendelünk 10-et.

A **Hurwicz** célfüggvény folytonos átmenetet biztosít a *Maximin* (p=0) és a *Maximax* (p=1) között.

A **Regret** szót itt *csalódás*nak vagy *elmulasztott nyereség*nek lehet fordítani. Ez is egy mátrix, amit úgy kapunk az eredeti 5x5-ös nyereségmátrixból, hogy mindegyik elemet kivonjuk a saját sorának a maximumából (ennyivel lett kevesebb a nyereségünk az elképzelhető maximálisnál). A program a **Results, Show Regret Table** parancsra előállítja ezt a csalódás-mátrixot. A **Minimax Regret** azt a döntést jelenti, hogy a csalódás maximumát akarjuk minimalizálni. **Expected Value** a nyereség *várható értéke*, **Expected Regret** pedig az elmulasztott nyereség *várható értéke* (ezt nyilvánvalóan minimalizálni akarjuk). Javasoljuk, hogy a többi célfüggvény jelentését keresse meg az olvasó a **Help Glossary** fejezetében.
Megemlítjük még, hogy *Results, Payoff Table Analysis* parancsra minden lehetséges döntéshez célfüggvények szerinti bontásban megtekinthetjük döntésünk számszerű következményét.

3.6.3 Bayes elemzés (Valószínűségek számítása)

Tekintsük a következő feladatot! Egy cég új üdítőital bevezetését fontolgatja. Piackutatás nélkül előzetesen úgy tűnik, hogy az új ital bevezetése **0.55** valószínűséggel lesz *országos siker*, és **0.45** valószínűséggel lesz *országos kudarc*. Lehetőség van (bizonyos költség mellett) helyi piackutatásra, ami előre jelzi, hogy az adott régióban az új ital bevezetése sikeres lenne, vagy sem. Korábbi piackutatások azt mutatják, hogy **országos siker esetén** a helyi piackutatás eredménye **0.9273** valószínűséggel *volt helyi siker*, és **0.0727** valószínűséggel *volt helyi kudarc*. Ugyanakkor **országos kudarc esetén** a helyi piackutatás eredménye **0.2** valószínűséggel *volt helyi siker*, és **0.8** valószínűséggel *volt helyi kudarc*.
A cégnek a döntéselemzéshez a *fordított feltételes valószínűségekre* van szüksége, tehát a felmérés **helyi sikere esetén** mekkora az *országos siker* és az *országos kudarc* valószínűsége, továbbá a felmérés **helyi kudarca esetén** mekkora az *országos siker* és az *országos kudarc* valószínűsége. Azt is tudni kell, hogy **általában** mekkora valószínűséggel lesz *helyi siker* illetve *helyi kudarc* a felmérés eredménye.
A **Decision Analysis** modul elindítása után az *Új feladat megadása* ikonra kattintunk, és megadjuk az alapadatokat.



Bayes elemzést végzünk.
Két **állapot**tal van dolgunk (az országos siker, illetve az országos kudarc), és a felmérésnek két lehetséges *kimenetele* van (helyi siker illetve helyi kudarc).
OK

Ezután begépeljük az ismert valószínűségeket, és elmentjük a modellt.

Outcome \ State	orsz. siker	orsz. kudarc
Prior Probability	0.55	0.45
helyi siker	0.9273	0.2
helyi kudarc	0.0727	0.8

Ügyeljünk arra, hogy az előzetes valószínűségek sorösszege 1, a feltételes valószínűségek esetén viszont az oszlopösszegek értéke 1. A program a százalékjelet nem fogadja el. Az állapotok neveit az **Edit, State of Nature Name**, a felmérés eredményeinek neveit pedig az **Edit, Survey Outcome/Indicator Name** paranccsal változtathatjuk meg.
A síbajnok ikonra kattintva (vagy *Solve and Analyze*, *Solve the Problem* paranccsal) megkapjuk a megoldást.

Indicator\State	orsz. siker	orsz. kudarc
helyi siker	0,85	0,15
helyi kudarc	0,1000	0,9000

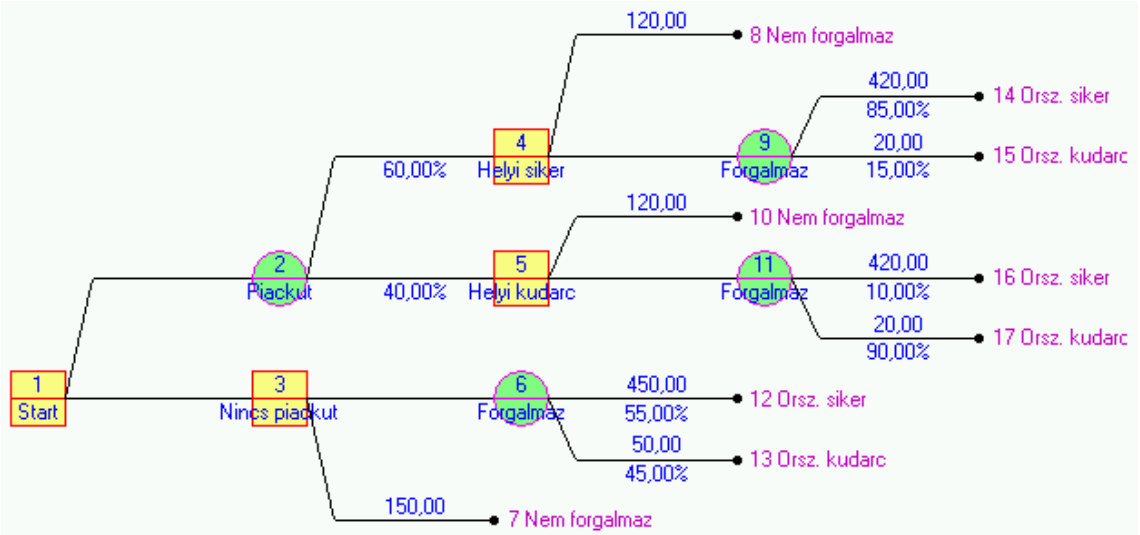
Ezúttal a valószínűségek sorösszege 1. Tehát: ha **helyi siker** a felmérés eredménye, akkor **0.85** valószínűséggel lesz *országos siker*, és **0.15** valószínűséggel lesz *országos kudarc*. Ha viszont a helyi felmérés eredménye **helyi kudarc**, akkor **0.1** valószínűséggel lesz *országos siker*, és **0.9** valószínűséggel számíthatunk *országos kudarcra*.
A **Results, Show Marginal Probability** paranccsal kapjuk meg azt, hogy a helyi felmérésnek az eredménye (úgy általában) **0.6** valószínűséggel lesz *helyi siker* és **0.4** valószínűséggel lesz *helyi kudarc*.
Megemlítjük még, hogy a **Results** menü *Show Joint Probability* ágában megtekinthetők az ún. *együttes valószínűségek* is. Például 0.51 annak a valószínűsége, hogy a felmérés eredménye helyi siker, és ugyanakkor az új üdítőital bevezetésének az eredménye országos siker.

3.6.4 Döntési fa

Itt az előző pont (3.6.3) feladatát fogjuk folytatni, ezért javasoljuk, hogy az olvasó csak annak elolvasása után térjen rá az itteni döntési fa feladat megoldására. Tehát egy cég új üdítőital bevezetését fontolgatja. Három lehetőség közül választhat:
1. Helyi piackutatást végeztet, és ennek eredményétől függően dönt az ital országos forgalmazásáról.
2. Helyi piackutatás nélkül úgy dönt, hogy az új italt országosan forgalmazni fogja.
3. Helyi piackutatás nélkül úgy dönt, hogy az új italt nem fogja forgalmazni.

A cég tőkéje jelenleg 150, országos siker esetén ez 300-zal fog nőni, országos kudarc esetén pedig 100-zal csökken (mindegyik adat ezer dollárban értendő). A helyi piackutatás költsége 30 (ezer dollár).
Ismertek továbbá a következő valószínűségek: A helyi piackutatás 0.6 valószínűséggel jósol helyi sikert, és 0.4 valószínűséggel helyi kudarcot. Ha a helyi előrejelzés helyi siker, akkor az országos siker valószínűsége 0.85, tehát az országos kudarc valószínűsége 0.15. Ha viszont helyi előrejelzés helyi kudarc, akkor az országos siker valószínűsége 0.1, tehát az országos kudarc valószínűsége 0.9. Az eddigiekből már következik (lásd előző pont), hogy ha nem végzünk piackutatást, akkor (úgy általában) az országos siker valószínűsége 0.55, az országos kudarc valószínűsége pedig 0.45.

Mindezek alapján a következő döntési fát kell elkészítenünk:



Itt a négyzet alakú csúcsok döntési elágazást jelölnek, a köralakúak pedig véletlen jellegű elágazások. A leveleknél nincsen se kör, se négyzet. A fának 17 csúcsa van, ezek oszlopfolytonosan vannak megszámozva. Véletlen jellegű elágazásnál látható az egyes lehetőségek valószínűsége, ezenkívül minden levélnél látható a cég tőkéjének értéke. Például a 10-es csúcs esetén a 120 úgy jön létre, hogy az eredeti 150-ből levonjuk a piackutatásra elköltött 30-at. A 17-es csúcsban a 20 ezer dollár úgy jön létre, hogy 150-ből piackutatásra költöttünk 30-at, azután pedig 100-at elvitt az országos kudarc.

A **Decision Analysis** modul elindítása után az Új feladat megadása ikonra kattintunk, és megadjuk az alapadatokat.

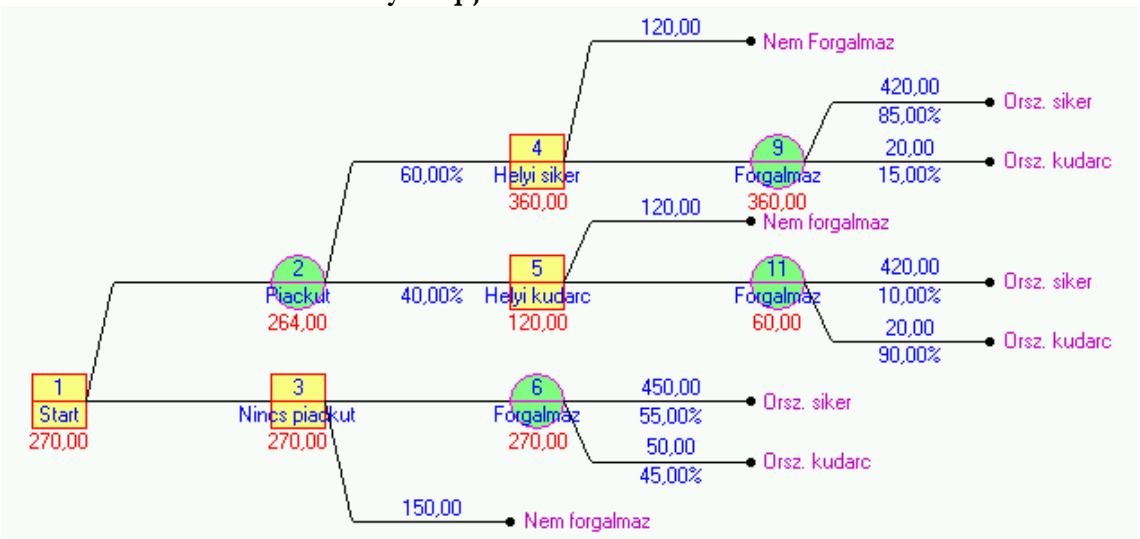
Döntési fát készítünk, aminek 17 csúcsa lesz. OK

A következő lépés a döntési fa adatainak begépelése. Döntési elágazásnál D betűt (Decision), véletlen jellegű elágazásnál C betűt (Chance) kell megadni. a Ezután begépeljük a döntési fa adatait, és elmentjük a modellt.

Node/Event Number	Node Name or Description	Node Type (enter D or C)	Immediate Following Node (numbers separated by ',')	Node Payoff (+ profit, - cost)	Probability (if available)
1	Start	D	2,3		
2	Piackut	C	4,5		
3	Nincs piackut	D	6,7		
4	Helyi siker	D	8,9		0.6
5	Helyi kudarc	D	10,11		0.4
6	Forgalmaz	C	12,13		
7	Nem forgalmaz			150	
8	Nem Forgalmaz			120	
9	Forgalmaz	C	14,15		
10	Nem forgalmaz			120	
11	Forgalmaz	C	16,17		
12	Orsz. siker			450	0.55
13	Orsz. kudarc			50	0.45
14	Orsz. siker			420	0.85
15	Orsz. kudarc			20	0.15
16	Orsz. siker			420	0.10
17	Orsz. kudarc			20	0.90

Minden elágazásnál megadjuk a közvetlenül utána következő két csúcs sorszámát. Véletlenszerű elágazás esetén az ágak végpontjaihoz tartozó csúcs sorában (tehát nem az elágazási pontnál) adjuk meg az adott ág valószínűségét. Itt használjuk fel a 3.6.3 pontban meghatározott valószínűségeket. A fa levelei esetén a Payoff rovatban tüntetjük fel a cég tőkéjének aktuális értékét. A csúcsok neveit közvetlen begépeléssel határozhatjuk meg.

A megoldást a *Solve and Analyze*, ***Draw Decision Tree*** paranccsal indítjuk (nem a síbajnok ikonnal), mert ebben az esetben kérhetjük, hogy minden csúcsnál tüntesse fel a program a **tőke várható értékét is**: **Display the expected values for each node or event**. **A következő eredményt kapjuk:**



Az eredmény magától értetődő. Maximálisan 270 ezer dollár lehet a tőke várható értéke. Ezt úgy érhetjük el, hogy a start pontból minden döntési elágazásnál a nagyobb várható értékű csúcsba lépünk. Tehát nem végzünk piackutatást, és országosan forgalmazzuk az új italt. A 270-es érték 450-nek és 50-nek a súlyozott számtani közepe 0.55 illetve 0.45 súlyokkal.

A **Results, Show Decision Tree Analysis** paranccsal táblázatos formában tekinthetjük meg ugyanezeket az eredményeket.

3.7 Sorbanállási feladat

Vizsgáljuk a kövekező feladatot! Egy bankba átlagosan 3 percenként érkezik egy ügyfél, hogy ott pénzt fizessen be, pénzt vegyen fel, vagy pénzt váltson. A befizetés átlagosan 2 percig, a pénzfelvétel átlagosan 3 percig, a pénzváltás pedig átlagosan 1 percig tart. Előzetes megfigyelés szerint az ügyfelek 40 százaléka befizet, 30 százaléka vesz fel, és 30 százaléka vált pénzt. A sor hossza legfeljebb 6, tehát a rendszer hossza legfeljebb 7.

A feladat az ún. M/M/s családba tartozik, ami azt jelenti, hogy:

1. Az ügyfelek érkezései közötti idők függetlenek, és exponenciális (Markov-féle) eloszlásúak.
2. A kiszolgálási idők is függetlenek és azonos (az előbbitől eltérő) exponenciális eloszlásúak
3. A kiszolgálók száma s.

Ezt a feladattípust explicit képletekkel oldja meg a program, más jellegű feladatok esetén közelítő módszereket alkalmaz.Először kiszámítjuk a legfontosabb paramétereket. Idő-egységnek a percet választjuk.

A kiszolgálás várható hossza T=2*0.4+3*0.3+1*0.3=2 perc.

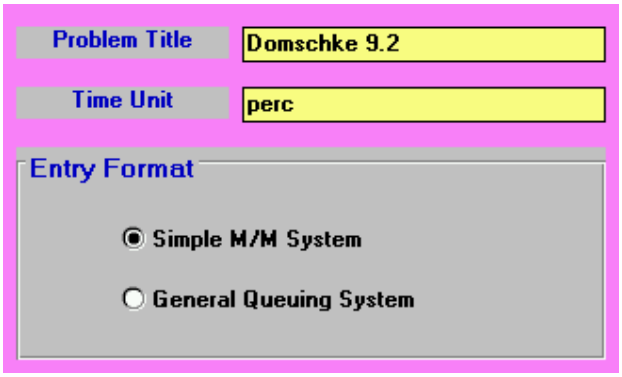
μ=1/2 lesz az *időegység alatt kiszolgált ügyfelek várható száma.* (Service rate per server per perc).

λ=1/3 pedig az *időegységre eső érkezések várható száma.*(Customer arrival rate per perc).

A *sor kapacitása* (Queue capacity) **6**.

Oldjuk meg a feladatot az s=1 (egy kiszolgáló személy) esetre!

A **Queuing Analysis** modul elindítása után az *Új feladat megadása* ikonra kattintunk, és megadjuk az alapadatokat:



Időegységnek a percet választjuk. Egyszerű M/M/s típusú feladatról van szó.

OK

Ezután begépeljük az input-adatokat, és elmentjük a modellt.

Data Description	ENTRY
Number of servers	1
Service rate (per server per perc)	0.5
Customer arrival rate (per perc)	0.333333
Queue capacity (maximum waiting space)	6
Customer population	M
Busy server cost per perc	
Idle server cost per perc	
Customer waiting cost per perc	
Customer being served cost per perc	
Cost of customer being balked	
Unit queue capacity cost	

Tehát 1 kiszolgáló van, percenként átlagosan **μ=0.5** ügyfelet tud ksizolgálni, az egy perc alatt érkező ügyfelek várható száma pedig **λ=0.3333.**
Legfeljebb hatan állhatnak be a sorba, ugyanakkor az ügyfelek számának nincs felső korlátja.
A többi sorban különböző költségeket lehetne megadni. Például a *Cost of Customer being bulked* annak a költsége, ha egy ügyfél érkezésekor a sor már telített, ezért az ügyfél visszafordul.
A síbajnok ikonra történő kattintás (vagy *Solve and Analyze, Solve the Performance* parancs) hatására a program kiértékeli a rendszert. A jelentés igen terjedelmes, ezért a költségeket tartalmazó sorokat ezúttal kihagyjuk.

System: M/M/1/7	From Formula
Customer arrival rate (lambda) per perc =	0,3333
Service rate per server (mu) per perc =	0,5000
Overall system effective arrival rate per perc =	0,3266
Overall system effective service rate per perc =	0,3266
Overall system utilization =	65,3132 %
Average number of customers in the system (L) =	1,6752
Average number of customers in the queue (Lq) =	1,0220
Average number of customers in the queue for a busy system (Lb) =	1,5648
Average time customer spends in the system (W) =	5,1297 percs
Average time customer spends in the queue (Wq) =	3,1297 percs
Average time customer spends in the queue for a busy system (Wb) =	4,7918 percs
The probability that all servers are idle (Po) =	34,6868 %
The probability an arriving customer waits (Pw) or system is busy (Pb) =	65,3132 %
Average number of customers being balked per perc =	0,0068

7=6+1 a rendszer hossza.
Felismerhető **λ és μ** értéke.
L=1.6752
Lq=1.0220
Lb=1.5648
W=5.1297
Wq=3.1297
Wb=4.7918
Po=0.3468
Pw=Pb=0.65

Az adatokat a következőképpen értelmezzük. Átlagosan **L** ügyfél van a bankban, közülük átlagosan **Lq** áll a sorban. Egy ügyfél átlagosan **W** időt tölt a bankban, ebből **Wq** ideig áll sorban. **Po** annak a valószínűsége, hogy egyik kiszolgálónak sincs munkája. **Pw=Pb** annak a valószínűsége, hogy az új ügyfélnek várnia kell, mert mindegyik kiszolgáló foglalt. **Wb=Wq/Pw** azt mutatja, hogy átlagosan mennyit kell várnia egy ügyfélnek, amikor foglalt a rendszer (ebben az átlagban a 0 nincs benne, azért nagyobb az eredmény a Wq-nál). **Lb=Lq/Pw** azt mutatja, hogy átlagosan hány ügyfél áll a sorban, amikor foglalt a rendszer (ebből az átlagból kimarad az, amikor senki nincs a sorban és nem foglalt a rendszer). Egy perc alatt átlagosan 0.0068 ügyfél *fordul vissza* azért, mert nincs hely a sorban. Végül a rendszer *átlagos kihasználtsága* (*Overall system utilization*) **1-Po=0.6531**, tehát 65.31 százalék., ami annak a valószínűsége, hogy legalább egy ügyfél bent van a bankban. Abban az esetben, amikor a sor hossza nem korlátos, ennek a kihasználtságnak (illetve hatékonyságnak) az értéke **λ/μ** (több kiszolgáló esetén **λ/μs).** Amikor ez az érték 1-nél kisebb, akkor a rendszer működőképes, ha viszont 1-nél nagyobb, akkor a rendszer véges időn belül "kipukkad".

A **Results, Probability Summary** paranccsal minden n-re megkapjuk annak a valószínűségét, hogy éppen n ügyfél van a bankban. Jelen esetben a következő értékek adódnak:

J	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*
$\mathbf{P_j}$	0.3469	0.2312	0.1542	0.1028	0.0685	0.0457	0.0305	0.0203

Megemlítjük még, hogy a feladat elejére visszatérve, a *Number of servers* sorban módosíthatjuk a kiszolgálók számát (azaz s értékét), s ezután újra lefuttathatjuk a programot. Például s=2 esetén a fenti táblázat a következőképpen módosul.

Annak valószínűsége, hogy j ügyfél van a bankban, s=2 esetén:

j	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
$\mathbf{P_j}$	0.5	0.3334	0.1111	0.0370	0.0123	0.0041	0.0014	0.0005	0.0002

Hatan állhatnak a sorban, két kiszolgáló van, ezért 8 ügyfél tartózkodhat egyszerre a bankban. Természetes módon a kiszolgálók számának növelésekor a rendszerparaméterek (pl. L, Lq, W, Wq) értékei javulnak.

Irodalomjegyzék

DANYI, P. – VARRÓ, Z. [2001]: Operációkutatás, Lineáris programozás, PTE Közgazdaságtudományi Kar, Pécs

DANYI, P. – VARRÓ, Z. [2000]: Operációkutatás Üzleti döntések megalapozásához, PTE Közgazdaságtudományi Kar, Pécs

DOMSCHKE, W. - DREXL, A. [2002]: Einführung in Operations Research, Springer, Berlin

DOMSCHKE, W. - DREXL, A. – KLEIN, R. – SCHOLL, A. – VOSS, S. [2002]: Übungen und Fallbeispiele zum Operations Research, Springer, Berlin

GÁSPÁR, L. – TEMESI, J. [1989]: Matematikai programozási gyakorlatok, Nemzeti Tankönyvkiadó, Budapest

GÁSPÁR, L. – TEMESI, J. [1989]: Lineáris programozási gyakorlatok, Nemzeti Tankönyvkiadó, Budapest

HILLIER, F. S. – LIEBERMANN, G. J. [1994]: Bevezetés az operációkutatásba (magyar fordítás), LSI Oktatóközpont, Budapest

TEMESI, J. [2002]: A döntéselmélet alapjai, AULA, Budapest

WINSTON, [2003]: Operációkutatás; algoritmusok és alkalmazások (magyar fordítás), AULA, Budapest