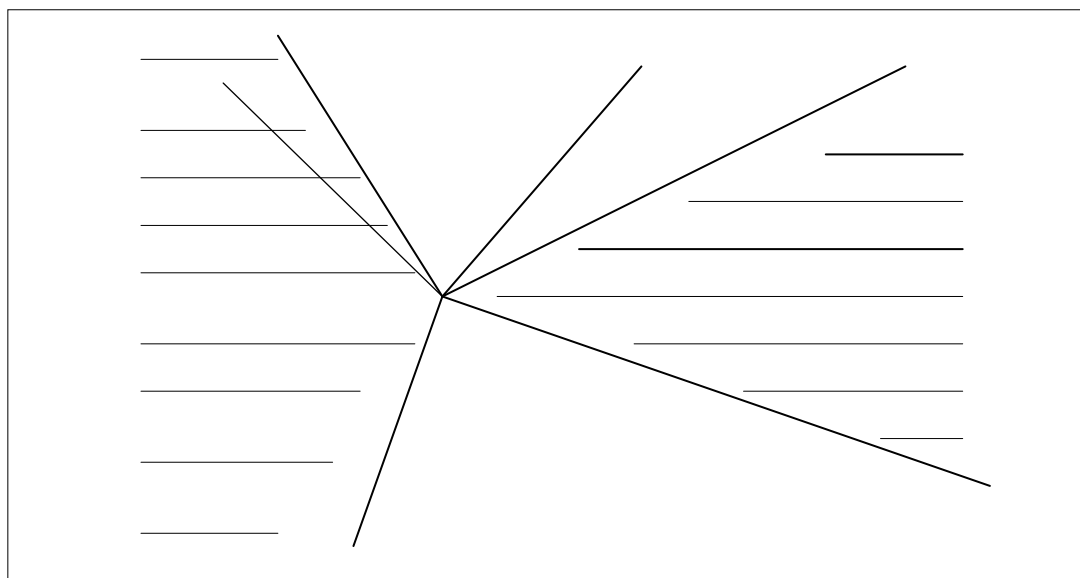


OPERÁCIÓKUTATÁS

No. 9.

Szűcs Gábor

DISZKRÉT SZIMULÁCIÓ MATEMATIKAI ALAPJAI



Szűcs Gábor:

DISZKRÉT SZIMULÁCIÓ MATEMATIKAI ALAPJAI

OPERÁCIÓKUTATÁS No. 9

A sorozatot szerkeszti: Komáromi Éva

Megjelenik a Budapesti Corvinus Egyetem Operációkutatás Tanszéke
gondozásában

Budapest, 2007

Szűcs Gábor:

DISZKRÉT SZIMULÁCIÓ MATEMATIKAI ALAPJAI

Lektorálták: Molnár István, Fiala Tibor, Nadabán János

Készült az Aula Kiadó Digitális Gyorsnyomdájában.

Nyomdavezető: Dobozi Erika

1. Bevezetés

Nagybonyolultságú rendszerekre vonatkozó döntéshozatalnál, melyeknél a döntéseknek igen nagy gazdasági és társadalmi jelentősége van, a döntések hatását nem célszerű azonnal a valóságos rendszereken kipróbálni, mivel a rossz, vagy legalábbis nem optimális döntések jelentős kárt okozhatnak. A különböző lehetséges stratégiák közül az optimális kiválasztását az alkalmazást megelőzően modelleken célszerű megvizsgálni. Ennek korszerű eszköze a számítógépes szimuláció, melynél felépítjük a rendszer modelljét, és a különböző döntések hatását e modell számítógépes szimulációjával vizsgáljuk. Ennek az eszköznek a segítségével kielégítő pontossággal megállapíthatjuk a különböző stratégiák hatását, és kiválaszthatjuk az optimálisat. Ilyen megközelítést a legkülönbözőbb területeken is alkalmazhatunk.

A nagybonyolultságú rendszerek adekvát módú, azaz a valóságos rendszert jól visszatükröző reprezentációját olyan egymással kapcsolatban és kölcsönhatásban levő objektumok hálózatával lehet leírni, melyeknél az egyes objektumoknak paraméterei, működési algoritmusai és kapcsolatai vannak. Ez biztosítja a valóságos világ párhuzamosan bekövetkező eseményeinek megfelelő reprezentációt.

Azoknak a rendszereknek a vizsgálatánál, melyeknél a rendszert leíró hatásmechanizmusok és azok kölcsönhatása nem pontosan ismert; az első feladat a rendszert jól leíró modell meghatározása annak érdekében, hogy a vizsgálatokat lefolytassuk. Ezek közé tartoznak például a mikro- és makroökónómiai folyamatokat reprezentáló modellek. Tovább bonyolítja a helyzetet, amikor a vizsgált folyamatok kiterjednek társadalmi, műszaki, környezeti, infrastrukturális és egyéb tényezők kölcsönhatására is.

A hagyományos matematikai módszerek alkalmazásával ellentétben általában nem egy-egy tényező maximalizálása vagy minimalizálása a cél, hanem a számos tényezőt figyelembe vevő optimum meghatározása. Egy nagy város közlekedésében a gépjárművek okozta környezetszennyezés ugyan megszüntethető lenne a gépjárművek teljes kitiltásával, azonban a gazdasági fejlődést ez az intézkedés jelentősen visszavetné, így nem elfogadható megoldás. A megoldásoknál figyelembe kell tehát venni különböző korlátozásokat.

Jelen mű oktatásra szánt formája követi azt a didaktikai módszert, hogy először a fogalmakat kell pontosan tisztázni, és utána ezek felhasználásával lehet különböző állításokat megfogalmazni, összefüggésekre rámutatni. Az állításokat, tételeket a szöveg kontextusban szervesen beleágyazva találhatjuk meg, a figyelemfelkeltés érdekében azonban egy külön jellel lettek ellátva (ez a § jel például a definíciók, bizonyítások elején is megtalálható). Egyes tételek után a bizonyításokat is közöljük, kevésbé közvetlenül a tárgyhoz tartozó tételeknél azonban csak a tétel kimondására kerül sor. A struktúra kialakításánál a szerző arra törekedett, hogy a szerkezet érthető, tagolt – fejezetek önállóak, de a mondanivaló a fejezetek között átívelő – átlátható és átvihető legyen.

A fejezetekre bontott mű felépítése olyan, hogy először a matematikai alapok kerülnek részletes tárgyalásra a véletlenszám generálástól kezdve a modellezéshez használható matematikai apparátusig. Majd a modellezés munkafázisainak bemutatása következik a modell kialakításától kezdve a futtatáson át a szimulációs eredmények kiértékeléséig. Tovább olvasva az elmélet felől közeledünk a gyakorlat felé, azaz különböző modellezési lehetőségekről, technikákról kapunk információkat, majd a legvégén az elmélet

megkoronázásáról: a szimulációs alkalmazásokról olvashatunk. Ezek az alkalmazási területek: mikro- és makrogazdaságtan, operációkutatás, közúti és vasúti közlekedési rendszerek modellezése, gyártórendszerek szimulációja, környezetvédelem, régiófejlesztési problémák, stb., melyek közül csak néhány bemutatására van lehetőség.

1.1. Szimulációs vizsgálati módszer

Rendszerek vizsgálatánál egy olyan – matematikai módszereknél finomabb – eszközre van szükség, amely nem csak egy modellt tud egyszerre kezelni, hanem annak különböző változatait is. A változatok kipróbálásának modern módszere korunkban a számítógépes szimuláció [7]. Ennek keretében felépítik a valóságos rendszer modelljét, és vizsgálják annak dinamikus működését adott peremfeltételek mellett. A dinamikus szimuláció segítségével meghatározhatók a rendszer működésének jellemzői. Ezt követően a rendszer működését befolyásoló stratégiák változtatásával találhatók meg a legjobb megoldások.

Némely esetben a probléma igen bonyolult, mivel az egyes egymásra ható tényezők egyrészt rendkívül interdiszciplinárisak, másrészt ezek némelyike ill. a köztük levő összefüggések sem egyértelműen ismertek. A modellezés folyamán néha többféle aspektusból kell megközelíteni a problémákat, egyszerre sok célt kell figyelembe venni, ez többszempontú optimalizálási feladatot követel meg. Az ilyen jelleggel bíró tulajdonságot nevezik a szimuláció ‘sok arcúságának’ [54]; az ilyen problémák megoldásánál lehet közelítő modellezést végezni szimuláció segítségével, hogy az összefüggéseket, hatásmechanizmusokat feltárjuk. A nem kielégítően ismert hatásmechanizmusok téves konklúziókhoz vezethetnek, ezért ezek meghatározása a modellek megbízhatóságának szempontjából döntő jelentőségű.

Szimuláció kategorizálása

Diszkrét szimulációról beszélünk, ha mind a szimulációs idő, mind a rendszer állapotai csak diszkrét értékeket vehetnek fel. Ha ezeket a dimenziókat folytonos változókkal kezeljük, akkor **folytonos szimulációról** beszélünk. A folytonos szimulációnak széles spektrumú irodalma van, a folyamatszimulációtól [5], a szabályozáson át [4], differenciál-egyenlet rendszerekig, a jelen opus azonban csak a diszkrét irányvonallal foglalkozik.

Diszkrét szimuláción belül, ha a rendszert leíró változók determinisztikusak, akkor **determinisztikus**, ha a változók minden időpontban egy véletlen eloszlásból származó értéket tartalmaznak, akkor **sztochasztikus** a szimuláció [37][29]. Abban az esetben pedig, ha az idő egy részében determinisztikusan, más részében sztochasztikusan viselkedik a rendszer, akkor **kvázideterminisztikus** szimulációról beszélünk [16].

1.2. Diszkrét esemény rendszerek alapfogalmai

Az alábbiakban a diszkrét esemény rendszerek alapfogalmait ismertetjük [10]:

§ Definíció. Rendszer: A vizsgálandó valóság egy részhalmaza, amely objektumok olyan halmazából áll, melyek interaktív kapcsolatban állnak egymással, és időben meghatározott szabályok szerint kölcsönhatást gyakorolnak egymásra.

§ Definíció. Modell: Absztrakt logikai vagy matematikai reprezentációja a rendszernek, mely leírja az objektumok közti kölcsönhatásokat a rendszerben.

§ **Definíció. Entitás:** A rendszerben illetve a modellben szereplő objektumok, melyek tulajdonságait attribútumoknak nevezzük. Az entitások lehetnek statikusak (helyhez rögzített) és mobilak (helyváltoztató képességgel rendelkezők, melyek a modell különböző helyein is előfordulhatnak).

§ **Definíció. Attribútum:** Az entitások tulajdonságainak leírója. Ez a leíró rendelkezik egy névvel vagy jellel az azonosítás miatt, és értékkel, azaz minden attribútummal rendelkező entitás értéket kap.

§ **Definíció. Rendszer-, modell- vagy állapotváltozó:** Nem csak az entitások rendelkezhetnek tulajdonságokkal (attribútumokkal), hanem az egész modellre (illetve rendszerre) vonatkozóan is be lehet vezetni változókat, melyek ezeket a globális tulajdonságokat képviselik. Ugyanúgy, mint az attribútumok: névvel és értékkel rendelkeznek, de az attribútummal ellentétben csak egy érték lehet egyszerre érvényben (entitások attribútumainál minden entitás kaphat más-más értéket).

§ **Definíció. Esemény:** Olyan jelenség, amely a modell állapotát megváltoztatja (változhat egy vagy több állapotváltozó értéke is), más néven tehát állapotváltozás.

§ **Definíció. Kísérlet:** Kísérleten a rendszer vagy modellje viselkedésének megfigyelését értjük adott feltételhalmaz mellett. Kísérlet a számítógépes modellezés esetén megfelel a futtatásnak (szimulációs futtatás).

§ **Definíció. Szimulációs futtatás:** Szimulációs futtatáson az elkészült modellen való kísérletezést és viselkedésének megfigyelését értjük adott szimulációs feltételrendszer mellett.

§ **Definíció. DEVS formalizmus:** DEVS (Discrete Event System Specification) [54] formalizmus egy struktúrát jelöl, mely a következő négy elemből áll: $M = \{X, S, \delta, ta\}$, ahol

- X a külső események halmaza.
- S az állapotok szekvenciális halmaza.
- δ az átmeneteket leíró függvény. Ez két részből áll: belső és külső átmeneti függvényből.
- ta az idő függvény; $ta(s)$ jelenti azt az időintervallumot, amit a rendszer az s állapotban tölthet, ha nincs külső esemény.

A szimulációs alkalmazások többsége tartalmaz véletlenszerű elemeket is, azaz vagy teljesen sztochasztikus vagy kvázideterminisztikus modellek felépítése a feladat (nagyon ritkán fordul elő teljesen determinisztikus eset). A „véletlen”-t viszont modellezni kell, ezért a szimuláció tudományában a véletlenszámok (véletlenszám sorozatok) kulcs fontosságú szerepet töltenek be, melyek a nem determinisztikus jellegű folyamatok számára biztosítják a sztochasztikusságot.

2. Álvéletlenszám generálás

§ **Definíció. Kongruencia:** egy olyan osztályokba sorolási számelméleti fogalom, miszerint két szám azonos osztályba kerül, ha egy harmadik számmal való osztás során ugyanazt a maradékot adják.

§ **Definíció. Véletlenszerűség:** Egy számsorozatot (Kolmogorov, Chaitin, Solomonoff javaslatára) véletlennek tekintünk, ha a legrövidebb algoritmus, melynek segítségével leírhatjuk, közel azonos mennyiségű információt tartalmaz, mint maga a számsorozat. Azaz a sorozat információtartalma komprimálhatatlan.

A „véletlen” modellezését elvileg két teljesen különböző irányból oldhatjuk meg, a gyakorlatban azonban csak az egyiknek van létjogosultsága: a fizikai és a matematikai modellezés közül az utóbbinak. A **fizikai** modellezés esetén valamilyen természetben lejátszódó véletlenszerű folyamatot vesznek alapul, melynek a sztochasztikus tulajdonságát kihasználva és áttanszformálva hoznak létre egymás után véletlen számokat. Ezek tehát valódi véletlenszámok, de a fizikai, kémiai folyamatok (mint például radioaktív bomlás, dióda zaj, diffúzió, stb.) mérését és értékeinek áttanszformálását szolgáló berendezés megalkotása és használata olyan gyakorlati nehézségeket gördít az eredeti cél elé, hogy ezt a fajta megoldást nem használják.

Marad tehát a másik, a **matematikai** algoritmusok segítségével történő előállítás, ahol a számsorozatok a megközelítés jellegénél fogva nem lehetnek valódi véletlenszámok, így ezeket **álvéletlenszámoknak** (pseudovéletlenszámoknak) hívják. Mivel fizikai modellezéssel a továbbiakban nem foglalkozunk, csak a matematikailag előálló álvéletlenszámokkal, így ezeket néha jelző nélkül csak véletlenszámoknak fogjuk nevezni álvéletlenszámmot értve alatta.

2.1. Monte Carlo módszer

A **Monte Carlo módszer** egy alkalmazott numerikus eljárás, amely egy sztochasztikus modell előállítását célozza meg véletlenszámok generálásán alapulva. A módszer lényege, hogy valamilyen véletlen kísérlettel kapcsolatos valószínűségben vagy várható értékben fellelő egy ismeretlen mennyiség. Ekkor az ismeretlen mennyiséget közelíthetjük úgy, hogy a kísérletet sokszor elvégezzük, illetve véletlen számsorozatok segítségével szimuláljuk.

A Monte Carlo elnevezést 1949-ben kapta egy publikációban (N. Metropolis és S. Ulam egyik cikkében) arra utalva, hogy a véletlenszám sorozatokként a játékkaszinókban használatos szerencsejátékok (például rulett) kimenetelei is jól használhatóak, a szerencsejátékok birodalma pedig nem más, mint: Monte Carlo [28].

A Monte Carlo módszereket olyan problémáknál lehet jól használni, ahol sztochasztikus folyamatokból (valószínűség-számításra alapozva) épül fel a feladat. Vannak azonban olyan problémák, amelyek semmiféle kapcsolatban nem állnak a valószínűség fogalmával, de a feladat számolásigénye olyan nagy, hogy Monte Carlo szimulációt érdemes bevetni [39]. Ebben az esetben a probléma analitikus megfogalmazásából indulunk ki, ezután ehhez keresünk megfelelő sztochasztikus modellt, majd megfigyeléseket végzünk ezzel a modellel

kapcsolatban, és végül különböző statisztikákkal megbecsüljük az eredeti feladatban szereplő paramétereket.

A Monte Carlo módszereket a matematikai statisztikában is használják, amikor egy ismeretlen eloszlásfüggvényt úgy határoznak meg, hogy nagy elemszámú mintából közelítik, approximálják azt [28]. Továbbá a statisztikát alkalmazó gazdasági vizsgálatoknál van nagy szerepe, sőt régóta használják már matematikában integrálszámításnál és a statisztikus fizikában is [6].

A Monte Carlo módszernél tehát nagy jelentősége van a véletlenszámok használatának, így a továbbiakban néhány álvéletlenszám előállítás metódust ismertetünk. Álvéletlenszámokra már a számítástechnika őskorában is szükség volt, így az egyik korai előállítási mód Neumann János nevéhez fűződik. Ő találta ki a négyzetközép és a szorzatközép módszereket [30]. További generálási módszerek közé tartozik például a Lehmer-féle multiplikatív kongruencia módszer, az Elfogadás-visszautasítás módszere és az Inverz transzformációs módszer.

2.2. *Lehmer-féle multiplikatív kongruencia módszer*

A módszer a kongruencia (azaz a maradékképzés) fogalmán alapul, azaz A és B akkor és csak akkor kongruensek modulo m szerint (ahol m egy egész szám) ha létezik olyan k egész, amelyre $A - B = k \cdot m$. Ekkor azt, hogy A és B kongruensek m szerint, a következőképpen jelöljük: $A \equiv B \pmod{m}$. A kongruenciarelációban m -et a kongruencia modulusának nevezzük.

Ha A és B kongruensek m szerint, akkor mindig található olyan C egész szám ($0 \leq C < m$), amelyre:

$$A \equiv C \pmod{m} \quad (1)$$

azaz A / m osztást elvégezve kapunk egy k értékét, és a maradék C lesz. A kongruenciareláció az egész számok halmazát egymást kölcsönösen kizáró ún. *maradékosztályokba* osztja.

A Lehmer-féle multiplikatív kongruenciamódszer a következő rekurzív formula szerint képzí az álvéletlenszámok sorozatát:

$$u_{n+1} = x \cdot u_n \pmod{m} \quad (2)$$

A rekurzív formulából levezethető a direkt formula:

$$u_n = x^n \cdot u_0 \pmod{m} \quad (3)$$

A kezdeti értékek (x, u_0, m) megadása lényegesen befolyásolja a sorozat periódushosszát, ezért igen fontos ezeknek a helyes megválasztása [16]. Kezdetben vonzónak tűnt m -et prímszámmak választani, x -et pedig m primitív gyökének, ami $m-1$ szám generálását biztosítja. Gyakorlatban azonban sokszor célszerűbb a számítógép adottságaihoz alkalmazkodni és m -et a számítógép szóhosszának választani (ami b bit szóhosszú bináris gépnél $m=2^b$ érték), ami gyorsítja a végrehajtást, mert az osztás shifteléssel (bináris számrendszerben felírva egy számot csúsztatni lehet a számjegyeket a helyiértékek között) végrehajtható, x -re pedig olyan

értéket adni, hogy megfelelően nagy periódusú álvéletlensorozatot kapjunk. Érdemes az x -et és m -et egymáshoz képest relatív prímnek megválasztani, és ugyanez érvényes az u_0 és m értékeire is, azaz a legnagyobb közös osztó ebben a két esetben 1 kell, hogy legyen:

$$(x, m) = 1 \quad (4)$$

$$(u_0, m) = 1 \quad (5)$$

2.3. Elfogadás-visszautasítás módszere (EVM)

Ez a módszer a sűrűségfüggvény segítségével állít elő véletlenszámokat, így nincs szükség az eloszlásfüggvényre (sem analitikus, sem közelítő formában), az elfogadás-visszautasítás módszerét viszont csak véges intervallumon lehet használni [53]. Az eljáráshoz szükség van a sűrűségfüggvényen kívül egy egyenletes eloszlású véletlenszám generátorra. A módszer lényege a következő: ha az eloszlás sűrűségfüggvénye $f(x)$ az $[a,b]$ véges intervallumon értelmezett és ismert, akkor 5 lépéses algoritmus segítségével generálhatók a véletlenszámok.

§ Algoritmus. Elfogadás-visszautasítás módszere:

1. Jelölje M az $f(x)$ maximális értékét az $a \leq x \leq b$ intervallumon.
2. Generáltassunk két egyenletes eloszlású véletlenszámot a $[0,1)$ intervallumon belül: r_1, r_2 .
3. Vezessünk be egy új változót (x^*), melyre $x^* = a + (b-a) r_1$, ami azt jelenti, hogy az x^* véletlen eloszlású lesz az $[a,b]$ intervallumon.
4. Határozzuk meg a sűrűségfüggvény értékét az x^* pontban, azaz $f(x^*)$ -ot.
5. Ha

$$r_2 \leq \frac{f(x^*)}{M} \quad (6)$$

akkor x^* -ot elfogadjuk, mint az $f(x)$ sűrűségfüggvényű véletlenszámot. Ellenkező esetben visszatérünk a 2. ponthoz és újra generálva 2 számot folytatjuk az iteratív eljárást mindaddig, amíg az 5. pontban a feltétel végre teljesül.

§ Állítás

Az EVM (Elfogadás-visszautasítás módszere) algoritmus $f(x)$ sűrűségfüggvényű eloszlást generál. Ennek a belátásához azt kell igazolni, hogy az EVM által generált véletlenszámok eloszlására (ezt az eloszlást jelöljük z eloszlásnak) igaz a következő egyenlet:

$$p(x \leq z \leq x + \Delta x) = f(x)\Delta x \quad (7)$$

§ Bizonyítás

Annak a valószínűsége, hogy a leggenerált r_1 alapján számolt x^* egy adott x -nek a környezetébe esik (pontosabban x és $x+\Delta x$ közé):

$$p(x \leq x^* \leq x + \Delta x) = \frac{\Delta x}{b-a} \quad (8)$$

ugyanis az x^* értékek teljesen egyenletesen helyezkednek el az $[a,b]$ intervallumon belül, így csak a vizsgálandó Δx szakasz és a teljes intervallum arányától függ ez a valószínűség. Az így kapott valószínűséget még meg kell szorozni annak a valószínűségével, hogy ezt az értéket nem utasítjuk vissza (azaz elfogadjuk). Az elfogadás feltétele:

$$r_2 \leq \frac{f(x)}{M} \quad (9)$$

Ennek a valószínűsége pedig:

$$p\left(r_2 \leq \frac{f(x)}{M}\right) = \frac{f(x)}{M} \quad (10)$$

mivel a véletlenszám képzés előtt rögzítettnek tekinthető $f(x)/M$ érték egy 0 és 1 közé eső szám, és annak a valószínűsége, hogy ennél a számnál kisebb lesz az r_2 véletlenszám: csak ettől az értéktől függ. Így annak az összetett eseménynek a valószínűsége, hogy egyszeri próbálkozás esetén az x^* az x és $x+\Delta x$ közé esik, és ezt el is fogadjuk a következő:

$$\frac{\Delta x \cdot f(x)}{(b-a) \cdot M} \quad (11)$$

Az elfogadást, mint összetett eseményt külön is vizsgálhatjuk. Egy próbálkozás esetén akkor fogjuk elfogadni a generálás eredményét, ha bármilyen Δx nagyságú intervallumba is esett, a (9)-es feltétel fenn áll. Azaz összegezni kell az elemi események valószínűségeit minden Δx kis szakaszra. Tehát a (11)-es egyenletet kell összegezni, azaz integrálni x szerint:

$$p(\text{elfogadas}) = \int_{x=a}^b \frac{dx \cdot f(x)}{(b-a) \cdot M} \quad (12)$$

$$p(\text{elfogadas}) = \frac{1}{(b-a) \cdot M} \int_{x=a}^b f(x) \cdot dx \quad (13)$$

Most használjuk ki azt a tényt, hogy az $f(x)$ sűrűségfüggvény véges intervallumon belül van csak értelmezve. Mivel ez az intervallum az $[a,b]$, ezért az $[a,b]$ intervallumon kívüli integrálás eredménye 0, azaz az $[a,b]$ intervallum integrál értéke megegyezik a mínusz végtelentől a plusz végtelenig terjedő integrál értékével, vagyis 1-el. (Az eloszlásfüggvény plusz végtelenben vett értéke minden eloszlás esetén: 1).

$$p(\text{elfogadas}) = \frac{1}{(b-a) \cdot M} \quad (14)$$

Ennek megfelelően egy próbálkozás (1 iteráció) esetén a visszautasítás valószínűsége:

$$p(\text{visszautasitas}) = 1 - \frac{1}{(b-a) \cdot M} \quad (15)$$

Nézzük meg, hogy mi annak a valószínűsége, hogy az eljárás végén pont egy x és $x+\Delta x$ közötti értéket kapunk véletlenszámként. Ez úgy jöhet ki, hogy az i -edik iterációra kapjuk ezt az elfogadott számot és előtte $(i-1)$ -szer visszautasítottuk az értékeket, vagyis ennek a sorozatnak kell kiszámítani a valószínűségét, és összegeznünk kell őket minden i -re:

$$\sum_{i=1}^{\infty} \{p(\text{visszautasitas})^{i-1} \cdot p(x \leq x^* \leq x + \Delta x) \cdot p(\text{elfogadas})\} \quad (16)$$

Az utóbbi két tényező szorzatát már a (11) egyenletben kiszámítottuk, azaz lehetett volna egyből az elfogadott és Δx szakaszon belüli érték valószínűségét felhasználni.

$$\sum_{i=1}^{\infty} \left(p(\text{visszautasítás})^{i-1} \cdot \frac{\Delta x \cdot f(x)}{(b-a) \cdot M} \right) = \quad (17)$$

$$= \sum_{i=1}^{\infty} \left(\left(1 - \frac{1}{(b-a) \cdot M} \right)^{i-1} \cdot \frac{\Delta x \cdot f(x)}{(b-a) \cdot M} \right) = \quad (18)$$

$$= \frac{\Delta x \cdot f(x)}{(b-a) \cdot M} \sum_{i=1}^{\infty} \left(1 - \frac{1}{d} \right)^{i-1} \quad \text{ahol } d = (b-a) \cdot M \quad (19)$$

A levezetés ezután 2 ágra bomlik, attól függően, hogy d értéke 1 vagy nagyobb, mint 1 (kisebb azért nem lehet, mert az $f(x)$ maximuma M , az a -tól b -ig integrált $f(x)$ értéke 1, és így a $b-a$, M oldalak által meghatározott téglalap területe ennél nagyobb, vagy egyenlő lehet csak).

a) $d=1$

Ez akkor fordulhat elő, ha az $f(x)$ egyenletes eloszlású az $[a,b]$ intervallumon.

Ebben az esetben a visszautasítás valószínűsége nulla ugyan, így sohasem lesz visszautasítva a generálás, azaz i mindig 1 lesz. Az x és $x+\Delta x$ közötti érték mindig elsőre el lesz fogadva, azaz a (16) képletben nem kell a szummázást elvégezni ($d=1$ -et felhasználva), így a keresett valószínűség:

$$\frac{\Delta x \cdot f(x)}{(b-a) \cdot M} = \Delta x \cdot f(x) \quad (20)$$

b) $d>1$

Ha az $f(x)$ bármilyen (egyenletes eloszláson kívül) eloszlású az $[a,b]$ intervallumon.

Így a (19)-as egyenlet tovább írható:

$$\frac{\Delta x \cdot f(x)}{(b-a) \cdot M} \sum_{i=1}^{\infty} c^{i-1} \quad \text{ahol } c = 1-(1/d) \text{ és } 0 < c < 1 \quad (21)$$

Felhasználva a mértani sor összegképletét:

$$\frac{\Delta x \cdot f(x)}{(b-a) \cdot M} \frac{1}{1-c} = \frac{\Delta x \cdot f(x)}{(b-a) \cdot M} \cdot d = \frac{\Delta x \cdot f(x)}{(b-a) \cdot M} (b-a)M \quad (22)$$

Így a keresett valószínűség:

$$\Delta x \cdot f(x) \quad (23)$$

Tehát mindkét (a és b) esetben ezt az $f(x) \cdot \Delta x$ értéket kaptuk. ■

2.4. Tetszőleges eloszlású álvéletlenszámok generálása

Tetszőleges eloszlású álvéletlenszám sorozatokat egyenletes eloszlású álvéletlenszámokból állíthatunk elő. Ehhez megadunk egy előállítási módszert is, de előbb nézzük meg Glivenko tételét, melynek ismerete fontos a módszer megértéséhez.

§ Tétel. Glivenko tétele

A minták számának (n) növelésével az $F_n(x)$ empirikus eloszlásfüggvény az egész számegyenesen egyenletesen konvergál az $F(x)$ elméleti eloszlásfüggvényhez, azaz

$$h_n = \sup_{-\infty < x < \infty} |F_n(x) - F(x)| \text{ jelöléssel élve: } P(\lim_{n \rightarrow \infty} h_n = 0) = 1 \quad (24)$$

ahol h_n a legnagyobb eltérés az empirikus és az elméleti eloszlásfüggvény között. A Glivenko tétel jelentősége miatt ezt a **matematikai statisztika alaptételének** is szokták nevezni.

Inverz transzformációs módszer

Tetszőleges eloszlású álvéletlenszámok generálása Inverz transzformációs módszerrel történik [38]. A módszerhez szükség van a kívánt eloszlás eloszlásfüggvényére (analitikus formában, vagy ha nem áll rendelkezésre ilyen, akkor: grafikus formában). Jelöljük ezt az ismert eloszlásfüggvényt $F(x)$ -el. Ezen kívül szükség van egy $[0,1]$ intervallumban egyenletes eloszlású álvéletlenszám generátorra. A módszer lényege, hogy az egyenletes eloszlású generátorral generálunk egy 0 és 1 közötti számot (jelöljük ezt u' -vel), majd ebből kiszámoljuk az x' -t: $x' = F^{-1}(u')$ képlet segítségével, ahol F^{-1} az $F(x)$ függvény inverzét jelöli (ha analitikusan nem áll rendelkezésre az a képlet, akkor grafikusan is megszerkeszthető). Az empirikus és az elméleti eloszlásfüggvény közötti kapcsolatot a Glivenko tétele fejezi ki.

§ **Állítás:** Inverz transzformációs módszerrel generálva az x' számokat, az x' -re a kívánt eloszlású álvéletlen-számokat fogjuk kapni ($F(x)$ eloszlásfüggvénnyel).

§ Bizonyítás:

Az állítás belátásához azt kell igazolni, hogy ha rögzítünk két számot: x_1 és x_2 , akkor annak a valószínűsége, hogy az Inverz transzformációs módszerrel kapott x értékek e két szám közé esnek, megegyezik az eloszlás függvényből vett értékek különbségével, azaz:

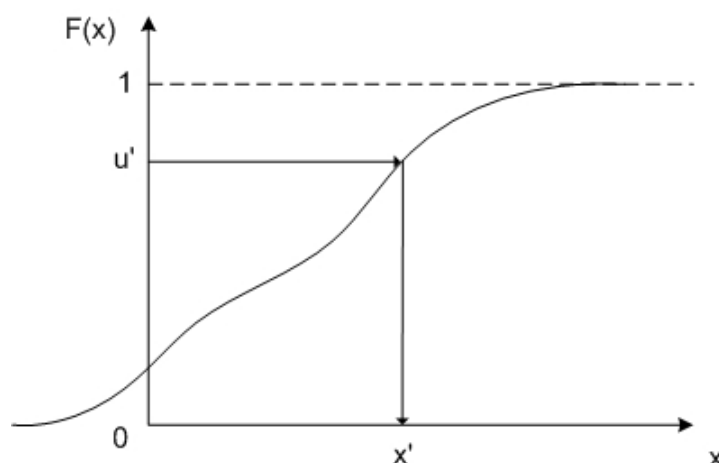
$$p(x_1 \leq x \leq x_2) = F(x_2) - F(x_1) \quad (25)$$

A baloldalt (azaz a generálási módszerrel kapott valószínűséget) úgy kapjuk meg, hogy megnézzük: mi annak a valószínűsége, hogy az u egyenletes eloszlású szám $u_1 = F(x_1)$ és $u_2 = F(x_2)$ érték közé esik. Ha ez a valószínűség megegyezik az egyenlet jobb oldalával, akkor sikerült az állítást belátni. Ehhez elég lenne bizonyítani a következőt:

$$p(x \leq x') = p(u \leq u') \quad (26)$$

Ugyanis x' -t és u' -t behelyettesítve egyszer az x_1 és u_1 másodszor az x_2 és u_2 helyére megkapjuk a valószínűségek egyenlőségét, amiből már következik a valószínűségek különbségének egyenlősége.

Tekintsük az 1. ábraát, ebben látható a generálási módszer lényege. Ha a $[0,1]$ intervallumban (u -val jelölt) egyenletes eloszlásból egy u' értéket generálunk, akkor azt az $F(x)$ függvénynek megfelelően vetíthetjük le az x tengelyre. Mivel u egyenletes eloszlású (és mind az értelmezési tartomány, mind az értékkészlet is 0 és 1 között van), ezért:



1. ábra Egy tetszőleges eloszlás

$$u = G(u) \quad (27)$$

ahol $G(u)$ az u valószínűségi változó eloszlásfüggvénye. Másrészt a bemutatott generálási módszer miatt:

$$x' = F^{-1}(u') \quad \text{vagy másképpen:} \quad u' = F(x') \quad (28)$$

Az eloszlásfüggvény (mely mindig egy monoton növekvő függvény) definíciója miatt az x eloszlásnál:

$$p(x \leq x') = F(x') \quad (29)$$

Felhasználva az előző egyenleteket:

$$p(x \leq x') = F(x') = u' = G(u') = p(u \leq u') \quad (30)$$

Azaz az állítást beláttuk. ■

2.5. Álvéletlenszámok jóságának tesztelése

A fenti álvéletlenszám generálásokon kívül is nagyon sok algoritmus [23][24] létezik még, ezeket akár rekurzívan módon [49] egymásba is ágyazhatjuk. Felmerül a kérdés, hogy ezeknél milyen szempontok szerint tudjuk értékelni a véletlenséget, összehasonlítani egy előre megadott jósággal és összevetni egymással? Az álvéletlenszám sorozatok lényege a kiszámíthatatlanság, így a sorozatot létrehozó generátor tesztelésénél két fogalmat kell megvizsgálni: igazi véletlenszámoktól való megkülönböztethetőséget és a megjósolhatóságot.

Akkor mondjuk, hogy a generátor az igazitól **megkülönböztethetetlen**, ha nincs olyan algoritmus, mely polinomiális időben az esetek több, mint 50%-ában helyesen tippelné meg, hogy melyik gép az, amelyik a valódi véletlen sorozatot adja.

A másik fogalom bemutatásához nézzük meg a generált sorozatot számról számra, de mielőtt egy-egy újabb jegyét megnéznénk, megpróbáljuk megtippelni, hogy mi lesz a következő. Az

előzőekhez hasonlóan a jósláshoz is csak polinomiális időt használhatunk, és annyit kell elérni, hogy az esetek több, mint felében sikeresen tippeljünk. Ha nincs ilyen algoritmus, akkor azt mondjuk, hogy a generátor **megjósolhatatlan**. Ez a két definíció ekvivalens: ha egy álvéletlen sorozat az igazítól megkülönböztethetetlen, akkor megjósolhatatlan, és viszont.

Álvéletlenszámok hátránya és előnye

Álvéletlenszámok hátránya a véletlenszámokhoz képest, hogy nehéz egy minden szempontból jó algoritmust megalkotni. Előnye viszont, hogy ugyanolyan kezdőbeállításokkal elindított generátor által alkotott sorozat többször is megismételhető (azaz ugyanolyan sorrendben következnek a számok), ami a tesztelésben (szimulációs modell és program tesztnél) hatalmas segítséget jelent, hisz a nem reprodukálható hibákat szinte lehetetlen lenne megtalálni a tesztelések során.

Álvéletlenszám generátorok tesztelése

Álvéletlenszámok jóságának tesztelésére a következő vizsgálatokat szokták elvégezni (a teljesség igénye nélkül) [42]:

- Egyenletes eloszlás vizsgálat
- Függelenség vizsgálat
- Permutációs teszt
- Maximum teszt

Az egyenletes eloszlás tesztelésénél az eloszlás teljes intervallumát felosztjuk azonos nagyságú szakaszokra, majd megnézzük, hogy a generált számok mely kis intervallum darabba esnek. Megszámoljuk az egyes szakaszokra eső generált értékeket, és ebből gyakoriságdigramot készítünk. A hisztogramnak egyenletes eloszlást kell mutatnia.

Függelenség vizsgálatot végezhetünk többféleképpen is. Megnézhetjük, hogy az egymást követő szám-párok függetlenek-e egymástól: Ha a két értéket 2 dimenziós koordináta rendszerben ábrázoljuk úgy, hogy a pár első tagját x , második tagját y koordinátának fogjuk fel, akkor az így kapott pontok egyenletesen kell, hogy a síkot beterítsék (bármiféle csomósodás valamilyen korrelációra utal). Ez kibővíthető 3 dimenzióra (ill. n dimenzióra) az egymást követő számhármak (szám n -esek) által.

Diszkrét esetben az egymást követő szám n -esek esetén nem csak az a követelmény, hogy egyenletesen töltsék be az n dimenziós teret, hanem az is, hogy ne legyenek lyukak, azaz a szám n -es minden permutációja egyforma valószínűségű legyen. Ezt hívják permutációs tesztnek.

Meg lehet vizsgálni az n számból álló csoportok legnagyobb elemét, majd a maximumok eloszlását tesztelni lehet, hogy teljesítik-e azt a követelményt, hogy nem megkülönböztethető az egyenletes eloszlásból származtatott eloszlástól. Ezen kívül is jó néhány tesztelési lehetőség van még [26], itt a teljesség igénye nélkül soroltuk fel a legfontosabb vizsgálatokat.

Az álvéletlenszámok generálásának módszerei és jóságuk tesztelése után a következő fejezetekben a szimulációs alkalmazás egyes munkafázisait fogjuk megvizsgálni, melyek a következők:

- Matematikai előkészítés, modellezési előkészületek
- Szimulációs modell felépítése
- Szimulációs modell futtatása
- Szimulációs eredmények kiértékelése

3. Matematikai és modellezési előkészítés

3.1. Matematikai apparátus

Ahhoz, hogy a számítógépes modellezést elő lehessen készíteni, szükség van néhány fontos statisztikai tételre, törvényre [20][21]. Ezek közül a nagy számú adatot tartalmazó adathalmazra építő tételek fontosak számunkra:

§ Tétel. Központi (centrális) határeloszlás-tétel

Ha X_1, X_2, \dots, X_n független, azonos eloszlású valószínűségi változók, várható értékük megegyezik $E(X_i) = \mu$, szórásuk is egyenlő és véges: $D(X_i) = \sigma$, ahol $i=1, 2, \dots, n$; akkor:

$$\lim_{n \rightarrow \infty} P\left(\frac{X_1 + X_2 + \dots + X_n - n \cdot \mu}{\sigma \sqrt{n}} < x\right) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^x e^{-\frac{u^2}{2}} du \quad (31)$$

vagyis az adatszám növelésével a valószínűségi változók átlaga a normális eloszláshoz tart.

§ Tétel. Nagy számok gyenge törvénye

Ha X_1, X_2, \dots, X_n független, azonos eloszlású, μ várható értékű valószínűségi változók, akkor bármely kis $\varepsilon > 0$ -ra:

$$P\left(\left|\frac{X_1 + X_2 + \dots + X_n}{n} - \mu\right| > \varepsilon\right) \rightarrow 0 \quad (n \rightarrow \infty) \quad (32)$$

a valószínűségi változók átlaga tetszőlegesen meg tudja közelíteni az elméleti várható értéket [38].

§ Tétel. Nagy számok erős törvénye

Ha X_1, X_2, \dots, X_n független, azonos eloszlású, μ várható értékű valószínűségi változók, akkor 1 valószínűséggel tart a valószínűségi változók átlaga az elméleti várható értékhez [25]:

$$P\left(\lim_{n \rightarrow \infty} \frac{X_1 + X_2 + \dots + X_n}{n} = \mu\right) = 1 \quad (33)$$

3.2. Időhorizont vizsgálata

§ Definíció. Eseményjelző: Egy olyan címke (rekord), mely egy esemény bekövetkezését jelzi. Az eseményeket ugyanis a szimulációs rendszernek fel kell dolgozni, végre kell tudnia hajtani, és az eseményjelző utal arra, hogy milyen eseményt és mikor kell majd végrehajtani (ha egyéb körülmények ezt nem gátolják meg).

§ Definíció. Láncolt lista: A láncolt lista olyan rekordok egymás utáni sorozata, ahol a rekordok azonos dolgokat képviselnek (például entitásokat vagy eseményjelzőket, stb.) és a rekordok sorrendje meghatározott.

§ Definíció. Jósolt esemény lista: (FES - Future Event Set) Olyan láncolt lista, melyben a sorozat elemeit az eseményjelzők alkotják.

§ Definíció. Idővezérlési eljárás: (Timing routine) olyan eljárás, amely karbantartja a jósolt esemény listát a szimuláció futása alatt.

§ Definíció. Trajektória minta: Trajektória mintának (Sample Path) nevezzük a vizsgált változók szimulációs futtatás alatti értékeinek sorozatát egy adott időintervallumon belül: $\{X_i, \text{ ahol } t_1 \leq i \leq t_2\}$, $\{Y_i, \text{ ahol } t_1 \leq i \leq t_2\}$, stb.

Események az időhorizonton

A matematikai apparátus sokat segít az adatok előkészítésében, ismert eloszlások felhasználásával akár szimuláció nélkül is jól modellezhetők az egyszerűbb rendszerek. Azonban a bonyolultabb valós rendszereknél az események sok esetben nehezen modellezhetők egy ismert eloszlással. A legtöbb valóságos eloszlás empirikus, és sokszor megfigyelhetők bizonyos sűrűsödések (más néven csomósodások, az angol szakirodalom pedig a 'burst' terminust használja) az események idő-dimenziójában. Azaz bizonyos időintervallumban ritkán jönnek az események, más időszakaszban pedig sűrűn követik egymást. Az ok a rendszer hatásmechanizmusában keresendő, ahol ha egy bizonyos esemény bekövetkezik, akkor nagy valószínűséggel egy másik eseményt vált ki rövid időn belül, ami szintén újabb eseményeket generál, és így lavinaszerűen megnő az események gyakorisága. Az ilyen intervallumokat csúcsintervallumnak nevezzük.

Időléptetési eljárások - idővezérlés

A szimulációs modellek dinamikus működésének vezérlését az időléptetési algoritmusok végzik, melyek közül kettőről teszünk említést: next event és a time mapping időléptetés.

A **next event** időléptetési eljárás a következő jósolt esemény idejére ugrik (azaz a rendszer idejébe a jósolt esemény idejét írja be), így a szimulált időlépések maximálisak, azonban a kezelendő listák (egy időpontban sorakozók listája) hosszúak.

A **time mapping** időléptetés esetében a minimális időinkrementeknek megfelelő lépések árán érhető el a rendkívül hosszú listák kezelésének elkerülése. Így láthatjuk, hogy a next event és time mapping időléptetési eljárások előnyei ill. hátrányai egymással ellentétes jellegűek [16].

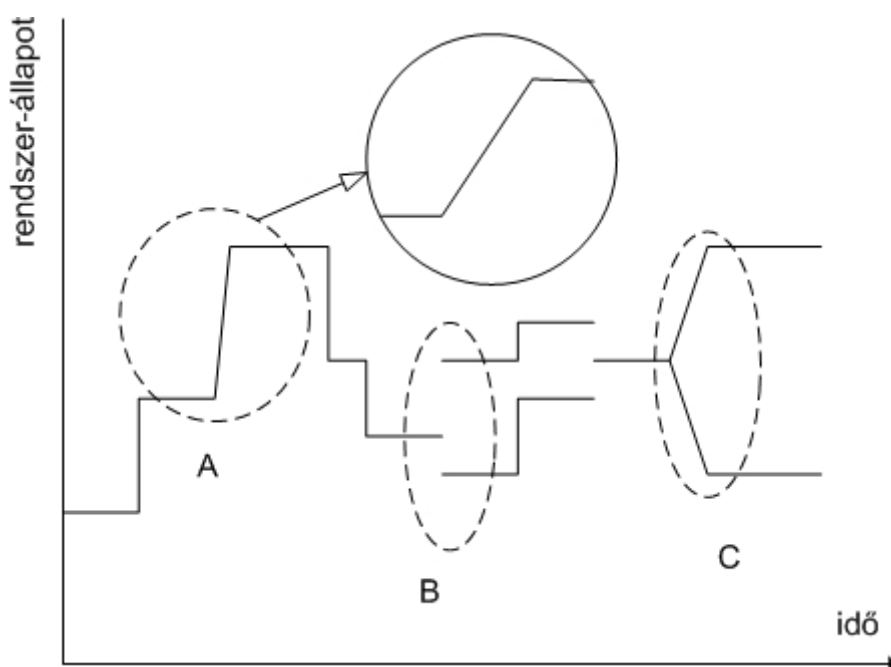
Indetermináltság

A rendszer állapotait vizsgálva az idő függvényében két jelenségre lehetünk figyelmesek. Diszkrét szimuláció esetében a szimulált rendszer diszkrét időpontokban van értelmezve, így ezekben az időpontokban megfigyelve a rendszert azt láthatjuk, hogy valamilyen diszkrét állapotban van. Azonban az események is csak diszkrét időpontokban történnek, így egy állapotváltozás elvileg nulla időhosszúságig tart. Azonban, ha bizonyos állapotváltozást nagyító alá vesszünk, akkor azt láthatjuk, hogy bármilyen rövid ideig is tart a változás lefolyása, az ehhez szükséges időintervallum nagyobb, mint nulla. Ennek az időintervallumnak az elején a rendszer az egyik, a végén pedig egy másik konkrét állapotban található; a kettő között azonban nem tudjuk értelmezni, hogy melyik állapotban van. Ezt a

fajta bizonytalanságot nevezünk elsőrendű indetermináltságnak (a rendszer ebben az indeterminált állapotban van), röviden ID1-nek.

A másik fajta jelenség a másodrendű indetermináltság (ID2). Ebben az esetben nem az átmenetből adódó bizonytalanságot értjük a jelenség alatt, hanem az állapotváltozás végének a bizonytalanságát, azaz hogy melyik állapotba érkezik meg a rendszer. Vagyis, ha tudjuk, hogy mely állapothalmazban lehet a rendszer, de nem tudjuk, hogy ezek közül melyik állapotot veszi fel éppen, akkor a rendszer másodrendű indeterminált állapotban van.

Az elsőrendű indetermináltság tehát a determinisztikus állapotok közötti átmenetből, másodrendű indetermináltság pedig a determinisztikus állapotok megvalósulásának bizonytalanságából adódik. Természetesen előfordulhat egyszerre mindkét fajta indeterminált állapot is, amikor az állapotváltozás folyamata hosszabb, mint nulla és így ID1 lép fel, az átmenet végén pedig nem tudjuk melyik diszkrét állapotba kerül a rendszer (ID2). A következő ábrán mindhárom esetre látunk példát, az *A* jelenségnél az ID1, *B*-nél az ID2, *C*-nél pedig mindkettő előfordul.



2. ábra Első- és másodrendű indetermináltság

3.3. Esemény, aktivitás és folyamat leírási módok

A modellezés elkezdése előtt érdemes a modellezni kívánt rendszert alaposan megvizsgálni, ugyanis különböző típusú rendszerek különböző megközelítési módot igényelnek. A megközelítési módszerek közül három leírási módot ismertetünk: esemény leírási, aktivitás leírási és folyamat leírási módot.

Esemény leírási mód

Az esemény leírási módnál a hangsúly az eseményeken van, melyek megváltoztatják a diszkrét rendszer egy vagy több állapotváltozójának az értékét. Az események hirtelen történnek, tehát zérus szimuláció idő alatt mennek végbe. Ezeket az ún. eseményosztályokba

sorolhatjuk. Egy eseményosztályba tartozó események által létrehozott állapotváltozók azonos módon írhatók le; ezeket a leírásokat eseményleírásnak nevezzük.

Egy eseménynek legalább két jellemzője van:

- az eseményidő, amikor megtörténik;
- az eseményosztály-jellemző, ami a megfelelő eseményosztályt határozza meg, amelybe tartozik.

Az események jellemzőit tartalmazó adatstruktúrát **eseményjelző**nek nevezik. A rendszer elemeinek attribútumait attribútumjelzők reprezentálják. Az eseményjelzők, valamint a nem állandó rendszerelemek attribútumainak kezelése valamilyen listakezelési módszerrel történik, ami gyakorlatilag valamennyi szimulációs nyelv, rendszer-, ill. szoftver-csomag részét képezi. Az események az eseményidőnek megfelelően valamilyen jósolt eseménylistába vannak sorolva. Az eseményleíró rutinoknak a következő feladatok végrehajtásáról kell gondoskodniuk [16]:

- Az állapotváltozók értékének megváltoztatása az attribútum, ill. eseményjelzők függvényében.
- A program további folytatásához szükséges eseményjelzők generálása.
- Adatgyűjtés a szimulációs program által szolgáltatott eredmény számára.

Aktivitás leírási mód

Ennél a módszernél az események helyébe aktivitások lépnek. Egy aktivitás szintén legalább egy állapotváltozó értékét változtatja meg, és az eseményekhez hasonlóan az aktivitások végrehajtási ideje is zérus. Lényeges különbség azonban, hogy míg egy esemény az eseményideje elérésekor következik be, addig egy aktivitás akkor és csak akkor, amikor az állapotváltozók és a szimulációs rendszeridő értéke eleget tesz bizonyos követelményeknek.

Az aktivitásleíró rutin feladata kettős: egy feltétel vizsgálatból és egy végrehajtásból áll. Először elvégzi azokat a vizsgálatokat, amelyek alapján eldönti, hogy az aktivitás végrehajtható-e. Amennyiben az eredmény pozitív, úgy gondoskodik a megfelelő állapotváltozó értékének megváltoztatásáról, az időváltozók értékének aktualizálásáról és - mint az előző módszernél, úgy itt is - adatgyűjtésről a szimulációs programfutás eredménye számára.

Folyamat leírási mód

A folyamat leírási módnál a lényeg a folyamatokon van, melyek mindig egy rendszerelem - rendszerben tartózkodási ideje alatti - viselkedését írják le. Azonos folyamatosztályba tartoznak az azonos módon jellemezhető folyamatok, ezeket a leírásokat szokás folyamatleírásnak nevezni. Minden alkalommal, amikor egy folyamatleíró szubrutin az aktuális paramétereivel hívásra kerül, szimuláljuk az adott folyamatot. Itt azonban alapvető különbséget találunk az eddigiekhez képest. A folyamatok ugyanis véges szimulált rendszeridő alatt zajlanak le. Mivel az egyes folyamatok párhuzamosan mennek végbe a rendszerben, és az ezek által megváltoztatott attribútumok értékén át kölcsönhatásban vannak, így egy folyamat csak addig az utasításig hajtható végre, amíg egy olyan utasítást nem talál, amely már "jövőbeni" időpontra hivatkozik. Ez az utasítás viszont csak akkor hajtható végre, ha már egyik folyamatnak sincs olyan része, ami közelebbi időre vonatkozik. Egy ilyen utasítás címét **reaktivációs pont**nak nevezzük. Folyamat leírási mód vezérléséhez tehát a fenti ismeretek szükségesek.

4. Szimulációs modell felépítése

§ **Definíció. Mobil entitások:** azok a személyek, tárgyak, objektumok, stb., melyeken különböző műveleteket kell végrehajtani, például kiszolgálni őket, összegyűjteni vagy átalakítani.

§ **Definíció. Verifikálás** alatt értjük a szimulációs modellt leíró procedurális (azaz utasításokat tartalmazó) program vagy nonprocedurális (azaz objektumokkal leírható) szerkezet ellenőrzését. Ez elsősorban szintaktikai ellenőrzést jelent, ahol azt vizsgáljuk, hogy helyesen működik-e (a modellt reprezentáló) program. A nonprocedurális szerkezetnél pedig azt ellenőrizzük, hogy betartja-e a szerkezeti, felépítési szabályokat a modell.

§ **Definíció. Validálás** a modell és a valóság közötti leképezés helyességének ellenőrzését jelenti. Ez egy olyan érvényességvizsgálat, ahol a modelltől csak az adott célnak megfelelő reprezentációt várjuk el és azt is csak egy előre megadott hűséggel. A vizsgálat tehát arra irányul, hogy megnézzük a procedurális vagy nonprocedurális modell valóban a kívánt rendszert modellezi-e.

A következőkben a modellek (általános modellt feltételezve, melybe beletartozik a véges automaták modellosztálya, a sorbanállási rendszerek modellosztálya, melyet később részletezünk) kialakításának lépéseit mutatjuk be szimulációs sajátosságok figyelembe vételével. A szimulációs modell struktúráját a modellt alkotó elemek kapcsolatrendszere alakítja ki, a szimuláció futtatás célja pedig egy olyan teljes modell (megfelelő struktúrával és paraméterezéssel történő) megtalálása, mely híven reprezentálja a valóságot a kapcsolatok rendszerével, és ezáltal egy jól használható eszközt ad a döntéshozók kezébe.

4.1. Szimulációs modell felépítésének fázisai

Egy valóságos rendszer modelljének felépítése a következő lépésekben történik:

a) Első lépésként az informatikai környezeti paraméterek beállítása címen kell előkészíteni a szimulációs modell futtatásához szükséges szoftver eszközöket, így az operációs rendszert, szimulátort installálni stb. Az installálás után egy teljesítményvizsgálattal lehet meggyőződni, hogy az adott hardver majd mekkora nagyságú modell szimulációját teszi lehetővé.

b) A következő lépésben el kell határolni a modellt a környezetétől, azaz meghatározni, hogy mely részek tartozzanak a modellhez és melyek nem. A modell és környezetének interakciói a modell input és output csatornáin keresztül történnek, így definiálni kell, hogy milyen input adatokat kell előállítani a modell számára. Különösen nagy hangsúlyt kell fektetni a bemenő adatok minőségére, hiszen rossz vagy pontatlan adatokkal hiába építünk fel egy jó modellt, az eredményünk hamis következtetésre enged jutni.

c) A szimulációs modell struktúrájának kialakítása a modellt alkotó objektumok meghatározásával kezdődik. A modell építése során figyelembe kell venni a modellben résztvevő elemek közötti kölcsönhatásokat. A feladat megoldása során olyan általános modelleket érdemes felépíteni, amelyek túlmutatnak a konkrét feladaton és más környezetben is alkalmazhatók. Az általános célkitűzésnek megfelelően olyan kölcsönhatás-mechanizmust

kell találni a modell segítségével, mely demonstrálni tudja az igazi rendszerben végbemenő folyamatokat, és a modell moduláris felépítése lehetővé tudja tenni más topológiai és más paraméterekkel bíró modellek felépítését is.

d) A probléma megoldásához vezető úton a modellstruktúra kialakítása után – ami gyakorlatilag az eredeti objektum leírásához általunk kiválasztott változók, állapotok kapcsolatainak leírását jelenti – a modellhez tartozó paraméterek meghatározása a feladat. Itt kell megadni a kezdőértékeket, együtthatókat, stb., melyek meghatározásával eljutunk a dinamikus modellünkhöz, ami már a teljes, szimulációs környezetben futtatható modellt jelenti. A modell paraméterein kívül rögzíteni kell a szimulációs környezeti beállításokat is, hogy mekkora lépésközzel, milyen hosszan történjen a futtatás, stb. A szimulációs futtatások előtt a modellezőnek el kell végezni a modell validációját és verifikálását is.

Az első két lépés részben túl technikai, részben pedig túlságosan egyedi (problémáktól függően nagyon eltérő lehet) ahhoz, hogy általános útmutatót lehessen hozzá még adni. A szimulációs modell struktúrájának kialakításánál (és a struktúrából adódó megbízhatóság esetén) már azonban beszélhetünk általános felépítési elvekről, nézzük meg ezeket:

Tekintsünk egy n komponensből álló rendszert [38], ahol minden komponensnek két állapotát különböztetjük meg: működő és nem működő állapotát. Az i -edik komponensnek ezt a bináris állapotát jelöljük: s_i -vel, értéke pedig legyen 1: ha működik és 0: ha nem működik.

$$s_i = \begin{cases} 1 \\ 0 \end{cases} \quad (34)$$

Nem csak egy komponensnél vezethetjük be a működést reprezentáló állapot változót, hanem az egész rendszerre általánosíthatjuk ezt. **Struktúra függvénynek** nevezzük azt a többváltozós függvényt, amely a komponensek kapcsolódásai alapján leképezi a komponensek állapotait a teljes rendszer bináris állapotát jelképező állapotra.

$$\varphi(s_1, s_2, \dots, s_n) = \begin{cases} 1 \\ 0 \end{cases} \quad (35)$$

A struktúra függvény értéke 1, ha a teljes rendszer működőképes, egyébként pedig 0.

Néhány gyakori struktúrájú komponens együttes struktúra függvénye:

a) Soros kapcsolású struktúra

Ha bármelyik komponens nem működik, akkor a teljes rendszer sem működőképes. Ezt többféle struktúra függvénnyel is leírhatjuk, például:

$$\varphi(s_1, s_2, \dots, s_n) = \bigwedge_{i=1}^n (s_i) \quad (36)$$

vagy

$$\varphi(s_1, s_2, \dots, s_n) = \prod_{i=1}^n s_i \quad (37)$$

vagy

$$\varphi(s_1, s_2, \dots, s_n) = 1 - \bigvee_{i=1}^n (1 - s_i) \quad (38)$$

vagy

$$\varphi(s_1, s_2, \dots, s_n) = \underset{i=1}{\overset{n}{\text{Min}}}(s_i) \quad (39)$$

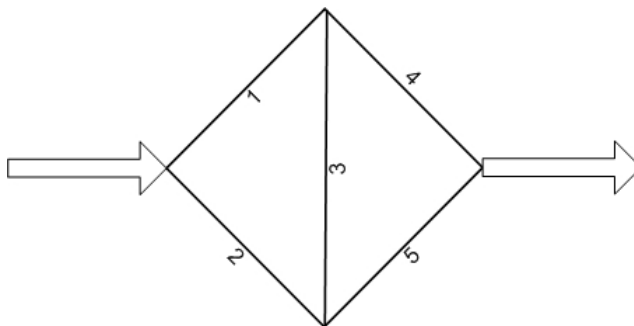
b) Párhuzamos kapcsolású struktúra

Az egész rendszer csak abban az esetben nem működik, ha az összes komponens működésképtelen, egyébként pedig a teljes rendszer működőképes. Ez szintén leírható többféle struktúra függvénnyel is, de most csak egyet emelünk ki közülük:

$$\varphi(s_1, s_2, \dots, s_n) = \underset{i=1}{\overset{n}{\text{Max}}}(s_i) \quad (40)$$

c) Hídkapcsolás

Legtöbb struktúra az előző kettő kombinációiból előállítható, van azonban olyan kapcsolati rendszer a komponensek között, amely nem vezethető le a soros és párhuzamos kapcsolatokról. Ilyen például az alábbi ábrán látható hídkapcsolás:



3. ábra Hídstrukturá

Ebben az esetben a teljes rendszer működését leíró struktúra függvény a következőképpen néz ki:

$$\varphi(s_1, s_2, s_3, s_4, s_5) = \underset{i=1}{\overset{n}{\text{Max}}}\{s_1 \cdot s_3 \cdot s_5, s_2 \cdot s_3 \cdot s_4, s_1 \cdot s_4, s_2 \cdot s_5\} \quad (41)$$

Ugyanis az 1-3-5, 2-3-4, 1-4 és 2-5 adja az összes lehetséges utat, a teljes rendszer pedig akkor működőképes, ha a lehetséges utak közül legalább az egyiknek minden eleme működik. Ebből a három struktúra típusból már elég sok kapcsolatrendszer felépíthető, de természetesen vannak még ennél bonyolultabb struktúrák is, amelyek nem származtathatók a felsorolt típusoknak (mint építőköveknek) a kombinációiból. Ezek szintén egyedi struktúra függvény leírást igényelnek. De általános megközelítésük a következő:

$$\varphi(s_1, s_2, \dots, s_n) = \underset{j=1}{\overset{k}{\text{Max}}}\{path_j\} \quad path_j = \prod_{i \in path} s_i \quad (42)$$

ahol a $path_j$ az összes lehetséges útvonalon végigmegy (k az utak száma), és értéke a hozzátartozó utak állapot-változóinak szorzatából áll.

Megbízhatósági függvény

Egy komponens megbízhatóságán annak a valószínűségét értjük, hogy a komponens működőképes állapotban van. Ez a meghibásodási valószínűség ellentéte, azaz a két említett

valószínűség összege 1. Ugyanígy értelmezzük a teljes rendszer megbízhatóságát is, a **megbízhatósági függvény** [38] pedig a komponensek megbízhatóságaiból fejezi ki ezt az egész rendszerre vonatkozó valószínűséget. A megbízhatósági függvény struktúrafüggő, így például a soros kapcsolású komponensek esetén:

$$m(p_1, p_2, \dots, p_n) = \prod_{i=1}^n p_i \quad p_i = P(s_i = 1) \quad (43)$$

ahol p_i értékek az egyes komponensek működőképességeinek valószínűségei, azaz megbízhatóságai. Párhuzamos kapcsolású struktúra esetén a megbízhatósági függvény:

$$m(p_1, p_2, \dots, p_n) = 1 - \prod_{i=1}^n (1 - p_i) \quad (44)$$

A fenti hídkapcsolású struktúra esetén a megbízhatósági függvény:

$$m(p_1, p_2, p_3, p_4, p_5) = 1 - (1 - p_1 p_3 p_5)(1 - p_2 p_3 p_4)(1 - p_1 p_4)(1 - p_2 p_5) \quad (45)$$

A fenti példák bemutatásával már látszik a különböző struktúra függvények és a megbízhatósági függvény közti kapcsolat, így ezektől eltérő struktúra esetén is felírható a megbízhatósági függvény.

A struktúra kialakítása után nézzük meg, hogy ha a felépített vázba entitások érkeznek a váznak megfelelő lehetséges útvonalakon feltöltve a statikus komponenseket, akkor milyen módon történhet a modellezés. A legtöbb ilyen modell hasonlít a tároló és sorbanállási rendszerekre, hisz az entitások a modell komponensein végighaladva várakoznak, tárolódnak, stb., ezért most röviden bemutatjuk az ilyen rendszerek szimulációjára vonatkozó ismereteket.

4.2. Kendall-féle osztályozás

§ Definíció. Erőforrás: kiszolgáló egységek, melyek a mobil entitásokat fogadni képesek, és különböző kiszolgálási tevékenységeket hajtanak végre.

§ Definíció. Technológiai útmeghatározás (routing): kiszolgálási tevékenységek sorrendjének leírása.

§ Definíció. Pufferek (buffers): azok a helyek, ahol a mobil entitások kiszolgálásra várakozhatnak.

§ Definíció. Ütemezés: előírt időrendi táblázatok, mint például erőforrások időtáblázatai, mobil entitások menetrendje, stb.

§ Definíció. Sorrendezés (sequencing): alatt értjük a sorban állási szabályok (FIFO: First In First Out, vagy más néven még: FCFS; LIFO: Last In First Out, vagy más néven még: LCFS, stb.) összességét. A modell egy adott pontján természetesen e szabályok közül 1 van érvényben.

§ Definíció. Beérkezési eloszlás: a sorbanállási rendszereknél a sorhoz csatlakozó entitások érkezésére vonatkozik. Egy adott beérkezési eseménysort két szempontból is lehet vizsgálni: a

két beérkezés között eltelt idő szerint és egy adott időegység alatt beérkezett entitások száma szerint. E kettő változó két különböző eloszlást követ, például ha beérkezések közötti idő exponenciális eloszlást követ, akkor az időegység alatti beérkezések száma Poisson eloszlású, mégis ugyanarra az eseménysorra vonatkoznak. A beérkezési eloszlás megadásánál tehát az egyértelműség miatt meg kell adni, hogy a két szempont közül melyiknek az eloszlását tekintettük.

§ Definíció. Kiszolgálási eloszlás: a sorbanállási rendszereknél a sorban álló entitások kiszolgálására vonatkozik. Ezt is lehetne két szempontból vizsgálni: a két kiszolgálás között eltelt idő (azaz a kiszolgálás időtartama) szerint és egy adott időegység alatt kiszolgált entitások száma szerint. Azonban ha külön nem hangsúlyozzák, akkor mindig az elsőt értik alatta, tehát a kiszolgáláshoz szükséges időtartamok eloszlását tekintjük.

Teljesítmény indikátorok:

§ Definíció. Teljesítmény: egységnyi idő alatt végzett (a modell előre definiált pontjáig eljutott vagy a rendszerből eltávozott) mobil entitások száma.

§ Definíció. Foglaltság: pufferekben levő mobil entitások átlagos száma illetve aránya a puffer teljes kapacitásához képest.

§ Definíció. Sorhossz: sorban állásra felkészített pufferekben levő mobil entitások pillanatnyi illetve átlagos száma.

§ Definíció. Várakozási idő (késleltetési idő): mobil entitások várási ideje a kiszolgálásuk előtt.

§ Definíció. Erőforrás kihasználtság: erőforrások használatban levő összesített ideje a teljes futási időhöz viszonyítva.

§ Definíció. Rendszer veszteségi ráta (system loss rate): véges kapacitású puffereknél előfordulhat, hogy nem tudnak több mobil entitást fogadni. Ha a nem fogadott entitás a rendszeren belülről érkezik (például másik pufferből), akkor a helyén marad; viszont ha a rendszeren kívülről érkezik, akkor nem tud belépni a modellbe és elveszik. Egy entitás elveszhet a rendszer számára úgy is, hogy a mobil entitás nem hajlandó túl sokat várni, kiállva a sorból elhagyja a rendszert (itt az entitásnak van egy maximális tolerálható várakozási ideje). Tehát a különböző korlátok miatt elveszett mobil entitások és az összes entitás arányát nevezik rendszer veszteségi rátának.

A tároló és sorbanállási rendszerek szimulációjánál a modellező kíváncsi lehet a sorhosszra, aktuális tároló tartalomra, átlagos tartalomra, várakozási időre és egyéb statisztikákra. A sorbanállási problémákat egységes módon lehet kezelni, azaz például ha az entitások egyenletes eloszlás szerint érkeznek a modellbe, ahol 2 sor van, és mindkét sornál a kiszolgálási idő determinisztikus (és egyéb feltételek is ugyanazok), akkor az entitásokra vonatkozó összes statisztikai adat ugyanaz minden rendszernél, függetlenül attól, hogy személyek, munkadarabok vagy járművek, stb. sorbanállásáról van szó.

A sorbanállási problémák leírásának egységesítését David G. Kendall végezte bevezetve a következő jelölésrendszert:

$$X / Y / s / r / k / p$$

Az 1. paraméter: X a beérkezési eloszlást írja le, melyre a következő standard rövidítések használhatók (mindegyik esetben a beérkezések egymástól függetlenek):

M: beérkezések közötti idő exponenciális eloszlást követ, azaz időegység alatti beérkezések száma Poisson eloszlású (az M rövidítés a Markov névből származik az emlékezetnélküliség miatt).

E_k : beérkezések közötti idő Erlang eloszlású k alakparaméterrel.

D: két beérkezés között eltelt idő determinisztikus.

GI: két beérkezés között eltelt idő valamilyen más, általános (general) eloszlásból származik (input).

Az 2. paraméter: Y a kiszolgálási idő eloszlását írja le, melyre a következő standard jelölések használhatók:

M: kiszolgálási idő exponenciális eloszlást követ, azaz időegység alatti kiszolgáltak száma Poisson eloszlású.

E_k : kiszolgálási idő Erlang eloszlású k alakparaméterrel.

D: a kiszolgálási időtartam determinisztikus.

G: a kiszolgálási időtartam valamilyen más, általános (general) eloszlásból származik.

A 3. paraméter: s a kiszolgálók számát jelöli, azaz egyszerre párhuzamosan maximum ennyi igényvel tud foglalkozni a rendszer.

A 4. paraméter: r (rule) a sorbanállási szabályt írja le:

FIFO (First In First Out) vagy más néven még FCFS (First Come First Served): beérkezés sorrendjében szolgálják ki az igényeket.

LIFO (Last In First Out) vagy más néven még LCFS (Last Come First Served): beérkezés fordított sorrendjében szolgálják ki az igényeket.

SIRO (Service In Random Order): véletlen sorrendben történik a kiszolgálás.

GD (General queue Discipline): általános kiszolgálási szabály.

A 5. paraméter: k a rendszer kapacitását adja meg, azaz a rendszerben maximálisan megengedhető igények számát jelöli.

A 6. paraméter: p (population) az alapsokaság nagyságát adja meg.

Az utolsó három paraméter értéke nagyon sok modellnél: $GD/\infty/\infty$, így ezeknél a korlátozás nélküli eseteknél ezt az utolsó hármast el is szokták hagyni. Így például: GI/M/4 jelenti azt a rendszert, ahol a bemenet tetszőleges eloszlású, a kiszolgálási idő exponenciális eloszlású, és 4 kiszolgáló van a rendszerben, továbbá a kiszolgálási szabály általános, nincs korlát sem a kapacitásnál, sem az igények alapsokaságára vonatkozóan [53].

Egy másik érdekesség, hogy a tároló és a sorbanállási rendszereket együtt is lehet kezelni az alábbi módon [16]:

$$Z_{t+1} = [Z_t + X_t - Y_t]^+ \quad (46)$$

ahol a $+$ jel jelenti azt a műveletet, hogy ha a szögletes zárójel értéke negatívvá válna, akkor helyette 0 lesz a kifejezés értéke. Az alábbi táblázat mutatja, hogy a (46) egyenletben szereplő

jelölések a tároló és sorbanállási rendszerben mit jelenthetnek (egy sorban helyezkednek el a jelhez hozzárendelhető jelentések), így egy egységes kezelést lehet megvalósítani.

1. Táblázat Tároló és sorbanállási rendszerek egységes kezelése

Jelölés	Tárolórendszer	Sorbanállási rendszer várakozási idő	Sorbanállási rendszer sorhossz
Z_t	tároló tartalma a (t, t+1) intervallum elején	a t-edik igény várakozási ideje	sorhossz a t időpontban
X_t	bemeneti érték az intervallum kezdetén	t-edik igény kiszolgálási ideje	érkezések száma a (t, t+1) intervallumban
Y_t	kimeneti érték az intervallum végén	a t-edik és (t+1)-edik igény érkezése közötti időtartam	(t, t+1) intervallum alatti kiszolgálások száma

A sorbanállási, kiszolgálási rendszerek elmélete [22] erre a jelölésrendszerre épít, és az elmélethez kapcsolódóan különböző raktározási, tömegkiszolgálási problémák vizsgálhatók komoly matematikai apparátus segítségével [45][2]. Az érdeklődő olvasók számára ajánlható még a [44].

A fejezet elején említett negyedik modell kialakítási fázis, a paraméterezés a fent tárgyalt struktúra kialakításában résztvevő objektumok paramétereinek megválasztását jelenti. Ezek után kell a szimulációs modell dinamikus működtetéseként az elkészült modell(ek)e)t az első lépésben megválasztott szimulációs rendszerben lefuttatni. Itt a modellek legtöbbször szimbolikus grafikus módon kerülnek megjelenítésre, és a futtatás alatt a modellek működése animációval látható. A következő fejezetekben mutatjuk be a szimulációs futtatások végzésével és a szimulációs eredmények kiértékelésével kapcsolatos tudnivalókat.

5. Szimulációs futtatás

A szimuláció időtengelyének vizsgálatánál 3 időintervallumot különíthetünk el:

- felfutási idő,
- futási idő,
- leállás előkészítési idő.

A **futási idő** jelenti a teljes időtartamot, amit a modell szempontjából modellezni kell. Ez nem azonos a **gépidővel**, hiszen a számítógépek gyorsabban vagy lassabban is működhetnek, mint a valós rendszer (például ms-ok alatt lejátszódó gyors kémiai reakciót a gép pár percig szimulál, vagy több éves gazdasági modell szimulációja 1-2 óra alatt lefut).

A futási idő elejét nevezzük **felfutási időnek**, és ritkán előfordul az is, hogy a futási idő vége előtt szükség van egy rövid leállás előkészítési szakaszra is (a futási idő tartalmazza tehát a kezdeti és a végső szakaszt is). Ennek akkor van szerepe, ha adott állapotban szeretnénk a szimulációt befejezni és ennek az állapotnak az eléréséhez idő szükséges. Mivel ilyen alkalmazások ritkák, ezért a továbbiakban nem foglalkozunk ezzel az időintervallummal, a másik kettőt viszont részletesen megvizsgáljuk.

5.1. A felfutási idő analízis

Induláskor a szimulációs modell legtöbb esetben üres, azaz nem tartalmazza még a dinamikus objektumokat (mobil entitásokat). A statikus részbe menet közben érkeznek a dinamikus elemek, melyekhez bizonyos változók tartoznak. A szimuláció elején ezekben az esetekben lesz egy olyan időszak, ahol a dinamikus objektumokhoz tartozó változók értékeinek átlaga nem lesz egyenlő a hosszú távú átlaggal. Ezt az időintervallumot hívják **felfutási időnek** (warm up interval) [10]. Ez idő alatt érik el a változók a stacionárius (egyensúlyi) állapotot. Például gyárakban való termelés szimulációjánál idő kell, míg feltöltődik a rendszer, vagy bankfiókok modellezésénél nyitáskor még üres a rendszer, ott is időre van szükség, míg a bank működése el nem éri az egyensúlyi állapotot reprezentáló, ügyfelekkel teli helyzetet.

A felfutási idő probléma a szimuláció folyamatos kiértékelésénél okoz hibát. Azaz, ha a fent említett változókból menet közben statisztikát készítünk (például kiszámoljuk az időben megfigyelt értékek átlagát), akkor a felfutási idő értékei elrontják az egyensúlyi állapotban leolvasott értékeket (így az átlagra hibás eredményt kapunk).

A felfutási idő okozta hiba megszüntetésére bevezették a nyesett időátlag (vagy röviden nyesett átlag) fogalmát, mely azt jelenti, hogy a felfutási idő értékeit kihagyjuk az átlagszámításból. Jelöljük k -val a felfutási időt (azaz az első $k-1$ értéket hagyjuk figyelmen kívül) és t -vel a teljes futáshosszt, így a nyesett időátlag:

$$\overline{X}_t = \frac{1}{t - k + 1} \sum_{i=k}^t X_i \quad (47)$$

Többszörös független futtatás

Többszörös független futtatás esetén a szimulációs futtatásokat ugyanazon feltételek mellett kell végrehajtani, hogy az eredmények összesíthetők legyenek. Ezek a szimulációs futtatási feltételek a következők:

- indítsuk a szimulációt mindig az s_0 kiindulási állapotból,
- legyen k a felfutási idő lépésszáma,
- végezzük a futtatást t időpontig mindegyik esetben.

Jelöljük $X_j^{(i)}$ -vel az X modellváltozó (vagy entitás attribútuma) értékét a j -edik lépésben az i -edik ismétlés esetén. A szimulációs futtatásokat n -szer végrehajtva úgy, hogy a fenti feltételeket betartjuk: megkapjuk az X -re vonatkozó értékek halmazát, melyet az alábbi táblázatban foglalhatunk össze [10]:

2. Táblázat Többszörös független futtatási eredmények

i-edik futtatás	Lépések				Nyesett átlag (k felfutási idővel)
	1	2	...	t	
1	$X_1^{(1)}$	$X_2^{(1)}$...	$X_t^{(1)}$	${}_k \bar{X}_t^{(1)}$
2	$X_1^{(2)}$	$X_2^{(2)}$...	$X_t^{(2)}$	${}_k \bar{X}_t^{(2)}$
...
n	$X_1^{(n)}$	$X_2^{(n)}$...	$X_t^{(n)}$	${}_k \bar{X}_t^{(n)}$
Futtatások átlaga	$\bar{X}_{1,n}$	$\bar{X}_{2,n}$...	$\bar{X}_{t,n}$	${}_k \bar{X}_{t,n}$

Ebben a táblázatban a sorok végén láthatóak a k felfutási idővel (első $k-1$ elemet figyelmen kívül hagyjuk, a k -adik elemet pedig már beleszámítjuk az öt követőkkel együtt) számolt nyesett átlagok (trajektória minták nyesett átlaga), és az utolsó sorban az adott oszlopra vonatkozó n futtatás átlaga. Az utolsó oszlop legalsó cellája tartalmazza az utolsó oszlop értékeinek átlagát (trajektória minták nyesett átlagainak n futtatásra vonatkozó átlagát), mely megegyezik az utolsó sor értékeinek nyesett átlagával (egy időpontra vonatkozó futtatási értékek átlagának nyesett átlagával), így ezt röviden ${}_k \bar{X}_{t,n}$ -vel jelölhetjük. (Ha csak 1 hosszúságú nyesett átlagot szeretnénk jelölni, akkor a bal alsó sarokban levő index értékére a jobb alsó sarok első indexét kell írni. Az egyszerűsített jelölés érdekében, ha a bal alsó sarokban hiányzik az index, akkor mindig ezt az 1 hosszúságú nyesett átlagot értjük alatta, mint ahogy azt a táblázat legalsó sorában láthatjuk. $\bar{X}_{t,n} \equiv {}_t \bar{X}_{t,n}$)

\$ Állítás:

Trajektória minták nyesett átlagainak n futtatásra vonatkozó átlaga megegyezik az egy időpontra vonatkozó ismételt futtatási értékek átlagának nyesett átlagával.

\$ Bizonyítás

Definíciók alapján írjuk fel a különböző átlagokat! Egy adott (j) időpontra vonatkozó ismételt futtatási értékek átlaga:

$$\bar{X}_{j,n} = \frac{1}{n} \sum_{i=1}^n X_j^{(i)} \quad 1 \leq j \leq t \quad (48)$$

Trajektória minták nyesett átlaga (első k-1 elemet levágjuk):

$${}_k\overline{X}_t^{(i)} = \frac{1}{t-k+1} \sum_{j=k}^t X_j^{(i)} \quad 1 \leq i \leq n \quad (49)$$

Ismételt futtatási értékek átlagának nyesett átlaga:

$${}_k\overline{X}_{t,n}^A = \frac{1}{t-k+1} \sum_{j=k}^t \overline{X}_{j,n} = \frac{1}{t-k+1} \sum_{j=k}^t \left(\frac{1}{n} \sum_{i=1}^n X_j^{(i)} \right) \quad (50)$$

Trajektória minták nyesett átlagának ismételt futtatási átlaga:

$${}_k\overline{X}_{t,n}^B = \frac{1}{n} \sum_{i=1}^n {}_k\overline{X}_t^{(i)} = \frac{1}{n} \sum_{i=1}^n \left(\frac{1}{t-k+1} \sum_{j=k}^t X_j^{(i)} \right) \quad (51)$$

Jelen esetünkben a szumma műveletek megcserélhetők és a szorzó tényezők kihozhatók az összegzések elejére, így a két átlag az állításunknak megfelelően egyenlő:

$${}_k\overline{X}_{t,n}^A = \frac{1}{n} \cdot \frac{1}{t-k+1} \sum_{j=k}^t \sum_{i=1}^n X_j^{(i)} = {}_k\overline{X}_{t,n}^B \quad (52)$$

■

A felfutási idő analízisnél nem csak a fenti számolásokra van szükségünk, hanem szeretnénk azt is tudni, hogy meddig kell a modellt adatgyűjtés nélkül futtatni, azaz a nyesett átlag miatt szükségünk van a felfutási időre megállapítására.

§ Algoritmus. Felfutási idő hosszának becslése [10]:

1. Válasszunk egy előzetes trajektória minta hosszát: t.
2. Döntsük el a független ismételt futtatások számát: n.
3. Inicializáljuk az álvéletlenség generátort kezdeti értékekkel.
4. Hajtsuk végre a szimulációt n-szer úgy, hogy az elsőt kivéve (amikor az előző pontban inicializált kezdeti értékeket használjuk az álvéletlenség generáláshoz) minden ismételt futtatás elején használjuk az előző futtatás végén érvényben lévő álvéletlenség generátor állapotot (azaz a generátor kezdeti értékeit az előző futtatás végén kapott generátor mag határozza meg). Futtatások alatt tároljuk el a trajektória minták értékeit.
5. Számoljuk ki az egy időpontra vonatkozó ismételt futtatási értékek átlagát és e trajektória minták nyesett átlagait (úgy, hogy ez legyen a nyesési pont), majd ábrázoljuk grafikusán a két sorozatot pontsorozatként. A grafikus ábrázolás segít megállapítani, hogy hol ér össze a két pontsorozatot összekötő vonal (ha elég adat áll rendelkezésünkre).
6. Ha az eredményekből a megfelelő felfutási idő megállapítható, azaz a grafikonon jól látható, hogy hol ér össze a két vonal (ebben az esetben elegendő információval

rendelkezünk), akkor az iterációs becslési folyamat véget ér (a megállapított felfutási időt lehet használni majd egy hosszú futtatáshoz).

7. Ha az előző feltétel nem teljesül, akkor

- növeljük a t -t vagy
- növeljük az n -t vagy
- növeljük mind a kettőt,
és menjünk vissza a 4-es lépéshez.

A felfutási idő analízis segítségével tehát a felfutási probléma kiküszöbölhető. A statisztikai adatgyűjtést akkor érdemes csak elkezdeni, ha a felfutás véget ért, azaz a stacionárius állapotot elértük. Természetesen lehet olyan feladat is, amikor nem az egyensúlyi állapotra, hanem éppen az átmeneti állapotra, a felfutásra vagyunk kíváncsiak: ilyenkor az adatgyűjtést a felfutás alatt kell végezni, és a leállási feltétel egybe esik a felfutási idő végével, tehát itt is fontos, hogy ezt az időpontot megtaláljuk.

6. Szimulációs eredmények kiértékelése

A szimuláció során nem cél a valós rendszer minden szempont szerinti modellezése [50], hanem a legfőbb tulajdonságait kell csak reprezentálni. Ezeket a lényeges komponenseket viszont minél pontosabban kell a modellben megvalósítani, azaz a kiértékelésnél is ezekre kell fókuszálni.

6.1. Szimulációs eredmények hibái

A szimulációs modell kimenetén (szimulációs futtatás során előálló kimeneti értékekben) több fajta hiba fordulhat elő [10]:

- (1) Véletlen hiba
- (2) Szisztematikus hiba

A **véletlen hibát** egyrészt a bemeneti értékek okozhatják, ha azok nem teljes populációt reprezentálnak, hanem abból csak egy mintát képviselnek. Másrészt a szimuláció folyamán generált véletlenszámok is csak egy eloszlásból származó példányok, így azok felhasználása a modellben ugyancsak véletlen hibát okoz.

A **szisztematikus hibát** a kezdőállapot kiválasztása és a kezdeti körülmények, feltételek váltják ki (együttesen kezdeti szimulációs körülményeknek nevezzük). Ezeknek a körülményeknek a megválasztása egyoldalúan befolyásolhatja a végeredményt, ami ezt a szisztematikus hibát okozza.

Szisztematikus és véletlen hiba csökkentése

A következő táblázat mutatja szisztematikus és véletlen hiba csökkentésére irányuló beavatkozási módokat és azok következményeit.

3. Táblázat Szisztematikus és véletlen hiba csökkentése

Beavatkozási mód	Következmény
Felfutási idő növelése (futáshossz változatlan értéken tartása mellett).	Szisztematikus hiba csökken, véletlen hiba viszont nő (különösen, ha a felfutási idő megközelíti a futáshosszt).
Trajektória minta hosszának növelése (azaz a szimulációs idő meghosszabbítása).	Szisztematikus és véletlen hiba is csökken.
Független ismételt futtatások (azonos kezdeti állapotból való indítás esetén) számának növelése.	Véletlen hiba csökken, de a szisztematikus hibára nincs hatással.

A hibák közül van egy matematikailag jól leírható hiba: a **szórás**, mely a várható érték bizonytalanságának mértékét fejezi ki. Ha tehát várható értékkel áll kapcsolatban a megfigyelt kimenetünk, akkor célunk, hogy a szórást minimalizáljuk. Először nézzük át milyen tételek vonatkoznak közösen a szórásra és a szimulációs mintavételekre:

§ Tétel. Normál eloszlás mintavételi tétele

Ha normál eloszlásból veszek egy adott számú (n elemű) mintát, akkor a minta (mely ugyanúgy normál eloszlású lesz) átlaga a normál eloszlás várható értékéhez tart, és a minta szórása a normál eloszlás szórásától és a mintanagyságtól függ a következő képlet szerint:

$$\sigma_s = \frac{\sigma}{\sqrt{n}} \quad (53)$$

§ Tétel. Általános mintavételi tétel

Ha bármilyen eloszlásból is veszünk egy adott számú (n elemű) mintát, akkor n növelésével:

- a minta átlaga normális eloszlású lesz,
- a minta átlaga tart az eredeti eloszlás várható értékéhez és
- a minta szórása az eredeti eloszlás szórásának négyzetgyök n -ed részéhez tart, ugyanúgy, mint az (53) egyenletben.

Ezeknek a mintavételi tételeknek nagy jelentőségük van a szimulációnál, mivel legtöbbször a futtatás alatt gyűlnek össze (generálódnak) az adatok. Persze van olyan szimulációs modell is, ahol az elején áll rendelkezésre az egész adatsor, de az esetek többségében a felhasználónak érdekes változók csak menet közben kapnak értékeket (hisz éppen ezért alkalmaz szimulációt), azaz itt a fenti képletben is említett mintavételi n szám a futáshosszal arányos.

Mivel a szórás (ahogy az (53)-as egyenletből is látszik) csak a futáshossz négyzetgyökével arányosan csökken, ezért kiemelt fontosságú a szimulációnál, hogy a varianciát más módon is próbáljuk meg lecsökkenteni. Erre több különböző módszer is született:

- Feltételes variancia formula módszere,
- Ellentétes változók módszere,
- Kontroll változók módszere,
- Rétegzett mintavételezés.

Nézzük végig az egyes módszereket, melyek más és más körülmények mellett használhatók illetve érdemes használni őket:

6.2. Feltételes variancia formula módszere

A feltételes variancia módszeréhez először nézzünk meg egy segédtételt és egy máshol is használható tételt:

§ Tétel.

$$E(X^2) = D^2(X) + E^2(X) \quad (54)$$

§ Bizonyítás.

Variancia (szórásnégyzet) definíciója alapján:

$$D^2(X) = E(X - E(X))^2 = E(X^2 - 2 \cdot X \cdot E(X) + E^2(X)) = \quad (55)$$

$$= E(X^2) - 2 \cdot E(X) \cdot E(X) + E^2(X) = E(X^2) - E^2(X) \quad (56)$$

Innen átrendezéssel már látható, hogy a kívánt állítás igaz. ■

§ Segédteétel.

$$E(E(X | Y)) = E(X) \quad (57)$$

§ Bizonyítás [36].

Ha X és Y is diszkrét valószínűségi változók:

X : x_1, x_2, x_3, \dots értéket felvéve p_1, p_2, p_3, \dots valószínűségekkel és

Y : y_1, y_2, y_3, \dots értéket felvéve q_1, q_2, q_3, \dots valószínűségekkel és a közös eloszlás:

$P(X=x_i, Y=y_j) = r_{ij}$, akkor a következőképpen írhatjuk fel a feltételes várható értéket:

$$E(X | Y = y_i) = \sum_j x_j P(X = x_j | Y = y_i) \quad (58)$$

Az $E(X|Y)$ szintén egy diszkrét valószínűségi változó, melynek lehetséges értékei:

$$E(X|Y=y_1), E(X|Y=y_2), E(X|Y=y_3), \dots$$

és ezt az $E(X|Y=y_i)$ értéket q_i valószínűséggel veszi fel, melyet a várható érték számolásnál használunk fel:

$$E(E(X | Y)) = \sum_i E(X | Y = y_i) q_i = \quad (59)$$

$$= \sum_i \left(\sum_j x_j P(X = x_j | Y = y_i) q_i \right) = \quad (60)$$

$$= \sum_i \sum_j x_j r_{ij} = \sum_j x_j \sum_i r_{ij} = \sum_j x_j p_j = E(x) \quad (61)$$

Folytonos X és Y változók esetén a Σ jeleket integrál jelek váltják fel, majd hasonló lépésekben (részletes bizonyítást most nem közöljük) eljuthatunk az eredményig. ■

A feltételes valószínűségű változónál megvizsgáltuk tehát a várható értéket. Vizsgáljuk meg a szórást is, melyet a következő módon értelmezhetünk:

$$D(X | Y) = \sqrt{E\{(X - E(X | Y))^2 | Y\}} \quad (62)$$

Erre támaszkodva felírhatjuk a következő tételt:

§ Tétel. Feltételes variancia formula

A variancia felírható az alábbi két feltételes valószínűséget tartalmazó tag összegeként:

$$D^2(X) = E(D^2(X | Y)) + D^2(E(X | Y)) \quad (63)$$

§ Bizonyítás [38].

A (54)-es tétel igaz X helyett $X|Y$ feltételes valószínűségváltozóra is, így:

$$D^2(X | Y) = E(X^2 | Y) - E^2(X | Y) \quad (64)$$

Ha mindkét oldalnak vesszük a várható értékét, akkor:

$$E(D^2(X|Y)) = E(E(X^2|Y) - E^2(X|Y)) \quad (65)$$

$$E(D^2(X|Y)) = E(E(X^2|Y)) - E(E^2(X|Y)) \quad (66)$$

A (57)-es segédtétel alapján az első tényező egyszerűsíthető, így

$$E(D^2(X|Y)) = E(X^2) - E(E^2(X|Y)) \quad (67)$$

Másrésről a (54)-es tétel X helyett igaz a feltételes valószínűségváltozó várható értékére is (azaz cseréljük ki az (55) egyenletben az X-et $E(X|Y)$ -re), így:

$$D^2(E(X|Y)) = E(E^2(X|Y)) - (E(E(X|Y)))^2 \quad (68)$$

A (57)-es segédtétel alapján a második tényező egyszerűsíthető:

$$D^2(E(X|Y)) = E(E^2(X|Y)) - (E(X))^2 \quad (69)$$

Adjuk össze a (67) és az (69) egyenleteket kiejtve ezzel a (67) egyenlet jobb oldalán szereplő második tagot:

$$E(D^2(X|Y)) + D^2(E(X|Y)) = E(X^2) - (E(X))^2 \quad (70)$$

A (56) egyenlet alapján megállapíthatjuk, hogy a jobb oldal nem más, mint X varianciája, így

$$D^2(X) = E(D^2(X|Y)) + D^2(E(X|Y)) \quad (71)$$

ahogy azt az állításban említettük. ■

Mivel a variancia sosem lehet negatív (így a variancia várható értéke sem), ezért a feltételes variancia formula tétele alapján felírhatjuk a következő egyenlőtlenséget:

$$D^2(X) \geq D^2(E(X|Y)) \quad (72)$$

Ha egy X változó (szimulációs futtatás kimeneti változója) várható értéke érdekel bennünket, azaz egy olyan paramétert szeretnénk becsülni, melyre: $\theta = E(X)$, és van egy olyan Y másik változó, melynél ismerjük a $E(X|Y)$ -t; akkor az X helyett érdemes az $E(X|Y)$ -t használni. Ugyanis $E(X|Y)$ is egy torzítatlan becslője a θ -nak:

$$E(E(X|Y)) = E(X) = \theta \quad (73)$$

ráadásul kisebb (vagy egyenlő) varianciával:

$$Var(X) \geq Var(E(X|Y)) \quad (74)$$

6.3. Ellentétes változók módszere

Ha egy X változóra (melynek a várható értéke érdekel bennünket, azaz $\theta = E(X)$) két kísérletet végzünk: X_1 és X_2 , akkor ennek a két változónak az átlagát használhatjuk majd fel a várható érték becslésre, és a két változó átlagának varianciája a következő:

$$D^2\left(\frac{X_1 + X_2}{2}\right) = \frac{1}{4}(D^2(X_1) + D^2(X_2) + 2Cov(X_1, X_2)) \quad (75)$$

A képlet alapján ez a szórásnégyzet akkor lenne kicsi, ha a két változó független lenne egymástól, mert ez esetben az utolsó tag (a kovariancia) nulla érték lenne. Azonban még előnyösebb lenne, ha a kovariancia negatív lenne, ez esetben az első két tag összegéből még le lehetne vonni egy értéket. A negatív kovarianciát mesterségesen úgy érhetjük el, hogy két olyan erősen összefüggő X_1 -et és X_2 -t hozunk létre, melyek között negatív korreláció áll fenn. Azaz X_1 -nek és X_2 -nek **ellentétes változóknak** kell lenniük [38].

Hogyan tudjuk ezt elérni? Tegyük fel, hogy X_1 m db egyenletes eloszlású, 0 és 1 közé eső véletlen számnak valamilyen függvénye:

$$X_1 = f(U_1, U_2, \dots, U_m) \quad (76)$$

ahol az U_i véletlen számok egymástól függetlenek. Mivel az U_i -k a $[0,1]$ intervallumon egyenletes eloszlású számok, ezért az $1-U_i$ számok is ugyanebbe az intervallumba esnek ugyanolyan egyenletes eloszlással. Tehát:

$$X_2 = f(1-U_1, 1-U_2, \dots, 1-U_m) \quad (77)$$

függvénnyel használt X_2 szám ugyanolyan eloszlású lesz, mint az X_1 függetlenül attól, hogy ez milyen eloszlás. Mivel az $(1-U)$ negatív korrelációban áll az U -val, ezért nagy valószínűséggel az X_1 és X_2 közötti kovariancia negatív lesz. Ha ez a többdimenziós f függvény monoton függvénye minden egyes koordinátájának, akkor ez bizonyíthatóan is negatív korrelációt eredményez X_1 és X_2 között [38].

Ezt az eredményt tehát úgy használhatjuk fel, hogy először m álvéletlenszámot (U_1, U_2, \dots, U_m) generálunk X_1 kiszámításához. Majd ezek után ahelyett, hogy újabb m álvéletlenszámot generálnánk, ezeket használjuk fel még egyszer az X_2 számolásánál úgy, hogy mindegyik értéket kivonjuk 1-ből ($1-U_1, 1-U_2, \dots, 1-U_m$). Így kapunk tehát két X -et, melyek kicsi varianciát eredményeznek. A másik nagy előnye ennek a módszernek az, hogy fele annyi álvéletlenszámot kell csak generálni, ami jelentősen meggyorsíthatja a szimulációt.

Nézzünk erre egy példát [38], hogy hogyan használhatjuk ki mindezt! Szeretnénk egy 1 kiszolgálós sorbanállási rendszert szimulálni, ahol a várakozási idők összegére vagyunk kíváncsiak. Jelöljük az i -edik igény várakozási idejét W_i -vel, kiszolgálási idejét S_i -vel ($i = 1, 2, \dots, n$), az i és az $(i+1)$ -edik igény érkezése között eltelt időt: T_i -vel ($i = 1, 2, \dots, n-1$). Így T_1 az első és a második közt eltelt idő, azaz T_1 megmutatja, hogy a 2. mennyivel később érkezett az elsőhöz viszonyítva. Ekkor a veszteség nélküli esetre a várakozási idő (ahogy azt a sorbanállási rendszerek tárgyalásánál már láttuk):

$$W_{i+1} = W_i + S_i - T_i \quad (78)$$

Mi a várakozási idők összegének várható értékét szeretnénk becsülni, melyre a következő paramétert vezetjük be:

$$\theta = E(X) = E\left(\sum_{i=1}^n W_i\right) \quad (79)$$

A szimuláció során az S_i és a T_i értékeket állítjuk elő véletlenszám generátorok segítségével, így X valamilyen f függvénye ezeknek a véletlenszámoknak:

$$X = f(T_1, T_2, \dots, T_n, S_1, \dots, S_n) \quad (80)$$

Ez az f függvény az S_i értékek növelésével nő, T_i értékek csökkentésével nő, azaz monoton növekvő függvénye a koordinátáinak, így a véletlenszám előállításnál az inverz-transzformációs módszer, a variancia csökkentésnél pedig az ellentétes változók módszere használható. Tehát a szimuláció alatt úgy képezzük a kívánt eloszlású $2n$ db véletlenszámot, hogy veszünk $2n$ egyenletes eloszlásút (u 0 és 1 között lehet) és annak egyik felét az érkezési időkre, másik felét a kiszolgálási időkre transzformáljuk a következő képlet szerint:

$$T_i = F^{-1}(u_i); \quad S_i = G^{-1}(u_{n+i}) \quad i = 1, 2, \dots, n \quad (81)$$

Variancia csökkenő módszer nélkül minden egyes szimulációs futtatásnál $2n$ számot kell tehát generálni. Az ellentétes változók módszerével viszont megtehetjük, hogy minden második futtatásnál a véletlenszám előállítást kihagyjuk, és helyette az előző ciklusban generált számokat $1-u$ formában használjuk fel, azaz:

$$T_i = F^{-1}(1 - u_i); \quad S_i = G^{-1}(1 - u_{n+i}) \quad i = 1, 2, \dots, n \quad (82)$$

Így nem csak véletlenszám generálást takarítunk meg, hanem az X változó becslésénél ugyanannyi ciklus esetén pontosabb értéket is kapunk.

6.4. Kontroll változók módszere

Ha a szimuláció valamelyik kimeneti változójának (jelöljük X -el) várható értékét: $\theta = E(X)$ -et szeretnénk megbecsülni, és van egy másik kimeneti változó (jelöljük Y -nal), melynek a várható értékét tudjuk $E(Y) = \mu_Y$, akkor használhatjuk a kontroll változók módszerét [38]. Ennek a lényegét a következőkben ismertetjük. Képezzünk egy új változót az alábbi egyenlet alapján:

$$X + c(Y - \mu_Y) \quad (83)$$

Ahogy a fenti egyenletben láthatjuk az X -hez hozzáadva egy kifejezést egy olyan mennyiséghez jutunk, amely szintén torzítatlan becslése a θ -nak. Ahhoz, hogy a legjobb c -t megtaláljuk, nézzük meg ennek a mennyiségnek a varianciáját:

$$\text{Var}(X + c(Y - \mu_Y)) = \text{Var}(X + c \cdot Y) = \text{Var}(X) + c^2 \text{Var}(Y) + 2 \cdot c \cdot \text{Cov}(X, Y) \quad (84)$$

mivel ha az Y változóból egy konstans értéket (μ_y) levonunk, akkor a variancia nem változik; továbbá a két változó kölcsönhatását a kovariancia fejezi ki. Ennek a kifejezésnek a minimumát keressük a legjobb c érték meghatározásához, amit deriválással kaphatunk meg, a derivált:

$$2 \cdot \text{Var}(Y) \cdot c + 2 \cdot \text{Cov}(X, Y) = 0 \quad (85)$$

A derivált kifejezést nullává téve a legjobb c érték (ahol variancia minimális):

$$c_1 = -\frac{\text{Cov}(X, Y)}{\text{Var}(Y)} \quad (86)$$

Ennél a módszernél az Y változót nevezzük kontroll változónak, amely a következőképpen tud tehát segíteni. Három esetet különböztethetünk meg az X és Y változó közötti korreláció szempontjából: a korreláció pozitív, negatív vagy nulla.

- Ha X és Y pozitív korrelációban (nagyobb X mellett az Y is nagyobb) áll egymással, akkor c_1 negatív. Ha Y kimeneti érték éppen nagyobb a várható értékénél (azaz $Y - \mu_Y$ pozitív), akkor az ebben a részben feltett pozitív korreláció miatt nagy valószínűséggel az X is nagyobb lesz a saját várható értékénél. A negatív c_1 miatt a $c_1(Y - \mu_Y)$ -nal növelt új mennyiség már visszahúzza (itt csökkenti) a várható érték felé az X -et, így a végső variancia sokkal kisebb lesz.
- Ugyanezt végignézve negatív korreláció esetén is tapasztalható a visszahúzás (itt növeléssel) a várható érték felé.
- Nulla korreláció esetén a módszer nem járul hozzá a variancia csökkenéséhez, hiszen a kovariancia ebben az esetben nulla.

A variancia tehát nulla korrelációs esetet kivéve mindenképpen csökken. A (84)-es egyenletbe behelyettesítve a (86)-t megkapjuk, hogy mennyire csökkent a variancia:

$$\text{Var}(X + c_1(Y - \mu_y)) = \text{Var}(X) - \frac{\text{Cov}^2(X, Y)}{\text{Var}(Y)} \quad (87)$$

Azt is felírhatjuk, hogy az eredeti $\text{Var}(X)$ -hez képest hányad részére tudtuk csökkenteni a szórásnégyzetet:

$$\frac{\text{Var}(X + c_1(Y - \mu_y))}{\text{Var}(X)} = 1 - \text{Corr}^2(X, Y) \quad (88)$$

ahol Corr^2 az X és Y között fellépő lineáris korrelációs együttható négyzete.

Nézzünk egy példát [38] a kontroll változók módszerére! Egy szimulációs modellben sok paraméter szimulálása történik, és tegyük fel, hogy van egy 0 és 1 között egyenletes eloszlású u változót használó e^u függvény, melynek a várható értékére vagyunk kíváncsiak. Azaz a következő paraméter becslését kell elvégezni:

$$\theta = E(e^u) = \int_0^1 e^x dx \quad (89)$$

A várható érték könnyen kiszámolható:

$$\int_0^1 e^x dx = [e^x]_0^1 = e - 1 \quad (90)$$

De itt a varianciára leszünk kíváncsiak.

$$Var(e^u) = \int_0^1 (e^x - (e - 1))^2 dx = \quad (91)$$

$$= \int_0^1 (e^{2x} - 2(e - 1)e^x + (e - 1)^2) dx = \quad (92)$$

$$= \left[\frac{e^{2x}}{2} \right]_0^1 - 2(e - 1)[e^x]_0^1 + (e - 1)^2[x]_0^1 = \quad (93)$$

$$= \frac{e^2 - e^0}{2} - 2(e - 1)(e^1 - e^0) + (e - 1)^2(1 - 0) = \quad (94)$$

$$= \frac{e^2 - 1}{2} - (e - 1)^2 = \quad (95)$$

$$= \frac{-e^2 + 4e - 3}{2} = 0,2420 \quad (96)$$

Hogyan tudjuk ezt a varianciát lecsökkenteni? Vegyük a 0 és 1 között egyenletes eloszlású u véletlen változót kontroll paraméternek! Számítsuk ki az u és az e^u közötti kovarianciát:

$$Cov(X, Y) = E(\{X - E(X)\} \cdot \{Y - E(Y)\}) = \quad (97)$$

$$= E(XY - E(X)Y - E(Y)X + E(X)E(Y)) = \quad (98)$$

$$= E(XY) - E(X)E(Y) - E(Y)E(X) + E\{E(X)E(Y)\} = \quad (99)$$

$$= E(XY) - 2E(X)E(Y) + E(X)E(Y) = \quad (100)$$

$$= E(XY) - E(X)E(Y) \quad (101)$$

Kihasználva ezt az egyszerűsítést a kovariancia számításnál:

$$Cov(e^u, u) = E(u \cdot e^u) - E(u) \cdot E(e^u) = \quad (102)$$

$$= \int_0^1 x e^x dx - \frac{1}{2} \cdot (e - 1) = \quad (103)$$

$$= \left([e^x \cdot x]_0^1 - \int_0^1 e^x dx \right) - \frac{1}{2} \cdot (e - 1) = \quad (104)$$

$$= ((e^1 \cdot 1 - e^0 \cdot 0) - (e^1 - e^0)) - \frac{1}{2} \cdot (e - 1) = \quad (105)$$

$$= 1 - \frac{e - 1}{2} = 0,14086 \quad (106)$$

A legjobb c ahogy az előzőekben láttuk:

$$c_1 = -\frac{\text{Cov}(X, Y)}{\text{Var}(Y)} = -\frac{\text{Cov}(e^u, u)}{\text{Var}(u)} \quad (107)$$

Ekkor az új varianciára (az egyenletes eloszlású u 0,5-ös várható értékét és 1/12-es szórásnégyzetét behelyettesítve) kapjuk, hogy:

$$\text{Var}(e^u + c_1(u - 0,5)) = \text{Var}(e^u) - \frac{\text{Cov}^2(e^u, u)}{\text{Var}(u)} = \quad (108)$$

$$= 0,2420 - \frac{0,14086^2}{(1/12)} = 0,0039 \quad (109)$$

Tehát a kontroll változó módszert használva 0,242-ről 0,0039-re, azaz 1,6%-ára csökkent le a variancia [38].

6.5. Rétegzett mintavétel (stratified sampling)

Rétegzett (más néven még: rétegezett) mintavételezést akkor használhatunk, ha a sokaság inhomogén (rétegekre bontható) és véges. Ezt a fajta variancia csökkentő megoldást akkor érdemes használni, ha tudjuk a rétegek egymáshoz viszonyított arányát, azaz hogyha egy véletlen darabot emelünk ki a sokaságból, akkor mekkora valószínűséggel lesz ez az első, második, stb. rétegből (csoportból) való. A rétegzett mintavételezés úgy történik, hogy minden csoportból veszünk mintát (rétegen belül ez egy véletlenszerű mintavételezés), így minden réteg képviselve lesz.

Szimulációnál az a sokaság, amiből a mintát kell venni, rendelkezésre állhat már rögtön a futtatás megkezdése előtt, de lehet, hogy ez a sokaság csak a szimuláció alatt gyűlik össze. Így a rétegenkénti minta alatt érthetjük a

- nulla időpontban egyszerre rendelkezésre álló adatok részhalmazát, vagy a
- futás alatt összegyűjtött adathalmazból származó mintát.

A rétegzett mintavételen belül is több megoldás létezik, egy adott feladatban a rendelkezésre álló információktól függ, hogy mikor melyiket érdemes használni:

a) Egyenletes elosztású rétegzett mintavételezés

Ennek lényege, hogy minden egyes rétegből azonos számú mintát veszünk: Itt az egyes rétegek mintavételi hibáinak összege minimális [13]. Ezt akkor érdemes használni, ha nem csak a főátlagra vagyunk kíváncsiak, hanem érdekelnek bennünket az egyes rétegek mutatói is.

b) Arányos elosztás

Az arányos elosztásnál a rétegek valószínűségeinek arányában választjuk meg, hogy melyikből mekkora mintát vegyünk.

$$n_j = n \frac{N_j}{N} = n \cdot p_j \quad (110)$$

ahol n_j a j -edik réteg mintavételi száma, n a teljes mintanagyság, N_j a réteg sokaságának számossága, N a teljes populáció száma, p_j annak a valószínűsége, hogy egy adott mintaelem a j -edik rétegből származik. Az arányos eloszlás előnye, hogy ennél a főátlag hibája csak kisebb, vagy egyenlő lehet, mint az egyszerű véletlen mintavételes (azaz a nem rétegzett) esetben. Ha a rétegenkénti szórásokat nem ismerjük (vagy ismerjük és azonosak), akkor a variancia csökkentő hatás ennél az arányos eloszlásnál maximális (azaz a módszer optimális megoldást nyújt).

c) Neyman féle optimális elosztás

Ha előre tudjuk a rétegenkénti szórásokat, akkor mindenképpen ezt a módszert érdemes alkalmazni, mert minimális varianciájú lesz az átlag (optimális megoldást kapunk).

$$n_j = n \frac{N_j \sigma_j}{\sum_{j=1}^k N_j \sigma_j} \quad (111)$$

ahol σ_j a j -edik réteg szórását jelöli, k pedig a rétegek száma.

Még ha semmilyen előzetes információnk nincs a rétegek szórásáról, akkor is lehet a módszert egy előzetes művelet segítségével alkalmazni. Ez az előzetes feladat: egy kis elemszámú elemet tartalmazó mintavételezés mindegyik rétegre vonatkozóan, mellyel meg lehet becsülni a rétegek szórását, amit felhasználva már meg lehet állapítani az optimális megoldást. Az előzetes mintavételezésnek viszont az a hátránya, hogy nehéz ehhez a feladathoz kitalálni a mintavételi nagyságot.

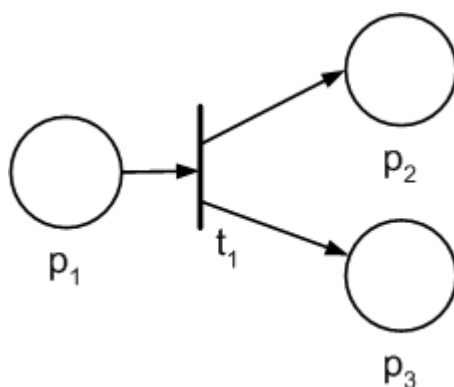
7. Modellezési lehetőségek

A legtöbb helyen a jelenség tanulmányozása olyan vizsgálattal történik, ahol nem is a valódi jelenséget vizsgálják, hanem inkább annak modelljét. A modell - gyakran matematikai formulákkal - az objektum tanulmányozása során fontosnak tartott jellemzőkből áll össze. A modell változtatásával új ismereteket kaphatunk a modellezett jelenségről, és ezt költség nélkül, mindenféle kényelmetlenséget elkerülve, a valós jelenségbe avatkozása veszélye nélkül kapjuk. Ilyen például az atomenergia, mert a radioaktív anyag kezelése drága és veszélyes. Legtöbb modell a matematika eszköztárát használja. A jelenségek fontos jellemzői numerikusan leírhatók, és az ezek közti kapcsolatok kifejezhetők egyenlet vagy egyenlőtlenség formájában. Például fizikában a jellemzők, mint tömeg, momentum, gyorsulás, helyzet, erő leírhatók matematikai egyenletekkel. A sikeres modellezés érdekében szükség van mind a modellezett jelenség, mind a modellezési technikák ismeretére, ez utóbbiak közül részletesebben most a Petri hálókkal ismerkedünk meg.

7.1. Petri hálók

Következőekben a szimulációban jól használható non-procedurális modellezési koncepcióról, a Petri hálóról lesz szó, melyet C. A. Petri vezetett be disszertációjában [34]. A Petri háló az információs folyam absztrakt és formális modellje [33]. A Petri hálók tulajdonságait, alapfogalmait és technikáját vezérlő és információs folyamatok leírására és analízisére fejlesztették ki, főleg olyan rendszerekben, amely aszinkron és konkurens tevékenységeket tartalmaznak. A Petri hálók fő használati területe események olyan rendszerének a modellezése, amelyben néhány esemény előfordulása egyszerre történhet, de vannak kényszerek az előfordulások frekvenciájára, előidejűségére, egyidejűségére vonatkozóan, így alkalmas szimulációs célokra is [51].

A Petri háló egy speciális irányított gráf, amelyre az alábbi képen (lásd 4. ábra) láthatunk egy példát. Ebben a példában a Petri háló gráfként van ábrázolva, a gráf a rendszer statikus tulajdonságait modellezi úgy, mint ahogy a folyamatára reprezentálja a számítógépes program statikus jellemzőit.



4. ábra Petri háló grafikus képe

A gráf 2 fajta csomópontot tartalmaz: **place**-eket (magyarul helynek fordítható, melyeket körökkel ábrázolunk) és **transition**-eket (azaz átmenetek, melyeket rudak jelképeznek). Ezek

a csomópontok, place-ek és transition-ök, éleken keresztül kapcsolódnak egymáshoz. Ha egy él az i -től a j csomóponttra mutat (mindegy, hogy place-től transition-ra vagy fordítva), akkor i inputja (bemenete) j -nek, és j outputja (kimenete) i -nek. A 4. ábra például azt mutatja, hogy a p_1 place a t_1 transition bemenete, míg a p_2 és p_3 place a t_1 transition kimenete.

A statikus összeállításra van egy megkötés: a két fajta csomópont csak felváltva követheti egymást, azaz place csak transition-höz kapcsolódhat és viszont. Azon kívül, hogy a gráf a statikus jellemzőket ábrázolja, a Petri hálónak van dinamikus komponense is, amely a futtatásából származik. Tételezzük fel, hogy a folyamatábrával megjelenített számítógépes program futtatásának nyomkövetésére egy zsetont használnak, azaz a folyamatábrában elhelyezett zseton mutatja, hogy a futtatás hol tart, és ahogy a futtatás halad, a zseton mozog a folyamatábrában. Hasonlóan, a Petri háló futtatása a **tokeneknek** nevezett zsetonok helyzetváltoztatásával történik. Tokenek, melyeket grafikusán fekete ponttal jelölnek, a körökkel ábrázolt place-ekben tartózkodhatnak. A tokenekkel ellátott Petri hálót **jelzett Petri hálónak** nevezik.

Tokenek használatára a szabályok a következők: a tokenek helyváltoztatási képessége a transition-ök **tüzelésével** valósul meg. A transition-nak engedélyezett állapotban kell lennie ahhoz, hogy tüzelhessen. (A transition akkor engedélyezett, ha az összes bemenetén levő place-ben van legalább annyi token, mint amennyi a kötés multiplicitása. A részletes magyarázatot lásd később.) A transition tüzeléskor a bemeneti place-ekben levő tokenek eltávolítását (letörlését) és új tokenek generálását végzi, melyeket a transition kimeneti place-eiben helyez el.

Petri hálók formális leírása

A Petri hálókat nem csak grafikusán tudjuk ábrázolni, hanem matematikai leírási módja is létezik. Ez az ún. formális leírás a következőképpen néz ki:

$$C = \{P, T, I, O\} \quad (112)$$

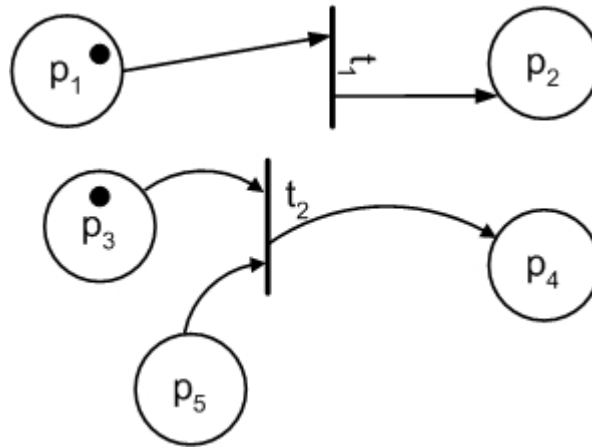
Egy Petri háló a fenti négyes, jelzett Petri háló pedig az alábbi ötös segítségével írható le:

$$M = \{P, T, I, O, \mu\} \quad (113)$$

ahol P a Petri hálót alkotó place-ek halmaza, T a transition-ök halmaza, I a bemeneti (input) függvény, azaz egy olyan leképezés, mely a transition-ök bemeneti összeköttetéseit írja le a place-ek kimeneteivel, O a kimeneti (output) függvény, azaz egy olyan leképezés, mely a transition-ök kimeneti összeköttetéseit írja le a place-ek bemeneteivel, μ pedig a tokenek eloszlását írja le a place-ekben. Például az 5. ábra által mutatott Petri háló formális leírása a következő:

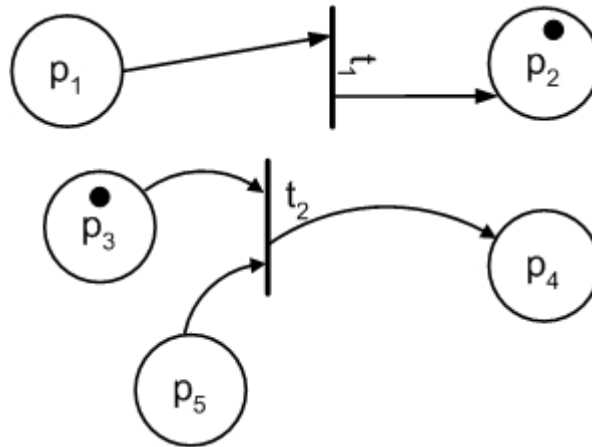
$$\begin{aligned} M &= \{P, T, I, O, \mu\} \\ P &= \{p_1, p_2, p_3, p_4, p_5\} & T &= \{t_1, t_2\} \\ I(t_1) &= \{p_1\} & O(t_1) &= \{p_2\} \\ I(t_2) &= \{p_3, p_5\} & O(t_2) &= \{p_4\} \\ \mu &= \{1, 0, 1, 0, 0\} \end{aligned}$$

Ha valamelyik él többszörös él (azaz multiplicitása nagyobb 1-nél), akkor a halmazbeli felsorolásnál is annyszor kell felsorolni, amennyi a multiplicitása.



5. ábra Petri háló tokenekkel

A 5. ábra egy jelzett Petri hálót ábrázol, ahol a t_1 engedélyezett, mivel a bemeneti place-jében (p_1) van token. A t_2 transition nem engedélyezett, mivel az egyik bemenetén levő place-ben (p_5) nincs token. Ha t_1 tüzel, a Petri háló eredménye a következő, 6. ábrán lesz látható, a t_1 transition tüzelésével letörli a p_1 place-ből az elvihető tokent, és a p_2 -be egy új tokent rak.



6. ábra Egy transition tüzelése utáni állapot

Tokenek eloszlása a jelzett Petri hálókban definiálja a háló állapotát, ezt a tokeneloszlást hívják **marking**-nak. A marking a transition-ök tüzelésével változhat (pontosabban: a markingok változását a transition-ök tüzelése okozza). Különböző marking különböző transition-öket engedélyezhet.

Nézzük meg általánosan, matematikai formában a tüzelési feltételeket és műveleteket, melyek a Petri háló dinamikus működését adják. Egy transition tehát akkor tüzel, ha engedélyezett állapotban van, ami akkor következik be, ha valamennyi bemeneti place $p_i \in P$ esetében

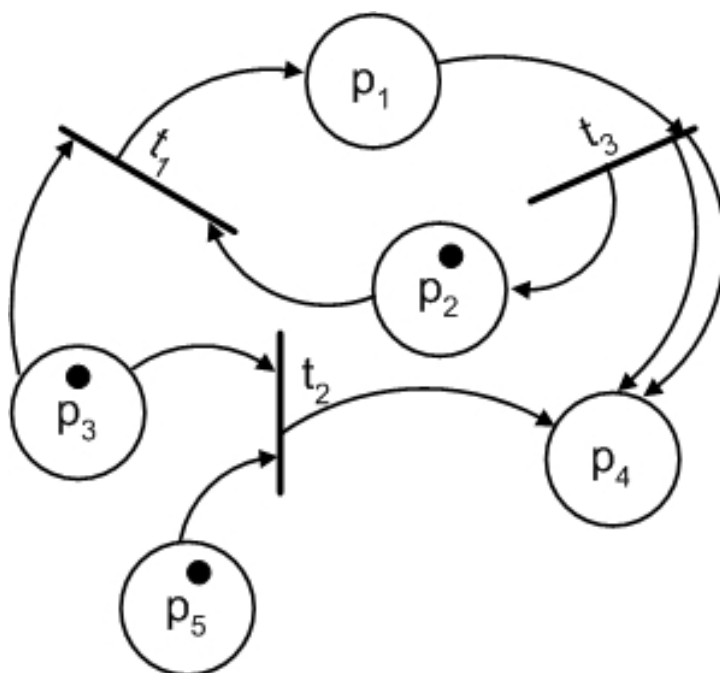
$$\mu(p_i) \geq \#(p_i, I(t_j)) \quad (114)$$

ahol $\#(p_i, I(t_j))$ adja meg a place elem és a t_j transition közötti multiplicitását, azaz a p_i -ből t_j -be mutató ágak számát, μ pedig a megadott place-ben levő tokenek számát. Amennyiben a t_j

transition tüzel, úgy megváltoztatja a Petri háló markingját, azaz a p_i place-ben levő tokenek számát μ -ról μ' értékre.

$$\mu'(p_i) = \mu(p_i) - \#(p_i, I(t_j)) + \#(p_i, O(t_j)) \quad (115)$$

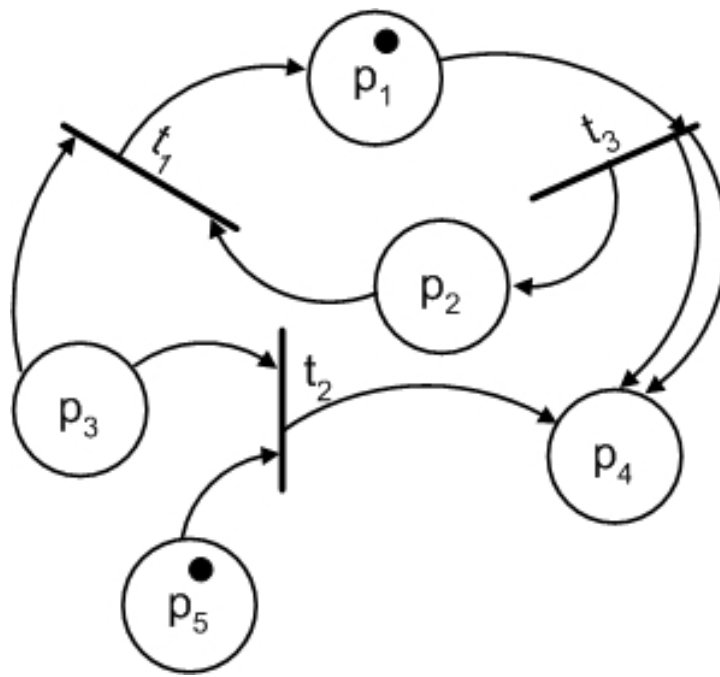
ahol $\#(p_i, O(t_j))$ jelenti az output multiplicitást. Ez lényegében azt jelenti, hogy valamennyi input place-ben megsemmisít és valamennyi output place-ben létrehoz egy tokenet.



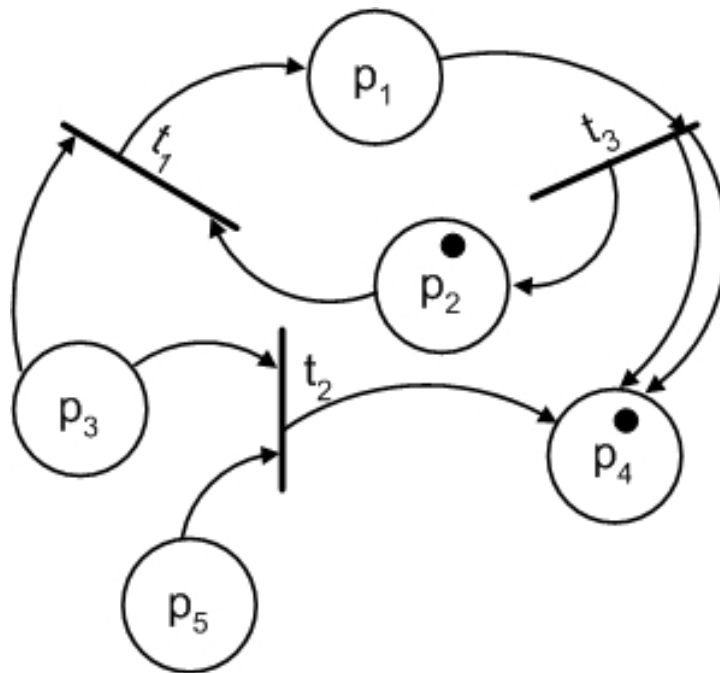
7. ábra Példa egy konfliktus helyzetre

A 7. ábra egy érdekes jelenségű Petri hálót mutat. Figyeljük meg, hogy a Petri hálóban 2 transition is engedélyezett: a t_1 és t_2 transition, ebben a helyzetben nincs meghatározva, hogy melyik transition tüzeljen legközelebb. A 8. ábra mutatja a 2 lehetséges származtatott markingot; melyekből további markingok elérhetők (jelen esetben csak a felső ábrán), mivel transition tüzelései addig folytathatók, míg van engedélyezett transition.

Ha a t_1 transition (7. ábra) tüzel, akkor ennek eredményeként a p_1 place-ben 1 token lesz (lásd 8. a) ábrát), és a p_3 -ban törlődik a token, így t_2 transition tüzelési feltétele megszűnik (ezen kívül a p_2 -ben is törlődik a token). Ha a t_2 transition tüzel, akkor ennek hatásaként a p_4 place-ben lesz 1 token (lásd 8.b) ábra), és t_1 transition nem lesz engedélyezve (mivel a p_5 place-ben levő tokenen kívül a p_3 -ban levő token is törlődik). A t_1 és t_2 transition közül bármelyik tüzelése letiltja a másikat, úgy mondják: **konfliktusban** vannak.



a)

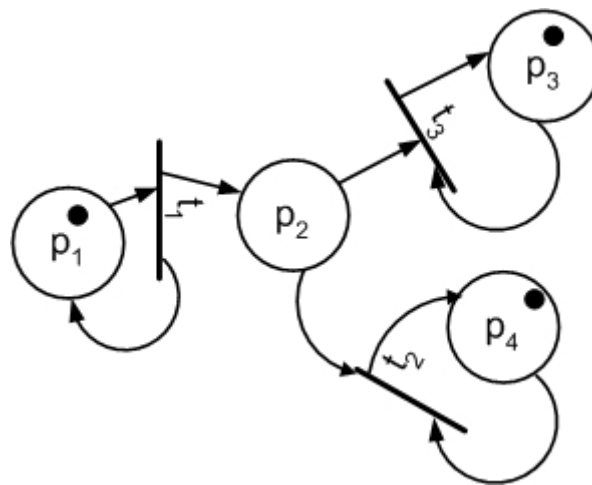


b)

8. ábra Két lehetséges származtatott marking

A Petri hálók gyakorlati alkalmazására jó példa a következő, 9. ábrán látható termelő-fogyasztói probléma, mely egy termelőt (p_1 és t_1 -ből álló rész) és két fogyasztót (p_3 és t_3 valamint p_4 és t_2 -ből álló részek) tartalmaz. A termelő által termelt darabokat a fogyasztóknak juttatják el a p_2 -es place-en keresztül, mely itt a piacot reprezentálja.

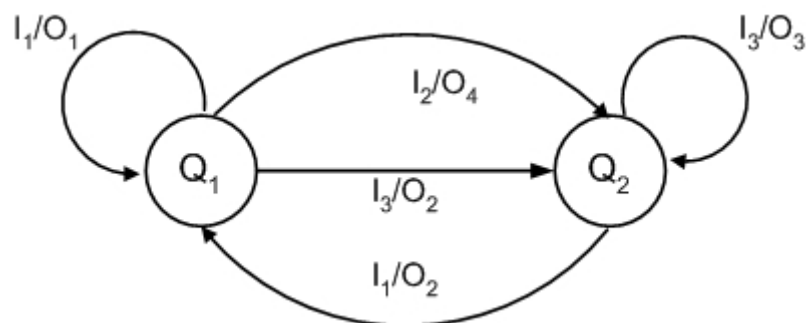
A 9. ábra által mutatott hálózatban az érdekelhet bennünket, hogy a termelő mennyivel jár a fogyasztók előtt, leghagyhatja-e a termelő a fogyasztókat, vagy elhasználható-e ugyanaz a darab kétszer, és így tovább. Ez a termelő-fogyasztói modell segít a kérdések megválaszolásában. Petri hálók alkalmazásainak köre széles, több helyen alkalmaztak Petri-hálós technikát ütemezési feladatokra egy gyáregységen belül is és logisztikai egységek között is [1].



9. ábra Példa termelő-fogyasztói problémára

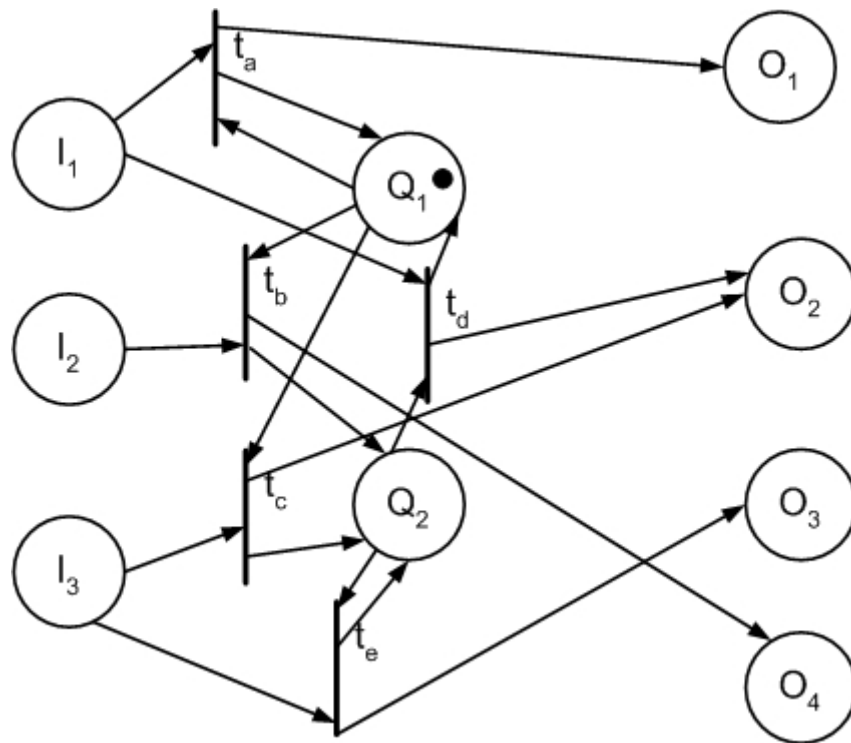
A fenti példánál még általánosabb csoportot, a véges automatákat is lehet modellezni. Véges állapotú gépek (véges automata) állapot átmeneti gráfja ekvivalens módon ábrázolható egy speciális Petri hálóval, ahol:

- Egy állapot egy place-nek felel meg.
- Egy állapotátmenetnek egy tüzelés felel meg.
- A tüzelés feltételei megegyeznek az állapotátmenet állapot gráfbeli feltételeivel.
- Ez az ekvivalens Petri háló 1 tokenet keringet.
- A kezdeti marking olyan, hogy a kezdőállapotnak megfelelő place-ben helyezkedik el az egyetlen token.



10. ábra Példa véges automatára

Nézzünk egy példát véges automatára (lásd 10. ábra), ahol 2 állapot és 5 állapotátmenet lett definiálva. Az automata 1 bemenettel és 1 kimenettel rendelkezik, a kimenetet és a következő állapotot a bemenet és a jelenlegi állapot határozza meg. Az állapotátmenetekre azok a bemeneti értékek (I_1, I_2, I_3 közül egy) és azok a kimeneti értékek (O_1, O_2, O_3, O_4 közül egy) lettek felírva, amelyek meghatározzák az adott átmenetet. Ha a két állapotnak megfeleltetünk egy-egy place-t, és az állapot átmeneteknek 5 transition-t, akkor ebből felépíthető a rendszer Petri hálós modellje. Ábrázoljuk még azonban a be- és kimenetek lehetséges értékeit is place-ekkel, hogy a véges automata teljes rendszerét modellezni lehessen. A megvalósult Petri hálót a következő (11. ábra) ábrán láthatjuk.



11. ábra Véges automata modellezése Petri hálóval

A felépített Petri hálókat különböző módon vizsgálhatjuk, analizálhatjuk őket, különböző kérdéseket tehetünk fel. Például hogyan jellemezhetjük egy adott állapotból elérhető markingok osztályát? Milyen érdekes jellemzői vannak a Petri hálóknak és hogyan tesztelhetjük ezeket?

A Petri hálók analízisére sokféle célból lehet szükség, például elosztott multitasking rendszerekben az egyik leggyakoribb probléma a holthelyzet (deadlock) mentesség vizsgálata. Egy Petri háló élő, ha minden transition újra tüzelhetővé válik a későbbi működés során. Az élő Petri háló garantálja a holthelyzet (deadlock) mentes működést függetlenül a bejárasi úttól. Nézzük meg azonban az általános analízishez szükséges egyéb tulajdonságokat is. A Petri hálók analízisének két vizsgálódási terület van: a strukturális és a kezdőállapot függő analízis, így az említett tulajdonságok ebbe a két csoportba sorolhatók [32].

Kezdőállapot függő vizsgálat tulajdonságai:

- Elérhetőség
- Korlátosság
- Élő tulajdonság

- Megfordíthatóság
- Perzisztens

Strukturális tulajdonságok:

- Strukturális korlátosság
- Strukturálisan élő
- Vezérelhető
- Konzervatív
- Konzisztens

§ **Definíció. Elérhetőség:** Egy adott (kezdő)állapotból elérhetőnek mondjuk egy másik állapotot (állapotok halmazát) ha találunk olyan tüzelési szekvenciát, amely segítségével ebbe a másik állapotba jutunk.

§ **Definíció. Korlátosság:** Ha a kezdeti markingból kiindulva a teljes elérhetőségi fán belül mindegyik place-ben maximum k db token található, akkor a Petri hálót **k-korlátosnak** nevezzük. Az 1-korlátos háló külön nevet kapott: **biztonságos** Petri háló.

§ **Definíció. Élő tulajdonság:** ha a kezdeti állapotból kiindulva valamennyi transition tüzelése végrehajtható, azaz újra tüzelhetővé válik a későbbi működés során. A transition-re bevezettek különböző szintű élő tulajdonságokat. Egy tr transition

- L_0 élő, azaz halott, ha tr soha sem tüzelhető kezdeti markinggal megadott Petri hálóban.
- L_1 élő, ha tr legalább egyszer tüzelhető valamely tüzelési szekvenciában.
- L_2 élő, ha tetszőleges pozitív k egészre k -szor tüzelhető valamely tüzelési szekvenciában.
- L_3 élő, ha tr tüzelése végtelen sokszor előfordul valamely tüzelési szekvenciában.
- L_4 élő, ha tetszőleges kezdeti markingból elérhető állapotok esetén a tr L_1 élő.

Egy Petri háló L_k élő, ha a benne szereplő minden transition L_k élő.

§ **Definíció. Megfordíthatóság:** Egy Petri hálót megfordíthatónak nevezünk, ha valamilyen tüzelési szekvencia segítségével visszajutunk a kiinduló állapotba.

§ **Definíció. Perzisztens:** a Petri háló, ha két tetszőleges engedélyezett transition közül az egyik sem tiltja a másikat, azaz nincs konfliktus helyzet a transition-ök között.

§ **Definíció. Strukturális korlátosság:** ha a Petri háló tetszőleges véges kezdőállapot esetén korlátos.

§ **Definíció. Strukturálisan élő:** ha van egy élő kezdeti marking.

§ **Definíció. Vezérelhető:** ha tetszőleges állapot elérhető tetszőleges más állapotból.

§ **Definíció. Konzervatív (P invariancia):** Ha található olyan súlyelosztás a place-ekhez, hogy a tokenek súlyozott összege állandó a működés folyamán, akkor a Petri hálót **konzervatív**nak nevezzük. Ezt a súlyelosztást **P-invariáns**nak nevezzük. Ha a súlyelosztás speciálisan olyan, hogy minden place-hez 1-et rendel, akkor **szigorúan konzervatív** Petri hálóról beszélünk. Ez utóbbi esetben tehát a tokenek száma állandó.

§ Definíció. Konzisztens (*T invariancia*): Ha van olyan kezdeti marking és olyan tüzelési szekvencia, amely úgy vezet vissza a kezdőállapotba, hogy minden egyes tranzíció tüzelése legalább egyszer előfordul ebben a szekvenciában. Ezt a tüzelési szekvenciát nevezzük **T-invariánsnak** [32].

A fenti tulajdonságok szerint ellenőrizhetők, analizálhatók a felépített Petri hálók, amelyek így nem csak a modelltől, hanem a modellezett rendszertől is állítanak valamit. A Petri hálók (mint absztrakt formális módszer [9]) továbbfejlesztésére sok irányvonal létezik, a különböző magasszintű Petri hálók [19] között megtaláljuk a színezett Petri hálótól kezdve a szofisztikált modellezési lehetőséggel bíró Tudás Attribútumú Petri Hálót [18].

7.2. Markov láncok

A Markov láncok olyan rendszerek, amelyek állapota függ a jelenlegi állapotuktól, viszont független a régi állapotuktól (előéletüktől). Az ilyen állapotok alakulását leíró folyamatok leírására szolgáló Markov-láncok tehát memóriával nem rendelkező rendszerek.

A pontos definícióhoz tekintsük egy olyan időben véletlenszerű állapotváltozásokat végző rendszert, amelynek lehetséges állapotai: s_1, s_2, s_3, \dots , stb. A rendszer állapotait $t = 0, 1, 2, \dots$ időpontokban megfigyelve definiáljuk a $\zeta_0, \zeta_1, \zeta_2, \dots$ valószínűségi változókat olyan módon, hogy a $\zeta_n = i$ értéket vesz fel, ha a rendszer $t = n$ időpontban éppen s_i állapotban van.

§ Definíció. Markov-láncnak nevezzük a rendszer állapotváltozásainak sorozatát, vagy – az ezzel egyenértékű – $\zeta_0, \zeta_1, \zeta_2, \dots$ valószínűségi változók sorozatát, ha tetszőleges egész $t_1 < t_2 < \dots < t_{n+1}$ időpontok és bármely állapot (időrendben felvett állapotok sorszámait: j_1, j_2, \dots, j_{n+1} -el jelölve) esetén fennáll a következő egyenlőség:

$$P(\zeta_{n+1} = j_{n+1} | \zeta_1 = j_1, \zeta_2 = j_2, \dots, \zeta_n = j_n) = P(\zeta_{n+1} = j_{n+1} | \zeta_n = j_n) \quad (116)$$

Szavakba öntve ez azt jelenti, hogy ilyen típusú rendszerek t_n időpont utáni felvett állapotainak valószínűségeit (röviden további történetét) egyértelműen meghatározza a rendszer t_n időpontban felvett állapota, és a rendszer korábbi állapotaira vonatkozó semmiféle információ – azaz, hogy hogyan került a rendszer ebbe az állapotba – nem szükséges [16].

Markov-láncok jellemzésére szolgálnak az egyes állapotok közötti **átmenetvalószínűségek** (állapot-átmeneti valószínűségek), melyek megadják hogy egy adott i állapotból kiindulva mekkora valószínűséggel lesz a rendszer a következő lépésben a k állapotban. Ezeket – általánosítva – mint r lépéses átmenetvalószínűség-értékeket az alábbi módon adhatjuk meg:

$${}_n P_{ik}^{(r)} = P(\zeta_{n+r} = k, \zeta_n = i) \quad (117)$$

Az átmenetvalószínűségeket szokás az ún. átmenetvalószínűség mátrixban összefoglalva felírni:

$${}_n \Pi_r = \begin{bmatrix} {}_n P_{11}^{(r)} & {}_n P_{12}^{(r)} & \dots \\ {}_n P_{21}^{(r)} & {}_n P_{22}^{(r)} & \dots \\ \dots & \dots & \dots \end{bmatrix} \quad (118)$$

A mátrix elemeit alkotó valószínűségek összege soronként mindig 1-et kell hogy adjon. Ugyanis r lépés múlva valahol (valamelyik állapotban) biztos, hogy megtalálható a rendszer, így ezeknek a valószínűségeknek az összege a biztos esemény valószínűségével egyezik meg.

Homogén Markov-láncokról beszélünk, ha valamennyi $n P_{ik}^{(r)}$ érték független n -től, így ez a bal alsó index elhagyható:

$$\Pi_r = \begin{bmatrix} P_{11}^{(r)} & P_{12}^{(r)} & \dots \\ P_{21}^{(r)} & P_{22}^{(r)} & \dots \\ \dots & \dots & \dots \end{bmatrix} \quad (119)$$

Homogén Markov-láncok esetén az r lépéses átmenetvalószínűségek mátrixa egyenlő az egylépéses átmenetvalószínűség-mátrix r -edik hatványával [11][36]:

$$\Pi_r = \Pi_1^r \quad (120)$$

A Markov-láncok jól használhatók a szimulációban különböző rendszerek vizsgálati modelljeként [8].

7.3. Diszkrét szimulációs eszközök

Ebben az alfejezetben a szimuláció szoftver eszközeit mutatjuk be, hogy erre alapozva a következő teljes fejezetet az alkalmazásoknak szentelhesük különböző példákon keresztül.

A szimulációs eszközök alapvetően három csoportra oszthatók [16].

1) **Szimulációs programnyelvek:** olyan magasszintű programnyelvek, melyek speciális tulajdonságaikkal elősegítik a szimulációs modellek felépítését és futtatását. Ezek általában a szimulációs problémák rendkívül széles körét fedik le, és az adott partikuláris szakterület terminológiája helyett általános szimulációs fogalmakkal dolgoznak. Hátrányuk, hogy egy modell felépítéséhez programírási ismeret is szükséges, azaz a modellezővel szemben van egy csekély programozás-technikai elvárás. (Ebbe a kategóriába tartozik például a GPSS, SIMSCRIPT, SIMAN, ACSL.)

2) **Szimulációs rendszerek:** olyan szoftver eszközök, melyek egy-egy szűkebb terület szimulációs problémáit a szakterület terminológiájának megfelelően írják le, ezáltal az adott problémakörre vonatkozólag igen kényelmesen használható eszközt jelentenek, azonban az alkalmazási kör ennek megfelelően lényegesen szűkebb. (Ilyen például a ProModel.)

3) **Általános célú programnyelvek** (C, C++, Pascal, Delphi, Java, stb.), melyeket egyes egyedi feladatokra ma is alkalmaznak (hiszen ha valaki ismeri és korábbról már rendelkezésére áll a szoftver, akkor nem kell energiát fordítani egy újnak a megismerésére). Kis feladatoknál általában nem jelent túl nagy problémát a szimulációs program megírása, azonban komplexebb probléma esetén érdemes az előző kettő csoport valamelyikéből választani eszközt.

Az első két csoport összehasonlításánál megállapíthatjuk, hogy a speciális célra kifejlesztett eszközök (2. csoport) általában felhasználó barátok, azaz a vizsgálatot végző személy számára az adott szakterületen alkalmazott fogalomrendszer alapján teszik lehetővé a kommunikációt,

azonban ezek az eszközök csak szűk területen alkalmazhatók. Az általános célra kifejlesztett eszközök (1. csoport) ugyan igen széles körben alkalmazhatók, azonban a kommunikációs réteg nem annyira testre szabott az egyes feladatok megoldásánál.

Fontos szempont lehet az általánosság ill. specializáltság ellentétes megközelítéséből adódó ellentmondás feloldása. A CASSANDRA [15] nevű szimulációs eszköz például ezt az ellentmondást oly módon oldja fel, hogy a szimulációs eszközrendszer egy rendkívül széles körben alkalmazható belső kernelre épül fel, melybe a különböző típusú feladatok (a modell és vizsgálati feltételek leírásai) leképződnek. További lényeges szempontot jelent, hogy ezek a leképezések hierarchikus módon is végezhetők. Ez azt jelenti, hogy a nagyobb modellek elemibb modell elem struktúrákra képződnek le, és így a rendszer vizsgálatot végző mindig az általa igényelt részletességgel „látja” a modellt.

8. Szimulációs alkalmazások

Ahogy az előző fejezetben láttuk: a szimulációs szoftverek egy része általános problémákra kínál modellezési megoldást szimulációs nyelvek alkalmazásával; másik részük szakterületekre koncentrál, és csak az adott felhasználási területen nyújt megoldási lehetőségeket. Az alkalmazási területeket számba véve a szimuláció használható a közlekedési rendszerektől elkezdve, a gazdasági modelleken és gyártórendszereken át a környezetszennyezési modellekig. Mindegyiknél több olyan szimulációs eszköz is készült, amely az adott szakterületen használható. Az alkalmazási területekből önkényesen válogatva jelen fejezetben a rugalmas gyártórendszereket és a közlekedési rendszereket mutatjuk be.

8.1. Gyártósor és logisztikai rendszer szimulációja ProModellel

Különböző problémák nemcsak egyszer jelentkeznek a vállalatnál, hanem folyamatosan merülnek fel újabb megoldandó feladatok, és ezek megoldásaként újításokat kell bevezetni a fejlődés érdekében, és egyre komplexebb problémákat kell megoldania a vállalati irányításnak. Egy operációkutatási feladatnál nagyon fontos, hogy a gyártani kívánt termékekhez szükséges anyagok, alkatrészecskék, részegységek mennyiségi és megrendelési adatait meghatározzuk, a munkafolyamatok kezdő- és befejező időpontjait megtervezzük adott feltételek mellett. E feltételek közé tartozik a gyártási határidők betartása, a felhasználható anyagok, erőforrások rendelkezésre állása, ill. hiánya, adott raktári kapacitás, stb. A cél, hogy ezeket a feladatokat rövid átfutási idővel, alacsony raktári készletekkel, nagyfokú rugalmassággal, a legjobb minőségben és a legolcsóbban oldjuk meg. [35][43]

A gyártás manapság már legtöbbször rugalmas gyártórendszeren folyik, melyek magukban foglalják a megmunkáló, raktározó, anyagmozgató, minőségellenőrző és a gyártáshoz kapcsolódó segédfolyamatokat. Ezekben a (FMS – Flexible Manufacturing System) rendszerekben a gyártóberendezések különböző munkadarabokon különböző megmunkálási feladatokat tudnak végrehajtani anélkül, hogy a folyamatosság az átállás miatt megszakadna. Ezt nagyfokú automatizálással lehet csak elérni, de még fejlettebb rendszert kapunk, ha a számítógépeket is bevonjuk a termelési és logisztikai folyamatokba. A számítógéppel integrált gyártási és logisztikai rendszerek (CIM–CİL) az egyik legkorszerűbb megoldást foglalják magukba, ugyanis itt egy rendszerbe kapcsolódik össze az ellátás, a gyártás és az elosztás, és ebben a komplex rendszerben együtt kezelik az anyagot és az információt.

A ProModel gyártás-szimulációs eszköz

Gyártórendszereknél a sztochasztikus tulajdonságokkal is rendelkező szimuláció lehetővé teszi, hogy a gyártási folyamatok idejét különböző eloszlások alapján modellezzük, és a véletlen események (mint például robotok, gépek működésképtelenné válása, sztrájk vagy egyéb előre nem látható esemény) generálása szintén könnyen megvalósítható, melynek segítségével különböző ütemezési és vezérlési feladatok oldhatók meg. A gyártórendszerek körében a sokfajta termék egyidejű gyártására legjobban elterjedtek az FMS rendszerek, melyek olcsón és hatékonyan tudnak termelni. Ebben az alkalmazási körben a szimulációs eszközök sorában többek között megtaláljuk az angol WITNESS és a holland Taylor termékeket, illetve a továbbiakban részletesebben ismertetésre kerülő ProModelt. A ProModel

(PROduction MODELer) egy könnyen használható, személyi számítógépen futtatható diszkrét szimulációs szoftver, amely rendelkezik automatikus modellépítővel, saját nyelvi jegyekkel és beépített szabályokkal, alkalmas mindenféle típusú gyártórendszer modellezésére – a kis gyárüzemekről kezdve a tömeggyártásra alkalmas rugalmas gyártórendszerekig [12].

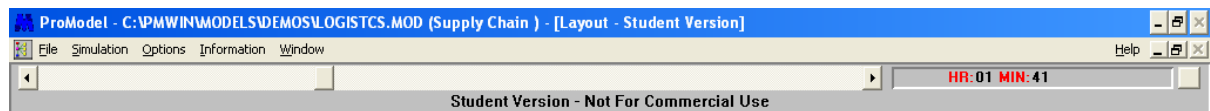
A logisztikai vizsgáldáshoz szükséges mobil elemeket, amelyek keresztül haladnak a rendszeren, a szimulációs szakirodalomban – így a ProModelben is – **entitásoknak** nevezzük. Az állandó elemek pedig lehetnek

- erőforrások, melyeket az entitások lefoglalhatnak és felszabadíthatnak,
- állomások, ahol különböző műveleteket végeznek az entitásokon, vagy
- szállítók, amelyek mozgatják azokat.

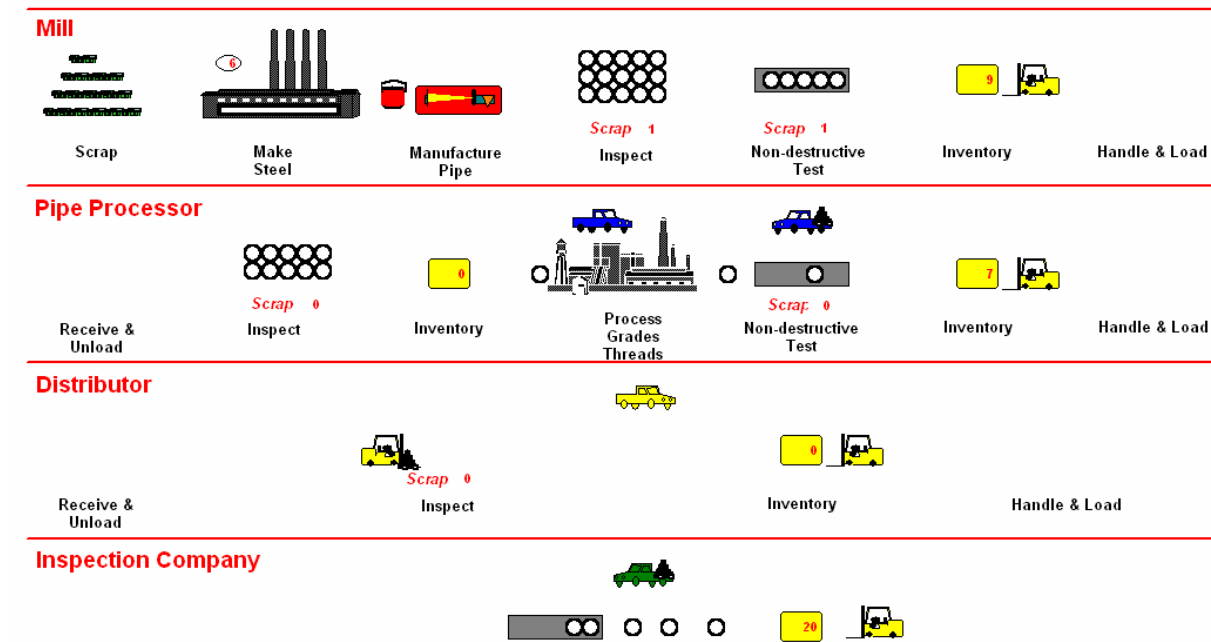
A modell felépítése ebben a szimulációs programban a különböző állomások (például gép, szállítószalag, raktár) munkatáblán való elhelyezésével kezdődik, majd definiálni kell az entitások folyamatát, azaz az állomások szekvenciáját (routing), ahol meg kell adni, hogy a különböző típusú elemek milyen állomásokon keresztül mozognak, és az egyes állomásokon mennyi időt töltenek. A routingnál használhatók olyan speciális műveletek, amelyek segítségével erőforrás-lefoglalás és különböző kötegelési feladatok oldhatók meg. Lehet bizonyos programutasításokat használni, mint például változók, attribútumok értékadása, feltételes elágazás, Pascal szubrutinok hívása. Megadhatjuk azt is, hogy az entitások melyik állomáson, milyen sűrűn, mekkora csoportokban érkezzenek a rendszerbe.

Egy logisztikai rendszer pontos modelljének felépítéséhez elengedhetetlen a szállítók valósághű leírása. Ennek tesz eleget a ProModel a szállítók sebességének, fel- és lerakodási idejének, indulási pontjának, gyorsulásának és lassulásának, kapacitásának definiálásával. Megadhatók a szállítók által használt hálózat pontjai a köztük levő távolságokkal. A váratlan helyzetekre felkészülés jelent igazán logisztikai kihívást, mert a váratlan események új, megoldatlan problémákat generálhatnak. A ProModelben definiálhatók véletlen események (mint például a logisztikai rendszer egyes elemeinek meghibásodása), így megadható az állomások váratlan meghibásodásainak jellemzői (például MTBF: Mean Time Between Failure, MTTR: Mean Time To Repair), leállás típusa, gyakorisága, első előfordulásának ideje, tartalék erőforrások száma stb. Az így elkészült modellek különböző módon futtathatók (például animációval vagy anélkül). Kötegelt szimulációs módban egymás után több forgatókönyv futtatható, egy forgatókönyvet pedig ugyanolyan paraméterekkel többször is lefuttathatunk, ez az ismétlési lehetőség biztosítja a véletlen elemeket is tartalmazó modell különböző kiértékelési eredményeit.

A 12. ábra a ProModel logisztikai problémamegoldására mutat példát [3], hogyan építhető fel egy gyártási-elosztási rendszer szimulációs modellje, ahol az adatok strukturális, operációs és szimulációs paraméterekből állnak. **Struktúrán** a rendszert alkotó statikus objektumok kapcsolati halmazát értjük. **Operációs adatok** közé tartoznak az áruk ki- és berakodási ideje a raktárból, illetve a raktárba, a raktárak közti szállítási idő, ütemezési adatok (gyakoriság, első előfordulás ideje, csomagnagyság) stb., tehát a munkafolyamatokhoz tartozó paraméterek. A **szimulációs paraméterek** pedig azok az adatok, melyek a szimuláció dinamikus futását befolyásolják a feltételek rögzítésével, például milyen hosszan történjen a futtatás, mikor kezdje el gyűjteni a futás során az eredményeket stb. Futtatás után megvannak mindazok a lehetőségek, amelyekről a korábbi fejezetekben szó esett.



Supply Chain - Tubulars



12. ábra Elosztási rendszer a ProModel szimulációs rendszerben

8.2. Szimuláció Mesterséges Intelligenciával (MI)

A mesterséges intelligencia [52][40] és a szimuláció kombinálása [41] azoknál a feladatoknál jelent egy magasabb minőséget, ahol valamilyen optimális pontot, ideális megoldást kell keresni. A legjobb megoldás felé vezető út egy olyan rekurzív, iteratív folyamat, amely ember-gép kapcsolaton alapul. A modell felépítését követően elvégezzük a szimulációt, ami a modell dinamikus működését, azaz az adott feltételek melletti viselkedését szolgáltatja. Ezt követően az eredmények kiértékelése valamint a modellen (megváltoztathatjuk a modell paramétereit és/vagy a struktúráját) és a vizsgálati feltételeken való módosítás következik, aminek célja, hogy a követelményeknek jobban megfelelő megoldást kapjunk. Ezt addig folytatjuk, amíg a kívánt célkitűzést elérjük. Ez a folyamat mesterséges intelligencia alkalmazásával rendkívüli módon felgyorsítható és hatékonyabbá tehető oly módon, hogy a dinamikus működés kiértékelését, a következtetések alapján a szükséges változtatásokat tudásbázissal rendelkező intelligens objektumokra bizzuk. Ez a megközelítés újdonságnak számít, de gyors megoldást biztosított már komplex problémák esetén, így vállalati flexibilis gyártó és minőségellenőrzési ill. városi közlekedési és környezetvédelmi feladatoknál, ahol a mesterséges intelligenciával rendelkező objektumok, ún. ágensek [15][17] alkalmazása vezetett a megoldáshoz. Ezeken a területeken a nagybonyolultságú modelleknél felmerült problémák által igényelt módszer, illetve eljárás a következőképpen általánosítható:

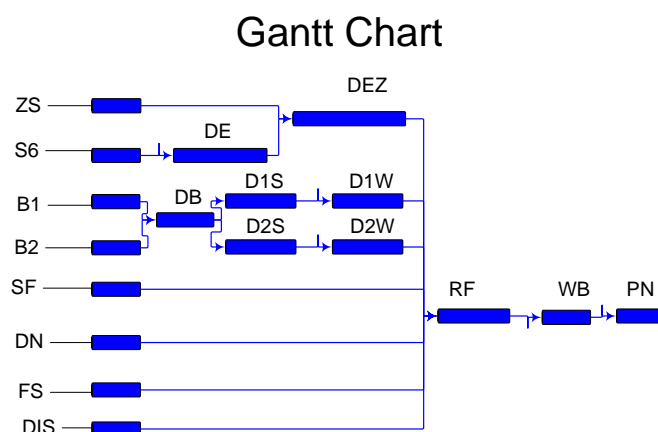
Első lépésként meg kell határozni az egyes alkalmazási területek modellstruktúráit és vizsgálandó problémaköreit az adott területek szakértőivel együttműködve. A problémát ismerő szakértők információja alapján fel kell építeni egy kiinduló modellt, majd a statisztikai

adatokon alapuló „historikus” adatsorokat felhasználva szimulálni kell a rendszert reprezentáló kiinduló modell működését. A mesterséges intelligenciával (tudásbázissal és következtetési eljárással) rendelkező ágensek folyamatosan monitorozzák a rendszer működési trajektóriáját és összevetik a historikus adatokkal. Amennyiben jelentős eltérést tapasztalnak, úgy módosítják a modell struktúráját, paramétereit ill. a modell objektumai közti kölcsönhatási mechanizmusokat (például gazdasági, műszaki, szociológiai, stb. tényezők lehetnek kölcsönhatásban egymással) egészen addig, amíg a múltira vonatkozó adatsorokkal elfogadható mértékű egyezést nem kapunk. Ezzel a rendszer-rekonstrukciós folyamattal a vizsgált rendszert kielégítő pontossággal visszatükröző modellhez jutunk. Ez a fázis jelenti azt a munkát, amely alapján a rendszer vizsgálatát a továbbiakban végezhetjük. A rendelkezésre álló elméletekből és „történelmi” idősorokból megalkotott adekvát modellek felépítése után már szimuláció segítségével megbízható előrejelzések kaphatók.

8.3. Optimalizálás szimulációval gyártósoroknál

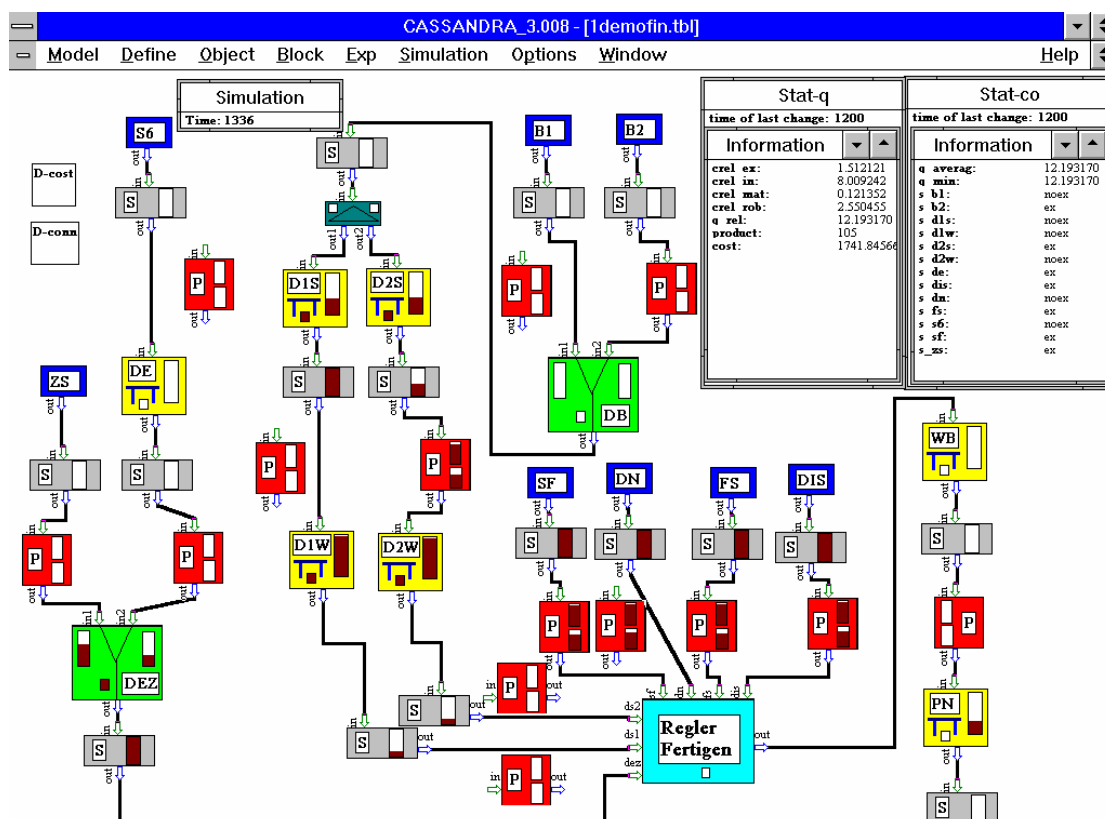
A szimulációs technika ott tud igazán segíteni, ahol a vizsgált rendszer olyan nagy és bonyolult [31], hogy analitikus megközelítése kivitelezhetetlen számításokat eredményezne. Ilyen összetett rendszer vizsgálata volt egy gyártórendszereket vizsgáló és új nemzetközi kutatási eredmények kikísérletező kutatási projekt feladata, ahol magas minőségi követelményeknek megfelelő technológiákat és folyamatokat kellett kidolgozni a gyártáshoz kapcsolódó területek széles skáláján (ütemezéstől kezdve az ellenőrzési technológiákig). A projekten belül a minőségellenőrzési stratégiák optimalizálását is el kellett végezni szimuláció segítségével műszaki és gazdasági feltételek egyidejű figyelembe vételével [14].

A projekt keretében egy konkrét közép kategóriájú ipari gyártó és összeszerelő vállalat gyártási folyamata képezte a szimulációs feladat tárgyát, ahol csőszervényeket, armatúrákat és ipari elektronikát készítenek. A rendszeranalízis után a következő lépés a meglévő rendszer szimulációs modelljének felépítése volt valamint annak verifikálása és validálása összevetve a valóságos rendszerrel, ami megalapozta a további szimulációs vizsgálatokat. A szigorú minőségi követelményeknek eleget tevő gyártási-összeszerelési folyamatok több gyárüzemben termelés esetén pontos ütemezést igényelnek. Az éles gazdasági versenyben rendkívül fontos szempont, hogy ez lehetőleg elfogadható költségek mellett történjék. A különböző lehetséges (logisztikai, gazdasági, stb.) stratégiák kipróbálásának ma az egyik leghatékonyabb útja a szimulációs vizsgálat. Ennek költsége ugyanis töredéke annak, mint amikor a vezetők különböző ötleteiket a valóságos rendszeren próbálják ki, ráadásul a kipróbálás eredménye is hamarabb rendelkezésre áll.



13. ábra Egy termék anyagáramlási folyamatának Gantt-modellje

A minőségellenőrzés műszaki-gazdasági optimalizálása a projektben résztvevő cég egyik legjellemzőbb termékének alapján történt, melynek Gantt diagrammját a 13. ábra mutatja [47]. Ez a folyamat került modellezésre számítógépes szimulációval a CASSANDRA rendszerrel [15]. A szimulációs vizsgálódásra kiválasztott termék egy hőszabályzó volt, melynek a gyártási folyamata röviden a következő (14. ábra). Nyersanyagokat vásárol a vállalat, amelyeket a gyárba szállítás után helyi raktárban, külön lerakodó helyeken tárolnak. Nyersanyagok közül a henger alakú vékony rudakat (ZS: Zylinderstift) eldarábolják és két ilyen darabot hozzáerősítenek (DEZ: Düseneinsatz + Zylinderstift) egy hatszög alakú alaphoz (S6: Sechskant), amely a feldarabolás után (DE Düseneinsatz) már megfelelő nagyságban áll rendelkezésre. Két fajta lemezt (B1, B2: Band) összehegesztenek, két különböző méretben feldarabolják (D1S, D2S: DuoStahlPlatte Stanzen), majd utómunkálatoknak vetik alá (D1W, D2W: DuoStahlPlatte Walzrichten). Ezután történik a fő összeszerelési folyamat (Regler Fertigen) az alkatrészekből, ahol az említetteken kívül szükség van még a következőkre: biztosító rugó (SF: Sicherungsfeder), szeleptű (DN: Düsennadel), vezető darab (FS: Führungsstück), távolságtartó (DIS: Distanzscheibe). A végső munkafázisban hőkezelik a terméket (WB: Wärmebehandlung) és csomagolják (PN: Packen). A közbenső folyamatok között meg kell oldani a tárolást (S: Store), és bizonyos pontokon minőségellenőrzést kell tartani (P: Prüfung). Itt a legkülönbözőbb minőségellenőrzési stratégiákat lehetett szimulációval kipróbálni, de ezen kívül más paraméterek is változtathatók. A raktárak (az ábrán S-el jelölt téglalapokkal voltak reprezentálva) kapacitása a szimuláció során szintén változtatható, és a gyártás folyamán az ütemezés is módosítható. A szimuláció során tehát különböző változatokat lehetett megvizsgálni, hogy a szimulációs rendszerben alkalmazott ágens segítségével meghatározzuk az optimális megoldást. [46]



14. ábra Az MI-vel vezérelt szimulációs kísérlet sorozatnál optimum keresésének pillanatképe

Szimulációval a minőségellenőrző pontok elhelyezését kellett megoldani a projektben, hiszen ezek a gyártási folyamat minden egyes részfolyamatai közé beilleszthetők lennének, továbbá az egész termelés végére is el lehet egyet helyezni. A túl sok ellenőrzőpont megnöveli a gyártási költséget, mivel az alkatrészeket el kell oda juttatni (még ha egy gyáregységen belül is történik ez a mozgatus, mindenképpen lefoglal valamilyen erőforrást) és ezen kívül a vizsgálat is pénzbe kerül. A közbenső vizsgálatok nélküli gyártás szintén drága, mert a sok hibás alkatrészt főlöszlegesen munkálják meg a munkagépek és az egész gyártási folyamat végén detektált hibás végtermékek kidobása növeli a költséget. Az ellenőrzési pontok egy adott kombinációja meghatároz egy logisztikai stratégiát, és ehhez a bonyolult gyártási folyamathoz rengeteg kombináció létezik. A felépített modellt a 14. ábra mutatta, amelyen a mesterséges intelligenciával vezérelt szimulációs kísérletsorozatnál éppen az optimális konfiguráció keresésének pillanatképe látható. A legjobb megoldás keresésénél a minőségellenőrző pontok elhelyezésénél az előzőekben említett két szélsőséges eset különböztethető meg:

- az egyik esetben minden egyes részfolyamat között van közbenső ellenőrzési pont az alkatrészek szűrópróbas vizsgálatával,
- a másik esetben sehol sincs minőségellenőrzés, csak a gyártósor legvégén.

Megvizsgálva ezt a két esetet, a jó végtermékekre vonatkoztatott relatív költség, ahogy az előző magyarázatban olvashattuk, mindkettőnél magas. A minimális költség a kettő között húzódik meg, amit a mesterséges intelligencia segítségével lehet megkeresni oly módon, hogy a modell struktúráját megváltoztatva új ellenőrzési pontok kijelölésével és régiek megszüntetésével ellenőrzési pontok olyan rendszerét keressük, mely a kisebb költség felé mutat.

A 4. Táblázat mutatja a szimulációval kapott részletes eredményeket összehasonlítva a két szélsőséges esettel. Az értékeken jól látható, hogy a megtalált optimum mindkét alapesetnél olcsóbb megoldást kínál. Az optimum ellenőrzési költsége kicsit magasabb ugyan az ellenőrzés nélküli esetenél, de a kidobott anyag és felhasznált erőforrások tekintetében oly mértékben megelőzi azt, hogy az összesített költsége kisebb lesz. A logisztikai (mozgatus, szállítás), raktározási, rendelés-feldolgozási és információs költségek valamint a készlettartási költségek mindegyik esetben ugyanakkorák voltak, így nem szerepelnek a táblázatban. A szimulációval megtalált megoldásnál tehát bizonyos részfázisok után nincs, másoknál van minőségellenőrző pont, ezt az optimális konfigurációt írja le az 5. Táblázat.

4. Táblázat A szimuláció eredménye és a két szélsőséges eset

Átlagos költség egy	Összes ellenőrzéssel	Ellenőrzés nélkül	Megtalált optimum
Ellenőrzési költség	2,663	1,302	1,513
Kidobott anyag	0,154	1,061	0,151
Erőforrás	3,231	3,611	3,252
Beépített anyag	8,009	8,009	8,009
Összes	14,057	13,983	12,926

5. Táblázat Minőségellenőrző pontok az egyes műveletek után az optimális konfigurációban

B1	B2	D1S	D1W	D2S	D2W	DE	DIS	DN	FS	S6	SF	ZS
Nincs	Van	Nincs	Nincs	Van	Nincs	Nincs	Van	Van	Van	Nincs	Van	Van

A szimulációs modell felépítést követően a mesterséges intelligenciát alkalmazó optimalizálás a projektben résztvevő német vállalatnál olyan költségcsökkenést eredményezett, ami szignifikánsan jelezte a módszer eredményességét. Az optimalizálást el lehet végezni akár vegyes, akár változó termékstruktúrák esetében is, és további előny, hogy a modellek felépítése könnyen bővíthető modellelem-könyvtár objektumaiból történik, mely egyrészt további elemekkel bővíthető, másrészt tetszőleges rendszerek építhetők össze belőlük. A feladatot tovább lehet még bonyolítani a minőségellenőrzési pontok optimális térbeli elhelyezkedésének vizsgálatával, vagy egyéb optimalizálási feladattal, de már ebből is látszik a metodika használhatósága.

8.4. Közlekedési alkalmazás

Közlekedési szimuláció alkalmazási példa keretében egy olyan Európai Unió projekt [48] kerül bemutatásra, mely arra vállalkozott, hogy nemzetközi összefogással egy standard, nyílt, úthálózat- és forgalom-szimulációs keretrendszert hozzon létre összekapcsolódási lehetőséget nyújtva különböző forgalom-szimulátorok, forgalom-irányító rendszerek és adatforrások között. Feladata egy város úthálózatán lévő forgalom szimulációja, analízise és ehhez kapcsolódó döntéstámogató rendszer megalkotása volt. A kutatás motivációja erősen felhasználó központú, a megcélzott felhasználók többek között kis- és nagyvárosi forgalomtervező szakemberek, tömegközlekedési szakértők. A feladatot a közlekedés szimulátoroknál egy olyan fajta hiány illetve rés hívta életre, amelynek a betöltésével leegyszerűsíthetők lennének a komplex feladatok megoldásai.

Egy ilyen új keretrendszerre a meglévő közlekedés szimulációs eszközök mellett nagy szükség volt, hogy a létező szimulátorok legjobb elemeit integrálja, valamint a forgalom szimulációjának eredményeivel a beavatkozások és a vezérlés lehetőségének biztosítása révén stratégiai döntések támogatására képes legyen. Például egy nagyvárosi környezetben konfliktushelyzetek, balesetek vannak, amelyek összefüggésbe hozhatóak a nagy forgalommal, dugókkal, rosszul tervezett kereszteződéssel, ezek kezelése illetve egy minimális szintre való szorítása lehet egy ilyen döntés célja.

A keretrendszer egy nyitott platform lett, amely lehetővé teszi különböző szimulációs rendszerek, szimulációs segédeszközök, adatgyűjtő rendszerek és forgalomirányító rendszerek összekapcsolódását. A keretrendszer alapjául egy általános modell szolgált, amely a forgalom szimulációjának minden részelemét (topológia, topográfia, környezeti tényezők, emberi tényezők stb.) magába foglalta. A keretrendszerrel történő szimulációnál az egyes modulok működés közben megkapják a bemenő adataikat, és szolgáltatják az eredményeiket, amelyeket más modulok is felhasználhatnak. Egy bonyolult forgalom-szimulációs rendszer, amely egzakt analízist végez nagyvárosok úthálózatán, rendkívül nagy számítási kapacitást igényel. Ezért volt célszerű moduláris elemekből építkezni, ahol az egyes modulok külön számítógépeken helyezkedhetnek el. A moduláris felépítés további előnye, hogy az egyes komponensek megfelelő módon cserélhetőek, amennyiben jól definiált interfészen keresztül kívánnak kommunikálni, így a rendszer nagy rugalmassággal is felvértezhető.

A nyílt keretrendszer célja az volt, hogy a kerethez kapcsolt modulok integrációjával komplexebb feladatokat lehessen megoldani, és a végfelhasználók kezébe egy könnyen konfigurálható hatékony eszközt adjon közlekedési szimulációs területen. A keretrendszer 3 fő részből tevődik össze:

- Keretrendszer szerver, amely a különböző modulok közti kommunikációt biztosítja.

- GUI (Graphical User Interface), amely a felhasználó számára ad egy kezelői felületet. Itt állíthatja be a végfelhasználó a szimulációhoz szükséges opciókat, és a konfigurálás mellett itt lehet dinamikusan futtatni a közlekedési modellt.
- Adatbázis, amely az útvonal hálózatot írja le a modulok számára közös, egységes terminológiával.

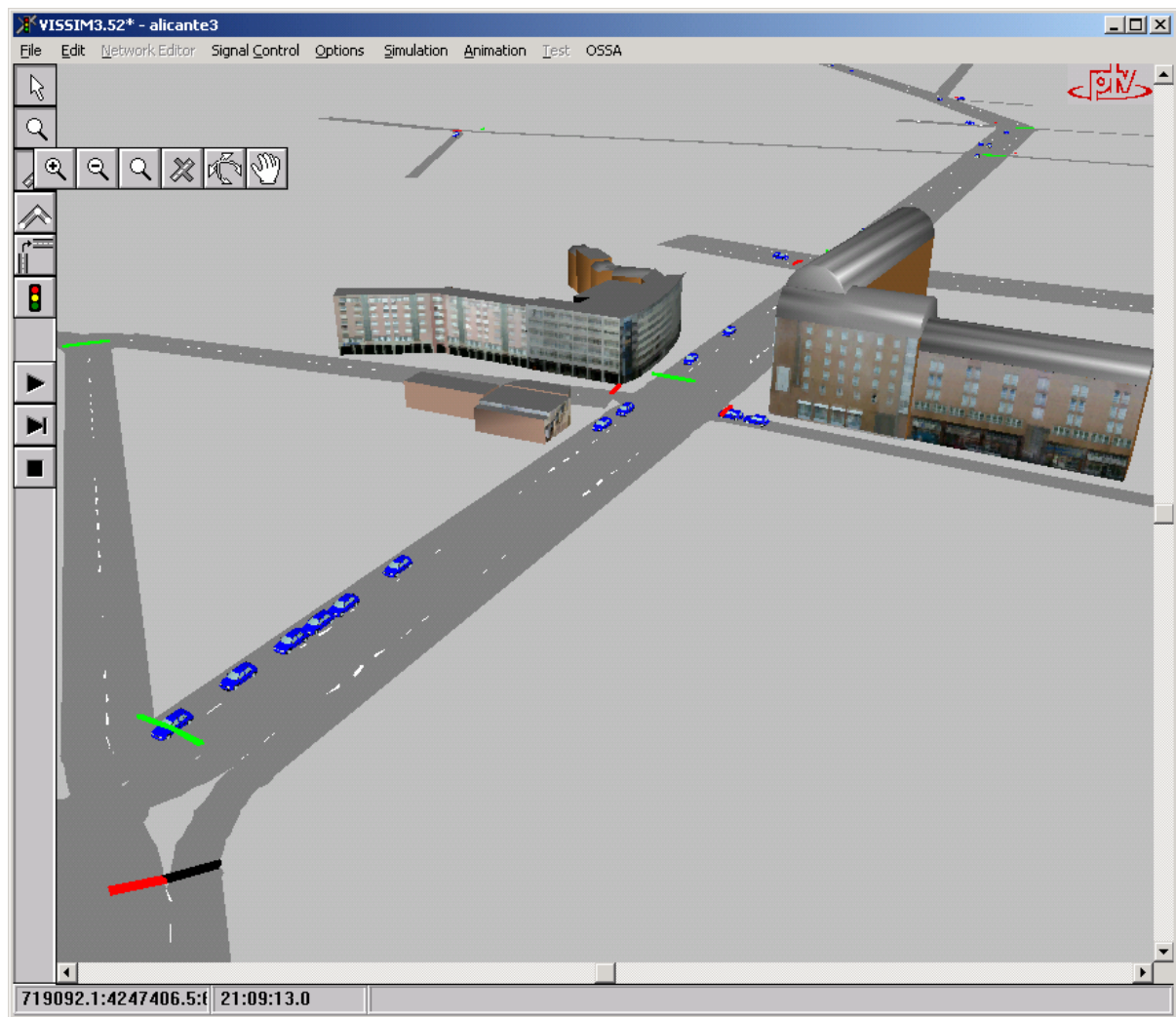
Modulszerepek

Az elkészült modulok továbbfejleszthetők, illetve igény szerint más keretrendszerekbe is átültethetők, az egyes meghatározott feladattal rendelkező modulok cserélhetőek, lehetővé válik ezáltal például az, hogy egy adott forgatókönyvet két különböző szimulációs modul segítségével futtassunk és értékeljünk.

A keretrendszerhez több típusú modul is csatlakoztatható, minden típus egy adott szerepet tölt be a teljes szimulációs rendszer szempontjából. A modulszerepek között megtaláljuk a szimulátort, megjelenítőt, analízátort, O/D mátrix számítót és környezet-szimulátort. 7 standard szerep lett definiálva a modulok számára, amelyek alkalmazhatók a szimulációs környezetben és bármilyen modul (amely e szerepek valamelyikét betölti) csatlakoztatható a nyílt keretrendszerhez, e standard szerepek a következők:

- Szimulátor (Simulator): A rendszer legfontosabb, központi motorja, amely az úthálózatban levő járművek mozgásának kiszámításáért felelős.
- Forgalomvezérlő (Urban Traffic Control - UTC): Egyrészt információkat szolgáltat a vezérlőrendszerrel, másrészt itt oldható meg a forgalomvezérlés akár különböző vezérlési stratégiák alkalmazásával.
- Útvonal mátrix becslő (Origin-Destination matrix Estimator - ODE): Egy iteratív eljárás segítségével kiszámítja az O/D mátrixot a historikus forgalmi adatok alapján.
- Forgalmi adat szolgáltató (Traffic Data Provided): Valódi historikus forgalmi adatokat szolgáltat a szimulációs környezet számára. Általában az UTC egységgel áll csak kapcsolatban, azaz rajta keresztül történik az adatok átadása.
- Megjelenítő (Visualizer): A járművek megjelenítésével segít a végfelhasználónak megérteni, illetve konklúziókat levonni a szimulációs modellel kapcsolatban.
- Analizátor (Analyser): Különböző eljárásokat tartalmaz a szimulációs végeredmény kiértékelésére és statisztikai eredmények elkészítésére (átlag, szórás, korreláció, stb.).
- Környezetszimulátor (Cosimulator): Szimulátor kiegészítő eszköze a járművek által kibocsátott szennyezőanyagok modellezésére és szennyezési értékek kiszámítására.

Ezeknek a moduloknak nem szükséges egy számítógépen futniuk, egy közös hálózaton levő gépekre külön-külön telepíthetők és futtathatók. Így a rendszer sokkal nagyobb teljesítményre képes, hiszen az egyes gépek kevésbé lesznek leterheltek. A rendszert konfigurálhatjuk úgy is, hogy egy-egy gépre 2-3 modult rakunk, de természetesen a legnagyobb sebesség növekedés abban az esetben érhető el, ha minden modul külön számítógépre kerül. A következő ábrán (lásd 15. ábra) a megjelenítő modul által mutatott városi úthálózat modellje (útvonalhálózat egy részlete útkereszteződésekkel, utakkal) kerül bemutatásra 3 dimenziós ábrázolással.



15. ábra Közlekedési rendszer szimulációja

Szimuláció indítása és analízálási lehetőségek

A szimuláció indítása előtt a felhasználónak meg kell még határozni a szimulációs keretfeltételeket, azaz be kell állítani a szimulálandó időintervallumot (az időszak elejét és végét), a szimulátor időinkremensét (ugyanis diszkrét szimulációs rendszerről lévén szó: az időt adott egységgel lehet csak növelni), a szimulátor kezdeti beállításához szükséges lépések számát.

A szimuláció végén nagyon fontos feladat az eredmények kiértékelése és analízálása. Az analízátor modul végzi a szimulációs keretrendszer segítségével fellelhető adatok összegyűjtését, az információk kiértékelését és továbbítását a megjelenítést végző modulhoz, valamint statisztikai analízist, korrelációs számítást, szenzitivitás analízist, környezet-szennyezési számításokat is képes elvégezni. További feladata a szimulátor futásidejű irányítása, az egész keretrendszerben felépített modellhez szükséges szimulációs lépések számának vagy az időtartamának a meghatározása. Az itt bemutatott keretrendszer gyakorlati alkalmazására két európai városban került sor: Alicante és Manchester.

8.5. Összefoglalás

A szimuláció elméleti és gyakorlati részeinek összefoglalásaként összegyűjthetjük a szimuláció előnyeit, melyeket két nagy csoportba rendezhetünk, teoretikai és technikai [10] jellegű csoportba. Nézzük először a teoretikai előnyöket:

- A szimuláció különböző hipotézisek kipróbálására alkalmas.
- Segít a rendszer megértésében.
- Könnyebb a modellt módosítani, mint a valós rendszert.
- Általában olcsóbb, mint a valós rendszer részletes vizsgálata.
- Módosítások száma lényegesen nagyobb lehet, mint a valós rendszer esetén.

A technikai előnyök pedig a következők:

- A szimuláció keretet biztosít a tesztelésre és a módosítások kipróbálására.
- A sebesség változtatható (gyorsítható, lassítható).
- A szimuláció megállítható, folytatható, az aktuális állapot elmenthető, visszaállítható.
- Egy megállított állapotban a modell tetszőlegesen vizsgálható.
- A kísérletek ugyanolyan körülmények mellett bármennyiszer megismételhetők.

A bemutatott gyakorlati feladatok megoldásával és a szimulációs vizsgálatok végzésével a módszertan ismertetése volt a fő cél. Ez a metodika egy olyan általánosítható eljárás, mely különböző alkalmazási területekről összegyűjtött, de hasonló jellegű feladatok megoldására alkalmas. A szimulációs technika alkalmazásával az informatikának egy olyan részterületét lehetett kihasználni, mely hasznos eszköz a közgazdászok, mérnökök, vezetők számára is. Előnye ugyanis, hogy nem csak egy valós rendszer működését lehet kipróbálni a segítségével, hanem egy elképzelt vagy tervezett rendszerét is, így már az új technológiák bevezetése előtt rá lehet mutatni az új módszer előnyeire, illetve meg lehet találni a hiányosságait. A műszaki-gazdasági optimalizálásnak azért van igen fontos szerepe, mert ezáltal számos stratégiát lehet kipróbálni a vállalat hatékonyságának és gazdaságosságának növelésére a valóságos rendszeren kísérletezéshez képest elenyészően kis költséggel szimuláció segítségével.

A szimulációs metodika magasabb szintjét tekintve a további fejlődési irányba tartozik az identifikációs-rekonstrukciós problémák megoldása (melyben jelenleg is folynak kutatások). Az algoritmikus szinten pedig az árnyaltabb és emberi gondolkodást jobban követő soft computing technikák, mint a fuzzy és a neurális hálózatok bevonása a szimulációba [27] mutatja a fejlődés irányát.

Irodalomjegyzék

- [1] Aalst, van der, W. M. P.; van Hee, K. M.: Business process redesign: A Petri-net-based approach. *Computers in Industry*, 29. k. 1–2. sz. 1996. p. 15–26.
- [2] Adan, Ivo and Jacques Resing: *Queueing Theory*, Department of Mathematics and Computing Science, Eindhoven University of Technology, Eindhoven, Netherlands, February, 2001.
- [3] Benson, D.: Simulation modeling and optimization using ProModel. *Proceedings of the '97 Winter Simulation Conference*, eds. Andradóttir, S.; Healy, K. J.; Withers, D. H.; Nelson, B. L., 1997. p. 587–593.
- [4] Benyó Z., Paláncz B., Szilágyi L.: *Insights into Computer Science with MAPLE*, Scientia Publishing House, Cluj-Napoca, 2005.
- [5] Benyó Zoltán: *Folyamatszimuláció – Kompartiment (Rekesz) modellek*, BME, Villamos és Informatikai Kar, Budapest, 1996.
- [6] Binder, Kurt: *Applications of the Monte Carlo Method in Statistical Physics*, Springer-Verlag, Berlin, 1987.
- [7] Bratley, P., Fox, B.L., Schrage, L.E.: *A Guide to Simulation*, Springer-Verlag, 1987.
- [8] Clymer, J.: *System Analysis Using Simulation and Markov Models*, Prentice-Hall, New Jersey, 1990.
- [9] Ferenczi M., Pataricz A., Rónyai L. (edited by): *Formal Methods in Computing*, Akadémiai Kiadó, Budapest, 2005.
- [10] Fishman, George S.: *Discrete-Event Simulation, Modeling Programming and Analysis*, Springer-Verlag, New York, 2001.
- [11] Györfi László, Páli István: *Tömegkiszolgálás informatikai rendszerekben*, Műegyetemi Kiadó, Budapest, 1996.
- [12] Harrel, C. R.; Tumay, K.: ProModel tutorial. *Proceedings of the Winter Simulation Conference*, eds. Swain, J.; Goldsman, D.; Crain, R.; Wilson, J.; Arlington, V. A. 1992. p. 405–409.
- [13] Hunyadi László, Mundruczó György, Vita László: *Statisztika*, Budapesti Közgazdaságtudományi Egyetem, Aula Kiadó, 1996.
- [14] Jávor A., Benkő T., Szűcs G., Vigh Á.: *Szimulációs modellezés szerelő-gyártó minőségellenőrző rendszerek minőségi követelményeinek biztosítására. A FAMOS-QUACAR EUREKA projekt keretében elért eredmények, valamint további alkalmazási lehetőségek*. Tanulmány, 1995.
- [15] Jávor András: CASSANDRA: Ein Dämon-Gesteuertes Universales Simulationssystem. *Symposium Simulationstechnik*, Wien, Austria, 25–27 September 1990. (6) p.130–134.
- [16] Jávor András: *Diszkrét szimuláció*, Széchenyi István Egyetem, Universitas-Győr Kht., 2000.
- [17] Jávor, András: Demons as Forerunners of Software Agents in Simulation, *Proc. of the International Conference on Modeling and Simulation – Methodology, Tools, Software Applications*, July 31, 2006, Calgary, Canada, pp. 151-155.
- [18] Jávor, András: Knowledge Attributed Petri Nets, *Systems Analysis, Modelling, Simulation*, 13, No. 1/2, 1993, pp. 5-12.
- [19] Jensen, G., Rozenberg, G., *High-Level Petri Nets*. Springer, 1991.
- [20] Kleijnen, J. P. C., *Statistical Tools for Simulation Practitioners*, Marcel Dekker, Inc., New York, 1987.
- [21] Kleijnen, J. P. C.: *Statistical Techniques in Simulation*, Parts 1 and 2, Marcel Dekker, New York, 1975.
- [22] Kleinrock, Leonard: *Sorbanállás, kiszolgálás – Bevezetés a tömegkiszolgálási*

- rendszerek elméletébe, Műszaki Könyvkiadó, Budapest, 1979.
- [23] Knuth, Donald Ervin: A számítógép programozás művészete, Műszaki Könyvkiadó, Budapest, 1994.
 - [24] L'Ecuyer, P.: Random Numbers for Simulation, Commun. Assoc. Comput. Mach., 33, 1990.
 - [25] Law, A. M., W. D. Kelton: Simulation Modelling and Analysis, 3rd ed. McGraw-Hill, New York, 1997.
 - [26] Makoto Matsumoto, Takuji Nishimura: Sum-discrepancy test on pseudorandom number generators, Mathematics and Computers in Simulation 62, 2003, pp. 431–442
 - [27] Moeller, Dietmar P. F.: Mathematical and Computational Modeling and Simulation, Fundamentals and Case Studies, Springer-Verlag, Berlin, 2004.
 - [28] Mogyoródi József, Michaletzky György: Matematikai statisztika, Nemzeti Tankönyvkiadó, Budapest, 1995.
 - [29] Nelson B. L.: Stochastic Modeling, Analysis and Simulation, McGraw-Hill, Inc. New York, 1995.
 - [30] Neumann J.: Various Techniques used in Connection with Random Digits, NBS Applied Mathematics Series (1951) 12, 36-38.
 - [31] Ören, T. I., B. P. Zeigler, M. S. Elzas (edited by): Modelling and Simulation Methodology (Knowledge Systems' Paradigms), Elsevier Science Publishers B.V. Amsterdam, 1989.
 - [32] Pataricza András: Formális módszerek az informatikában, Typotex, Budapest, 2006.
 - [33] Peterson, J.L.: Petri Nets, Computing Surveys, Vol. 9, No. 3, September, 1977, pp. 224-252.
 - [34] Petri, C., Kommunikation mit Automaten. PhD Dissertation, University of Bonn, Germany, 1962.
 - [35] Prezenszki József: Logisztika II (Módszerek, eljárások), Logisztikai Fejlesztési Központ, Budapest, 2000.
 - [36] Reimann József: Valószínűségelmélet és matematikai statisztika mérnököknek, Tankönyvkiadó, Budapest, 1992.
 - [37] Ripley, B. D.: Stochastic Simulation, Wiley, New York, 1986.
 - [38] Ross, Sheldon M.: Simulation, Academic Press, San Diego, California, 2002.
 - [39] Rubenstein, R. Y.: Simulation and the Monte Carlo Method, Wiley, New York, 1981.
 - [40] Russell, S. J., P. Norvig: Mesterséges Intelligencia modern megközelítésben, Panem-Prentice Hall, 2000.
 - [41] Sarjoughian, H. S., F. E. Cellier (edited by): Discrete Event Modeling and Simulation Technologies, A Tapestry of Systems and AI-Based Theories and Methodologies, Springer-Verlag, New York, 2001.
 - [42] Srinivasan A., Mascagni M., Ceperley D.: Testing parallel random number generators, Parallel Computing 29 (2003) pp 69-94.
 - [43] Szegedi Zoltán: Prezenszki J.: Logisztika-menedzsment, Kossuth kiadó, 2003.
 - [44] Sztrik János: Kulcs a sorbanállási elmélethez és alkalmazásaihoz, Debreceni Egyetem, Matematikai és Informatikai Intézet, Debrecen, 2000.
 - [45] Sztrik János: Raktározási és kiszolgálási problémák matematikai modellezése, Debreceni Egyetem, Informatikai Kar, Debrecen, 2004.
 - [46] Szűcs G., Vigh Á.: Simulation of a quality controlled manufacturing and assembly system. IMACS – SAS'95, June 26–30, 1995, Berlin, Germany, 279–282.
 - [47] Szűcs Gábor: Termelési logisztika optimalizálása szimulációval, Logisztika, IX. évf., 2. szám, 2004. március, pp. 41-51.
 - [48] Szűcs Gábor: Városi közlekedési modellek moduláris szimulációjának vizsgálata és analízise, Közlekedéstudományi Szemle, LIII. évf., 2003. október, pp. 385-389.

- [49] Tang, Hui-Chin: Reverse multiple recursive random number generators, *Stochastics and Statistics, European Journal of Operational Research* 164, 2005, pp. 402–405.
- [50] Tarlós Béla: A diszkrét szimuláció módszertana, Tankönyvkiadó, Budapest, 1989.
- [51] Tricas, F., J. Martinez: Distributed control systems simulation using high level Petri nets. *Mathematics and Computers in Simulation*, 46 (1998), 47-55.
- [52] Winston, P. H.: *Artificial Intelligence*. Addison-Wesley Publishing Company, 1977.
- [53] Winston, Wayne L: *Operációkutatás (Módszerek és alkalmazások)* 1-2. kötet, Aula Kiadó, 2003.
- [54] Ziegler, B. P. *Multifaceted Modelling and Discrete Event Simulation*, Academic Press Inc., 1984.

Tartalomjegyzék

1.	Bevezetés.....	3
1.1.	Szimulációs vizsgálati módszer	4
1.2.	Diszkrét esemény rendszerek alapfogalmai	4
2.	Álvéletlenszám generálás.....	6
2.1.	Monte Carlo módszer	6
2.2.	Lehmer-féle multiplikatív kongruencia módszer	7
2.3.	Elfogadás-visszautasítás módszere (EVM).....	8
2.4.	Tetszőleges eloszlású álvéletlenszámok generálása.....	10
2.5.	Álvéletlenszámok jóságának tesztelése.....	12
3.	Matematikai és modellezési előkészítés.....	14
3.1.	Matematikai apparátus	14
3.2.	Időhorizont vizsgálata	14
3.3.	Esemény, aktivitás és folyamat leírási módok	16
4.	Szimulációs modell felépítése.....	18
4.1.	Szimulációs modell felépítésének fázisai.....	18
4.2.	Kendall-féle osztályozás	21
5.	Szimulációs futtatás.....	25
5.1.	A felfutási idő analízis.....	25
6.	Szimulációs eredmények kiértékelése.....	29
6.1.	Szimulációs eredmények hibái.....	29
6.2.	Feltételes variancia formula módszere	30
6.3.	Ellentétes változók módszere	33
6.4.	Kontroll változók módszere	34
6.5.	Rétegzett mintavétel (stratified sampling)	37
7.	Modellezési lehetőségek	39
7.1.	Petri hálók	39
7.2.	Markov láncok.....	47
7.3.	Diszkrét szimulációs eszközök.....	48
8.	Szimulációs alkalmazások.....	50
8.1.	Gyártósor és logisztikai rendszer szimulációja ProModellel	50
8.2.	Szimuláció Mesterséges Intelligenciával (MI).....	52
8.3.	Optimalizálás szimulációval gyártósoroknál	53
8.4.	Közlekedési alkalmazás	56
8.5.	Összefoglalás.....	59
	Irodalomjegyzék.....	60

Az „OPERÁCIÓKUTATÁS” sorozatban
eddig megjelentek:

Nagy Tamás¹ – Klafszky Emil²:
SZTOCHASZTIKUS JELENSÉGEK

Komáromi Éva³:
LINEÁRIS PROGRAMOZÁS

Deák István⁴:
BEVEZETÉS A SZTOCHASZTIKUS PROGRAMOZÁSBA

Hujter Mihály⁵:
PERFEKT GRÁFOK ÉS ALKALMAZÁSAIK

Etienne de Klerk⁶ – Cornelis Roos⁷ – Terlaky Tamás⁸:
NEMLINEÁRIS OPTIMALIZÁLÁS

Szántai Tamás⁹:
PERT ALKALMAZÁSOK

Komáromi Éva:
KOCKÁZAT, DÍJ, TARTALÉK

Rapcsák Tamás¹⁰:
NEMLINEÁRIS OPTIMALIZÁLÁS

A kötetek megrendelhetők az AULA könyvesboltjában:
1093 Budapest, Fővám tér 13-15. Telefon: (36)-482-8771

¹ Miskolci Egyetem Matematikai Intézete, Alkalmazott Matematika Tanszék, e-mail: matente@gold.uni-miskolc.hu

² Budapesti Műszaki és Gazdaságtudományi Egyetem Építéskivitelezés Tanszéke, e-mail: klafszky@ekt.bme.hu

³ Budapesti Corvinus Egyetem Operációkutatás Tanszéke, e-mail: komaromi@bkae.hu

⁴ Budapesti Műszaki és Gazdaságtudományi Egyetem Matematikai Intézete, Differenciálegyenletek Tanszék, Operációkutatási Csoport, e-mail: deak@math.bme.hu

⁵ Budapesti Műszaki és Gazdaságtudományi Egyetem Matematikai Intézete, Differenciálegyenletek Tanszék, Operációkutatási Csoport, e-mail: hujter@math.bme.hu

⁶ Department of Combinatorics and Optimization, University of Waterloo, Waterloo (ON), Canada, edeklerk@math.uwaterloo.ca; <http://www.math.uwaterloo.ca/~edeklerk>

⁷ Department of Information Systems and Algorithms, T.U. Delft, Delft, The Netherlands, C.Roos@ewi.tudelft.nl; <http://www.isa.ewi.tudelft.nl/~roos>

⁸ Department of Computing and Software, McMaster University, Hamilton (ON), Canada, terlaky@mcmaster.ca; <http://www.cas.mcmaster.ca/~terlaky>

⁹ Budapesti Műszaki és Gazdaságtudományi Egyetem Matematikai Intézete, Differenciálegyenletek Tanszék, e-mail: szantai@math.bme.hu

¹⁰ MTA Számítástechnikai és Automatizálási Kutató Intézet, Operációkutatás és Döntési Rendszerek Laboratórium és Osztály, e-mail: rapcsak@oplab.sztaki.hu