Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Méréstechnika és Információs Rendszerek Tanszék

Paraméteres modellellenőrzés folytonos idejű Markov-láncokhoz

TDK dolgozat

Készítette:

Nagy Simon József

Konzulens:

Marussy Kristóf

2018

Tartalomjegyzék

Kivonat

Kritikus műszaki alkalmazások esetén fontos a funkcionális helyesség mellett a különböző extra-funkcionális jellemzőkre is garanciákat adni. Mind a kritikus beágyazott rendszerekben, a kiberfizikai rendszerekben, mind pedig a nagy rendelkezésre állású felhő alapú rendszereknek komoly extrafunkcionális követelményeknek kell megfelelniük. Azonban az extra-funkcionális követelmények gyakran különböző tényezők és paraméterek lehetséges értékeitől függenek. Ezen paraméterek meghatározása fontos a megfelelő extra-funkcionális követelmények teljesítése érdekében. Azonban a paraméterek meghatározását mind a külső zavarok, mind az alkatrészek meghibásodása, mind pedig a tényezők pontatlan ismerete nagyban megnehezíti. A paraméterek hatásának elemzése a tervezési folyamat lényeges része, melyre a legtöbb esetben heurisztikus megoldási módszereket alkalmaznak. Viszont ezen módszereknek mind az optimalitására mind pedig a megbízhatóságára a legtöbb esetben nincs precíz matematikai bizonyítás és garancia. Garanciák nyújtása különösen fontos kritikus rendszerek esetén. A megbízhatósági problémák jelentős hányada a folytonos idejű paraméteres Markov láncok analízisére vezethető vissza, vagy pedig nagy pontossággal modellezhetők így. Ezen sztochasztikus modellek egyik legfontosabb matematikai problémája a paraméteres modellellenőrzés, vagyis hogy a paramétertér milyen tartományain állnak fenn a Markov láncnál különböző feltételek, erre példák: elérési valószínűség, adott időkorláton belüli elérési valószínűség, várható elérési idő, elérés esetén várható reward (valamilyen aggregált mérték) mennyisége. Munkám során a jelenleg használt algoritmusok továbbfejlesztésével a fenti matematikai problémák automatizált megoldására tudok összetett rendszerekre garanciákat adni a modellezett rendszer felé támasztott extra-funkcionális követelmények teljesítésére a megfelelő paraméterek függvényében. Munkám által lehetőség nyílik tervezés időben a hibaanalízisre és a működés optimalizálására. Az általam kidolgozott algoritmusok újszerűsége, hogy a meglévő diszkretizált, diszkrét idejű algoritmusok működését kiterjeszti folytonos idő- és paramétertartományba. Így nemcsak a kapott megoldások pontossága nő meg, de ezáltal lehetőség nyílik ezen algoritmusok alkalmazhatóságát kibővíteni a kiberfizikai rendszerekre. Algoritmusaim széleskörű alkalmazhatóságát dolgozatomban esettanulmányban demonstrálom a kritikus beágyazott rendszerek és az útvonaltervezés területén. Végül még érdemes megjegyezni, hogy számos jelenlegi algoritmussal ellentétben (például állapot redukálás) megoldásom a jobb skálázhatóság elérésére elosztott számítógépes rendszereken is implementálható, mely a további munka részét fogja képezni.

1. fejezet

Bevezetés

A technológia rohanó fejlődésével, egyre inkább kulcskérdéssé válik a megbízható működés garantálása valamint az erőforrások optimális felhasználása. Hiszen a számítógép vezérelt elektronikus rendszerek már az iparból fokozatosan áterjednek hétköznapi életünkbe is. Ezek közül a kritikus beágyazott rendszerekben, a kiberfizikai rendszerekben, mind pedig a nagy rendelkezésre állású felhő alapú rendszerek kiemelt jelentőségűek. Hisz ezek jórészt működésük során folyamatos igénybevételnek vannak kitéve és működés közben történő meghibásodásuk komoly problémát jelent, és kiküszöbölésük elsődleges szempont. Így ezen rendszereknek számos magas szintű extra-funkcionális követelménynek kell megfelelniük. Ezeknek a teljesítéséhez a rendszernek mind a felépítését, mind a működését teljes egészében átfogó képre van szűkség, mely a valós üzemelést hitelesen modellezi. Erre a problémára a legtöbb esetben intuitív, heurisztikus modelleket alkalmaznak. Viszont ezen modellek helyességére nincsen matematikai bizonyítás és garancia. Továbbá a külső zavarok, zajok, az alkatrészek meghibásodása, fokozatos amortizációja illetve a felhasznált adatok bizonytalansága nagy mértékben lecsökkenthetik ezen módszerek megbízhatóságát és alkalmazhatóságát.

Ezek miatt felmerül az igény a rendszereink magas szintű precíz matematikai leírására, melyeknek elemzésével lehetővé válik az üzemi viselkedés pontos modellezése és ezáltal a üzemi meghibásodások megelőzése. Erre a célra a számos megközelítés létezik, melyek közül kiemelkednek a Markovi sztohasztikus modellek. Segítségükkel a megbízhatósági problémák nagy része precízen és átláthatóan modellezhető. Továbbá az ezen sztohasztikus modelleken értelmezett megoldó algoritmusok segítségével lehetővé válik a mérnöki problémáink pontos matematikai megoldása. Valamint ezen kívül lehetővé válik garanciák biztosítása a megbízhatóságra, illetve lehetőség nyílik meghatározni a hibamentes működés feltételeit. Így a paraméteres modellellenőrzés segítségével lehetővé válik meghatározni, hogy a vizsgált sztohasztikus modellek paramétereinek, mely értéktartományain állnak fenn a Markovi modell bizonyos tulajdonságai. Ilyen tulajdonságok, például az elérési valószínűség, adott időkorláton belüli elérési valószínűség, várható elérési idő, elérés esetén várható reward (valamilyen aggregált mérték) mennyisége. Viszont ekkor az esetek túlnyomó többségében azt vizsgáljuk, hogy a paraméterek, mely értékei esetén fog ezen tulajdonság átlépni egy adott, határt és mely paraméterértékek esetén lesz biztosan alatta.

A kutatásom során ezeket a jelenlegi paraméteres modellellenőrzőket fejlesztettem tovább, a még összetettebb problémák megoldása valamint a még pontosabb eredmény elérése érdekében. Így az általam kifejlesztett valamint továbbfejlesztett algoritmusok segítségével lehetővé válik olyan tulajdonságok precíz vizsgálata, amiket a korábbi algoritmusokkal eddig csak becsülni lehetett, mint például a várható megérkezési idő folytonos

idejű modellek esetén. Továbbá algoritmusaimat, speciális funkciókkal bővítettem, melyek megkönnyítik a mérnöki problémák matematikai módszerekkel történő vizsgálatát. Így lehetővé válik nemlineáris algebrai összefüggésekkel való számítás, a paraméterek bizonytalan ismeretéből származó hiba kiküszöbölése, valamint mérési adatok közvetlenül a modellbe történő illesztése. Ennek következményeként nemcsak a modellből kapott megoldás eredménye lesz pontos, hanem a modell is pontosan jellemzi a vizsgált rendszert. Hiszen az algoritmus nagy fokú pontossága elveszne, ha segítségével a valós problémákat csak nagy pontatlansággal tudná modellezni.

Munkám során először megvizsgáltam a már meglévő sztohasztikus modell megoldókat. Ezután megvizsgáltam milyen algoritmusok alapján működnek, különös tekintettel, hogy a milyen módszerek léteznek a paraméteres modellellenőrzésre. Ezeket a dolgozatomban a 3 fejezet 3.1 és 3.2 szakaszban mutatom be. Majd kutatómunkám zárásaként megvizsgáltam a sztohasztikus modellek gyakorlati alkalmazásait (és akár jövőbeli alkalmazási lehetőségeit), melyeket a 3.4 szakaszban találhatóak. Ezen kívül a 2 fejezetben bemutatom a munkám során felhasznált matematikai eszköztáramat, modelleim elemeit (2.1 szakasz), modelleim típusait (2.2 szakasz) valamint a modelleim tulajdonságait (2.4 szakasz). Majd rátérek részletesebben is egy már létező paraméteres megoldóra a 4.2 szakaszban, mely a megoldásaim alapját képezi. A 4.3 szakaszban bemutatom az általam kifejlesztett megoldásokat, algoritmusokat és azok bizonyításait. Végül a 5 fejezetben bemutatom az algoritmusom gyakorlati alkalmazhatóságát. Így a 5.1 szakaszban bemutatom a 4.3.2 részben található algoritmusom működésének hatékonyságát az általam létrehozott tesztprogramok segítségével. Ebben a részben megmutatom, hogy az általam vizsgált problémákra egy az algoritmusom egy rendkívül jól skálázódó, megoldást jelent. Majd ezután egy a 5.2 szakaszban MoDeS3 projekten keresztül bemutatom algoritmusaim gyakorlati alkalmazását, melyet követve számos másik mérnöki probléma is megoldható.

2. fejezet

Háttérismeretek

Ebben a fejezetben ismertetem matematikai eszköztáramat, mind azokat a fogalmakat és definíciókat, melyekkel a különböző mérnöki problémákat modellezem és melyeknek precíz elemzésével (lásd 4. fejezet) optimális megoldást biztosítanak.

Dolgozatom során kizárólag Markovi modellekkel dolgozom, így a fejezet első szakaszában ezen modellek elemeit (tranziciók, lokációk stb...) mutatom be, második szakaszában a Markovi sztohasztikus modellek fajtáira térek ki (folytonos és diszkrét idejű modellek, valamint Markov döntési folyamatok), míg a harmadik szakaszában a rajtuk értelmezett tulajdonságokat definiálom, mint például az adott célállapot elérésének valószínűsége, vagy az aggregált mértéknek várható értéke.

A könnyebb megértést elősegítendő a következő matematikai definíciókat gyakorlati példákkal illusztrálom a MoDeS3 projektből. Ezen példákban a MoDeS3 projektben kifejlesztett intelligens modellvasút terepasztal működését fogjuk elemezni, melyet egy elosztott állapotgép alapú program vezérel. Számos szenzor és kamera segítségével lehetővé válik a különböző veszélyhelyzetekre történő azonnali reakció. Ilyen például a vonat megállítása ha valami eléje kerül. Ezáltal lehetővé válik olyan kérdések precíz definiálása, mint a megérkezési idő vagy a hibahelyzetek előfordulási gyakorisága. Algoritmusaim használatát és alkalmazásait ezen projektben részletesen az 5. fejezetben, esettanulmány formájában mutatom be.

2.1. Markovi sztohasztikus modellek elemei

Mindenek előtt vegyük sorra a Markovi modellek elemeit, melyek a modelleim alapját képezik. Illetve definiálom a hozzájuk tartozó jelölésrendszert, amit a dolgozat során használni fogok.

Lokációk A lokációk [18] [1] határozzák meg az általam vizsgált, diszkrét állapotterű modellek lehetséges állapotait, beleértve az s kezdeti lokációt. Illetve általam használt sztohasztikus modelljeim definícióiban már $l \in L$ lokációhalmaz elemeiként szerepelnek. A dolgozatomban véges állapotterű modelleket vizsgálok így $|L| \in \mathbb{N}$ a lokációhalmaz is véges.

[1] Habár a nemzetközi szakirodalom egyszerre használja rá a "location" és a "state" kifejezést. Viszont mivel a dolgozatomban állapotgépeket is bemutatok, így azoktól való egyértelmű megkülönböztetés végett a dolgozatomban szisztematikusan a "lokáció" kifejezést használom.

Tranzíziók A lokációk közti átmenetet biztosítják a tranzíciók [18], melyek mindig pontosan két lokáció között futnak. Az átjárás (tüzelés) tulajdonságait viszont már a különböző modelltípusok határozzák meg.

- Diszkrét időben, a tranzíciót valószínűsége p jellemzi, mely meghatározza, hogy mekkora valószínűséggel választjuk ki őt tüzelésre a lokációból kimenő többi tranzíció közül.

- Folytonos időben a tranzíciót mindig a késleltetése jellemez, mely egy exponenciális eloszlást ($P(X < t) = 1 - e^{-\lambda \cdot t}$) mutató valószínűségi változó jellemzi. Ezt általában az eloszlás λ rátájával [2] szokás jellemezni.

Az általam használt modellekben minden esetben egyszerre csak egy tranzíció tüzel, és a tüzelés pillanatában atomi módon, azonnal átkerülünk az általa meghatározott lokációba. Továbbá ezen értékeket a Markov-láncok esetén egy $R : (L \times L) \to \mathbb{R}$ kétdimenziós rátafüggvény [3] határoz meg. Ezen függvény esetén az első argumentum a tranzíció kiindulási míg a második argumentum a célállapotot adja meg. Míg a függvény értéke diszkrét idejű modellek esetén a tranzíció valószínűségét, míg folytonos idejűeknél a késleltetési idő eloszlásának rátáját adja meg.

Paramétertér Legyen p egy Markov-lánc paraméter mely a hozzá tartozó (2.1) paraméterintervallumon bármely értéket felvehet.

$$I_p = [limit_{lower}; limit_{upper}] \subsetneq \mathbb{R} \tag{2.1}$$

Modelleinkben egyszerre számos paraméter szerepel. Így legyen V paramétertér [27] $m \in \mathbb{R}$ véges sok p paraméter halmaza. Továbbá $\mathcal{Q}_V$, V paramétereitől függő többváltozós multi-affin polinomok halmaza, ahol (2.2) fennáll.

$$f_V \in \mathcal{Q};\ f_V = \sum_{i=1}^{m} a_i \times \prod_{p \in V_i} p \tag{2.2}$$

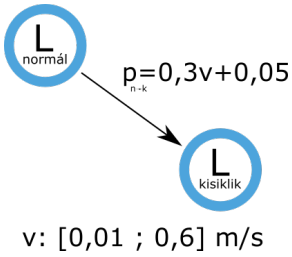
Ahol a $V_i \subseteq V$ a V paramétertér egy tetszőleges részhalmaza.

Definíció 1 (Lokális paramétertér). A paraméterek Markov-láncokon történő értelmezésekor rendkívül hasznos ha a különböző $l \in L$ lokációkhoz hozzá tudjuk rendelni azon $V_l \subset V$ lokális paraméterteret, mely azokat és csak azokat a $x \in V$ paramétereket tartalmazza, melyekre létezik olyan l-ből kimenő tranzíció, melyekhez tartozó $\mathcal{Q}_V$-beli polinom függ x paramétertől (hozzá tartozó polinom együttható nem nulla). ▪

Példa diszkrét idejű modellekre A MoDeS3 terepasztalon például egy gyakran megesik, hogy a túl nagy sebesség esetén a vonat kisiklik. Természetesen kis sebességnél is előfordul, viszont a tapasztalatok alapján megállapítható, hogy annak a valószínűsége, hogy egy adott hosszúságú sínszakaszon a vonat kisiklik lineárisan változik a vonat sebességével. Természetesen ez az összefüggés csak azokon a sebességeken érvényes amivel a vonat képes haladni. Ezen paraméter által meghatározott ráta a 2.1 ábrán látható.

Példa folytonos idejű modellekre Folytonos idejű modellek esetén a paraméterekkel a tranzíciókhoz tanzíció rátája megadja a késleltetés várható értékét. Emiatt a MoDeS3

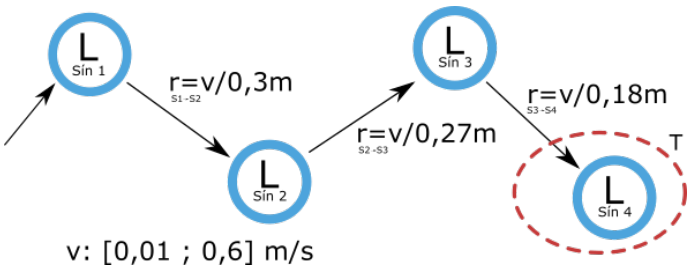
[2] Az ábrákon technikai okokból az r betűt használom.
[3] Ez a szakirodalomban sok helyen [3] rátamátrixként szerepel.



2.1. ábra. Paraméter diszkrét idejű sztohasztikus modellekben

terepasztalban a vonat mozgását a különböző sínszakaszokon rendkívül praktikusan lehet modellezni. Hiszen feltételezve, hogy a rövid, elemi sínszakaszok megtételéhez szűkséges idő exponenciális eloszlást követ a 2.3 levezetés segítségével a ráta egyszerűen kifejezhető a sebességgel mint paraméterrel (ekkor s, a megtett út hossza konstansnak tekinthető).

$$v = \frac{s}{t_{expected}} = s \cdot \lambda \rightarrow \lambda = \frac{v}{s} \tag{2.3}$$



2.2. ábra. Paraméter folytonos idejű sztohasztikus modellekben

2.2. Vizsgált sztohasztikus modellek

Ebben a szakaszban mutatom, be a munkám során felhasznált sztohasztikus modelleimet, az előző, 2.1 részben bemutatott elemek segítségével.

2.2.1. Diszkrét idejű Markov-lánc definíciója

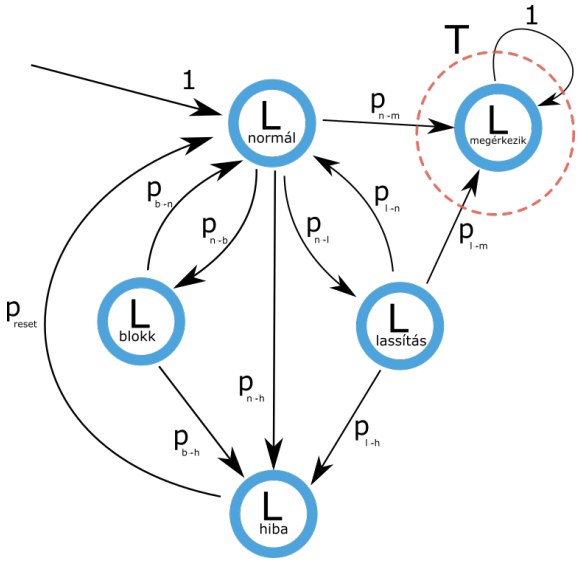
A diszkrét idejű Markov-lánc $\mu \in DTMC; \mu := \langle L, s, T, R \rangle$ (Discrete-Time Markov-Chain) egy L lokációhalmaz, egy s kezdeti lokáció, egy T célhalmaz illetve egy R rátafüggvény összessége alkotja. L a lokációk véges halmaza a Markov-lánc állapottere, R egy két változós függvény $R : (L \times L) \rightarrow \mathbb{R}$, amely megadja hogy egy adott lokációból milyen valószínűséggel lehet eljutni a többi lokációba, ahol bármely lokációhoz tartozó kimenő valószínűségek összege egy.

$$\forall li \in L \sum_{l \in L} R(li; l) = 1 \tag{2.4}$$

Illetve a $T \subseteq L$ a célhalmaz melynek elérése után a nem történik több lokációk közti átmenet.

Ezen modelleknek egy tipikus alkalmazása [21], amikor állapotgép alapú programot modellezünk vele, mint amilyen a MoDeS3 terepasztalban a vonatvezérlő szoftver. Az 2.3 ábrán látható ezen vezérlő állapotai illetve milyen külső események hatására következik

be. Ezen állapotgépből úgy kaptunk DTMC-t ha nagy mennyiségű naplóból meghatározzuk, hogy egy átlagos út során milyen relatív gyakorisággal következnek be az állapot átmenetek. Ekkor az állapotgép állapotait lokációknak feleltetjük meg, például a lassítás állapotát az $L_{\text{lassít}}$ lokációra képezzük le. Továbbá a lokációk mellé írt betűk a tranzíciók valószínűségeit jelölik, melyeknek értéke a 2.1-es táblázatban található. Ekkor az ábrán a szaggatott piros kör a T célhalmazt jelöli ki.



2.3. ábra. A vonat Markov-lánc alapú modellje

Tranzíciók	Valószínűségei
$p_{\text{normál}\to\text{megérkezik}}$	0,8
$p_{\text{normál}\to\text{hiba}}$	0,04
$p_{\text{normál}\to\text{lassít}}$	0,11
$p_{\text{normál}\to\text{blokk}}$	0,05
...	

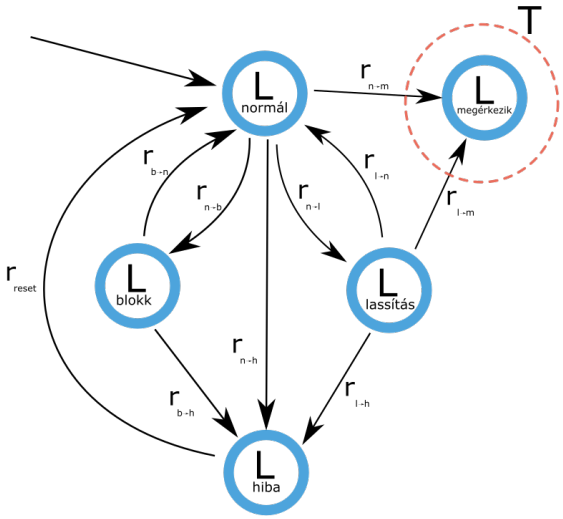
2.1. táblázat. Tranzíciók valószínűségei

Definíció 2 (Folytonos idejű Markov-lánc). A folytonos idejű Markov-lánc[22] [1] $\mu inCTMC\mu := \langle L, s, T, R_\lambda \rangle$ (Continous-Time Markoc-Chain) egy L lokációhalmaz, egy s kezdeti lokáció, egy T célhalmaz illetve egy R rátafüggvény, ahol L, s és T a jelentése DTMC-hez képest nem változott, viszont és $R_\lambda : (L \times L) \to \mathbb{R}$ rátafüggvény már nem egy valószínűséget hanem csúcsok közti időt meghatározó exponenciális eloszlás λ[4] rátáját [5] határozza meg. Így egy adott $l \in L$ lokációban a következő lokáció meghatározása, úgy történik hogy mindegyik tranzíció sorsol egy adott t-t a saját exponenciális eloszlása alapján és ezek közül a legkisebb érvényesül. ▪

Ezen matematikai eszközök segítségével lehetővé válik az idő számontartása és együttes vizsgálata a modell többi részével anélkül, hogy bármit is diszkretizálni kellene.

[4] $P(\leq t) = 1 - e^{-\lambda \cdot t}$

[5] Habár csábító lenne ezen λ változót paraméternek nevezni, de mivel az összekeverhető lenne a paraméteres Markov-láncok paramétereivel, így a dolgozat folyamán szisztematikusan rátaként hivatkozok rájuk.



2.4. ábra

Példa Így a 2.3 ábrán látható modellt kibővíthetjük a folytonos idejűre. Ezáltal lehetővé válik például a megérkezés idejének várható értékének számítása. Az így kapott folytonos idejű paraméteres Markov-lánc a 2.4 ábrán[6].

A CTMC-k esetén megvizsgálhatjuk, hogy egy adott lokációban várhatóan mennyi időt fogunk tölteni illetve ezen idő milyen valószínűségi eloszlást mutat [3].

$$P(t_0\text{-kor még l-ben vagyunk}) = \tag{2.5}$$

$$= P(t_0\text{-kor még t}_1\text{ nem tüzelt}) \cdot \ldots \cdot P(t_0\text{-kor még } t_n \text{ nem tüzelt}) = \tag{2.6}$$

$$= e^{-\lambda_1 \cdot t_0} \cdot \ldots \cdot e^{-\lambda_n \cdot t_0} = e^{-(\lambda_1 + \ldots + \lambda_n) \cdot t_0} \tag{2.7}$$

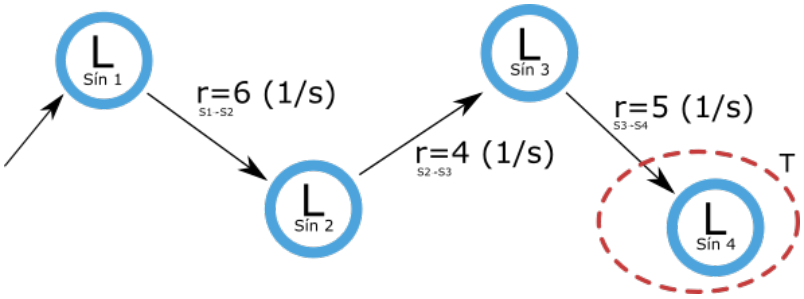
Így a fenti (2.5) levezetés miatt ezen idő is exponenciális eloszlást fog mutatni és λ rátája az adott csúcsból kimenő tranzíciók λ rátáinak összege lesz.

Példa A folytonos idejű Markov láncra egy példa a 2.5 ábrán látható, mely a 5.2.4 szakaszban bemutatott 5.7 ábrán lévő modell leegyszerűsített változata. Mivel a vezérlőrendszer képes meghatározni a vonat helyzetét, így a vezérlő azon állapotát, hogy melyik sínszakaszon van leképezzük lokációkra. Így például azt az állapotot, hogy a vonat a 3-as sínszakaszon van az $L_{\text{Sín 3}}$ lokáció modellezi. A piros csíkkal a T célhalmazt jelölöm.

Ezek alapján az kimeneti ráta[7] $r\colon L \to \mathbb{R}$ egy olyan függvény, mely minden egyes Lokációhoz hozzárendeli a kimenő tranzíciók rátáinak összegét, amely a bent tartozkódási idő

[6] Technikai okokból az ábrán a rátát r-rel jelölöm.

[7] Az angol szakirodalomban exit rate.



2.5. ábra. Vonat haladásának modellezése folytonos idejű Markov-lánccal

exponenciális eloszlás λ rátája lesz.

$$r(li) = \sum_{l \in L} R(li; l) \tag{2.8}$$

Több matematikai probléma megoldása során a számításokat és a bizonyításokat nagy mértékben leegyszerűsíti ha minden lokációhoz ugyanakkora kimeneti ráta tartozik.

2.2.2. Uniform Folytonos idejű Markov-lánc definíciója

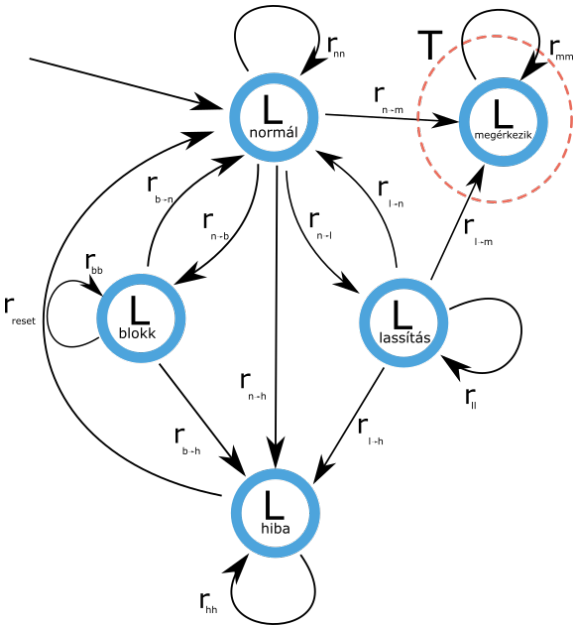
Egy $CTMC$-t $\langle L, s, T, R_\lambda \rangle$ uniformnak nevezünk [3] [22] ($uCTMC$) ha $\forall l \in L; r(l) = U$, ahol $U \in \mathbb{R}$ egy tetszőleges pozitív valós szám.

Felmerül ekkor, hogy hogyan lehet egy tetszőleges CTMC-ből uniform CTMC-t előállítani úgy, hogy a tulajdonságai nem változnak.

Definíció 3 (Uniformizáció). Uniformizáció [3] $\mathcal{U} : CTMC \to uCTMC$ során minden minden egyes lokációhoz $\forall l \in L$ hozzáadunk egy új hurok (saját magából induló és saját magába tartó) tranzíciót, hogy ezen új tranzíció λ rátájával az kimeneti rátát pont egy adott $\mathcal{U} \geq max(r(l \in L))$ értékre egészítse ki. ▪

Fontos kérdés, hogy ezen művelet nem fogja-e megváltoztatni az eredeti CTMC tulajdonságait. És bár az uniformizáció során új élek jönnek létre, melyek a viselkedésre is kihatnak mégis a az exponenciális eloszlás örökifjú tulajdonságának köszönhetően a CTMC legfontosabb tulajdonságai változatlanok maradnak. Hiszen miden új él hurok, így ha bármelyik közülük tüzel az nem befolyásolja a többi tranzíció érvényesülését, mert az örökifjú tulajdonságnak köszönhetően a tranzícióhoz kapcsolódó késleltetések memóriamentesek.

Példa Vizsgáljuk meg, hogy az uniformizáció hogyan befolyásolja a 2.4 ábrán bemutatott Markov-lánc viselkedését az uniformizáció. Az uniformizált modell az 2.6 ábrán látható. Ekkor ha az $L_{\text{normál}}$ állapotban vagyunk akkor a $L_{\text{normál}} \to L_{\text{megérkezik}}$ tranzíció tüzelése esetén átkerülünk a T célhalmazba, melyet a piros kör jelöl. Amennyiben az $L_{\text{normál}} \to L_{\text{hiba}}$ tranzíció korábban tüzel akkor a hiba állapotba kerülünk. Vizsgáljuk meg az uniformizáció során keletkezett $L_{\text{normál}} \to L_{\text{normál}}$ tranzíció hatását. Amennyiben ezen tranzíció később tüzelne mint bármelyik a "Blokk", "Hiba", "Lassítás" és "Megérkezik" lokációkhoz tartó tranzíciók közül, akkor értelem szerűen nem jutna érvényre. Amennyiben elsőként tüzelne azonnal visszakerülnénk a "Normál" lokációba. A tranzíciók tüzelési idejét egy exponenciális eloszlás határozza meg és ezen időt mindig a tranzícióba történő megérkezéstől számoljuk. Így ezen idő "számítása" újrakezdődik a $L_{\text{normál}} \to L_{\text{normál}}$ tranzíció tüzelése esetén.



2.6. ábra. Uniformizált folytonos idejű Markov-lánc

Az előbbiekben bemutatott modelleknek hiányossága, hogy a legtöbb esetben a modellek paramétereit (például a tranzíciókhoz tartozó valószínűség diszkrét időben, vagy az exponenciális eloszlás λ rátája folytonos időben) nem ismerjük pontosan, vagy épp ezen rátákat meghatározó paraméterek optimális megválasztása a feladat.

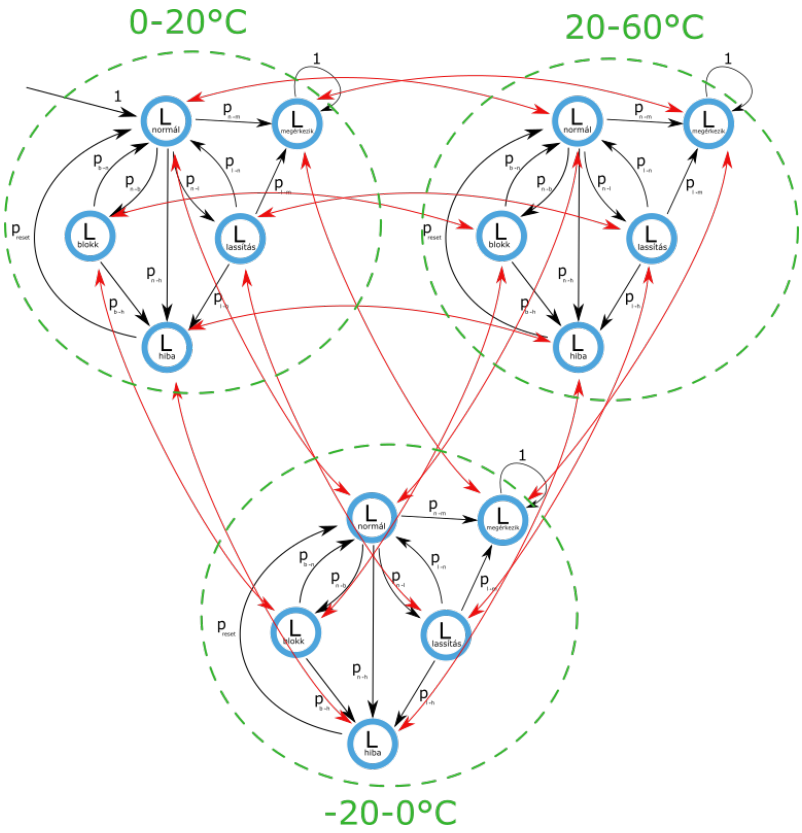
Példa Ugyanis például rengeteg mérnöki problémánál köztük a MoDeS3-nál is egy fontos megvizsgálni a rendszerünk hőmérsékletfüggését és meghatározni milyen tartományokon lehet a különböző hibák valószínűségeit egy adott szint alatt tartani. Ekkor még felmerülhet hogy vegyünk fel a rendszerünkbe párhuzamos állapotokat, melyek a rendszer viselkedését modellezik különböző tartományokon mint az a 2.7[8] ábrán is látható. Viszont ezzel a módszerrel amellett, hogy nagy mértékben leegyszerűsítjük a rendszerről alkotott képünket, de ha szeretnénk új változók felvételekor mint például a vonat súlya, akkor ezen párhuzamos állapotoknak a száma exponenciálisan növekedni fog. Emiatt a futási idő is megengedhetetlenül nagy lesz.

A 2.7 képen látható, hogy a modellt nehéz átlátni, de egyúttal a vizsgálathoz szűkséges idő is drasztikusan megemelkedik.

Definíció 4 (Paraméteres diszkrét idejű Markov-lánc). A $pDTMC := \langle L, s, T, R, H \rangle$ egy diszkrét idejű paraméteres Markov-lánc (parametric Discrete-Time Markov-Chain) [27], ahol $L; s; T$ a DTMC-nél megismert szerepet tölti be $R : (L \times L) \to \mathcal{Q}_V;\ (l_1; l_2) \to f_{(l1;l2)}$ ráta függvény már nem egyszerű valószínűséget rendel a bármely két csúcshoz hanem egy első fokú V-től függő polinomokat, melyek numerikus értéke mindig az adott paraméterek kiértékelésétől függ. ▪

Ezen pDTMC definíció csak a lineáris paraméterfüggőségeket enged meg. Vagyis a tranzíciók valószínűségei az egyes paramétereknek (ha a többi paraméter addig állandó) lineáris függvényei. Ennek oka, hogy a nemlineáris függőségekkel történő számoláshoz jóval bonyolultabb algoritmusok szűkségesek így rájuk csak az 5. fejezetben térek ki.

[8] Az ábrán található piros nyilak a párhuzamos állapotok összetartozását jelölik, ezek nem tranzíciók.



2.7. ábra. Vonat vezérlő állapottere párhuzamos állapotokkal a hőmérséklet alapján

Példa A 2.3 ábrán látható modellt paraméterekkel lehetővé válik a v sebesség illetve a hőmérséklet hatásainak modellezése. A tranzíciók valószínűségeinek a paraméterekkel történő kifejezése a 2.2 táblázatban található.

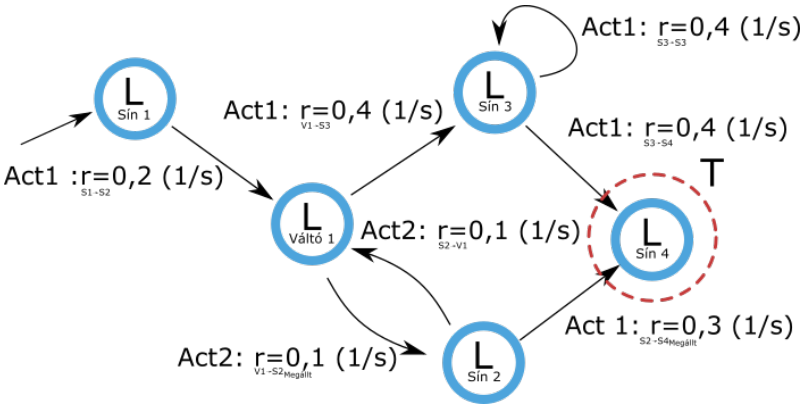
Tranzíciók	Valószínűségei
$p_{\text{normál}\to\text{megérkezik}}$	$0,9 - 0,4 \cdot v - 0,02 \cdot t$
$p_{\text{normál}\to\text{hiba}}$	$0,1 \cdot v + 0,01 + 0,01 \cdot t$
$p_{\text{normál}\to\text{lassít}}$	$0,3 \cdot v + 0,01 + 0,01 \cdot t$
$p_{\text{normál}\to\text{blokk}}$	$0,08$
...	

2.2. táblázat. Tranzíciók valószínűségei

Definíció 5 (Jól definiáltság). Egy $pDTMC := \langle , s, T, R, V\rangle$-t jól definiáltnak nevezünk [27], ha $\forall p \in V$ paraméterek minden lehetséges értéke esetén a rátamátrix által meghatározott tranzíció valószínűségek nulla és egy közé esnek és minden lokációban a kimenő valószínűségek összege 1. ▪

A továbbiakban kizárólag jól definiált Markov-láncokkal foglalkozok.

Definíció 6 (Paraméteres folytonos idejű Markov-lánc). A paraméteres folytonos idejű Markov-lánc [14] $\mu \in pCTMC; \mu := \langle L, s, T, R_\lambda, V\rangle$ a diszkrét idejű paraméteres Markov-láncokhoz hasonlóan definiált, egyedül a $R_\lambda : (L\times L) \to \mathcal{Q}_V$; $f_{(l1;l2)} \in \mathcal{Q}_V$ ráta függvény kapcsán van eltérés hiszen itt a $\mathcal{Q}_V$-beli polinomok kiértékelése a tranzíciók λ rátáit határozzák meg. ▪



2.8. ábra. Útvonaltervezés CTMDP modellezéssel

2.3. Markov döntési folyamatok

A paraméteres Markov-láncok segítségével már rengeteg különböző folyamat modellezhető. Viszont az optimális útvonaltervezés meg egyéb döntés alapú problémák nem formalizálhatóak a fenti eszközökkel. Ugyanis a vonat útvonalát az eddigiektől eltérően mi határozhatjuk meg, és ezáltal lehetőségünk van megvizsgálni, milyen döntések esetén lesz a különböző hibaállapotok elérése minimális vagy épp maximális. Így a következőkben definiáljuk a döntés alapú Markov modelleket melyekkel képesek vagyunk tanulmányozni a nondeterminizmust és annak feloldását, tudatos az aktuális helyzethez alkalmazkodó döntésekkel.

Definíció 7 (Diszkrét idejű Markov döntési folyamat). Diszkrét idejű Markov döntési folyamatok $DTMDP := \langle L, s, T, Act, R \rangle$ (Discrete-Time Markov Decision Process) [3] a DTMC -hez hasonlóan egy lokációhalmaz, egy kiindulóállapot, egy célhalmaz, egy döntéshalmaz és egy döntésmátrix összessége. Ezek közül az első három szerepe megegyezik a DTMC-beli társaikkal, viszont a $R : (L \times Act \times L) \to \mathbb{R}$ 3 dimenziós döntésfüggvény viszont megadja, hogy a különböző döntések esetén mekkora valószínűséggel jutunk el a többi állapotba. Így a T célhalmazba történő eljutás egyszerre függ a véletlentől és a nemdeterminizmustól. ▪

Definíció 8 (Folytonos idejű Markov döntési folyamat). A folytonos idejű Markov döntési folyamat [3] $CTMDP := \langle L, s, T, Act, R_\lambda \rangle$ (Continous-Time Markov Decision Process) nagyban szinte mindenben hasonlít diszkrét idejű társára. Az egyedüli különbség a $R_\lambda : (L \times Act \times L) \to \mathbb{R}$ rátafüggvényben van, mely a CTMC-khez hasonlóan a egy döntéshez tartozó két lokáció közti exponenciális eloszlás rátája. ▪

Példa A MoDeS3 terepasztalon az útvonaltervezés matematikai leírása válik lehetővé folytonos idejű Markov döntési folyamatok segítségével. Ugyanis mint az a 2.8 ábrán látható, a váltók segítségével lehetővé válik a kiindulópont és acél között több útvonalon is eljutni. Ekkor sok esetben mint például a "Sín 1"-es lokációban csak egy lehetséges döntésünk van. Ezzel szemben a "Váltó 1"-esben már dönthetünk, hogy a "Sín 2"-be (Act1 döntés) vagy a "Sín 3"-ba (Act2 döntés) szeretnénk eljutni. Végül érdemes megjegyezni, hogy a folytonos idejű Markov-láncok definíciója megengedi az olyan szituációkat, ahol egy döntés két tranzíciót is lehetővé tesz. Ugyanis például a "Sín 3"-ról továbbmenve elképzelhető, hogy az (egyetlen) Act1-es döntést választva nem a "Sín 4"-véglokációba kerülünk, hanem (műszaki hiba miatt) a "Sín 3"-as lokációban maradunk.

2.4. Tulajdonságok értelmezései Markovi sztochasztikus modelleken

Ebben a szakaszban definiáljuk a dolgozatban vizsgált Markovi modelleken értelmezett tulajdonságokat. Ezek segítségével lehet megadni, hogy modellünket milyen szempontok alapján szeretnénk vizsgálni, mik a számunkra érdekes tulajdonságai a modellnek.

Ezek a tulajdonságok $prop \in \mathbb{R}$ valós számok. Viszont mivel ezek számítása a különböző paraméterek függvényében rendkívül bonyolult. Így a dolgozatom során kizárólag azt vizsgálom (4 fejezet), hogy ezen számok nagyobbak-e egy adott határértéknél. Ugyanis a paraméteres Markov-láncoknál fontos kérdés, hogy a paramétertér mely halmazain lesz nagyobb ezen tulajdonság az adott határértéknél.

2.4.1. Temporális logikák használata

A tulajdonságok definiálása során mindenképpen definiálnunk kell a temporális logikákat [1] [4]. Ezen kifejezések segítségével meg tudjuk adni, hogy a vizsgált tulajdonság a modellekben milyen típusú folyamatokon értelmezett.

Célhalmaz elérése A modellezés során az egyik leggyakoribb kérdés a célállapotok elérése, vagyis hogy a modellezett folyamat során a kezdeti állapotból milyen körülmények között tudunk eljutni egy adott végállapotba vagy végállapotok halmazába. Ez a legtöbb esetben folyamat végét jelenti. A dolgozatomban kizárólag véges méretű és 1 valószínűséggel véget érő folyamatokat vizsgálok. Ezáltal az általam vizsgált folyamatok bizonyos végállapotokat (nyelőket) elérve ér véget.

Definíció 9 (Célhalmaz elérése). Modelljeimben adott $T \subseteq L$ célhalmaz elérése $\Diamond T$ azt a jelöli, hogy a vizsgált folyamat során a kezdeti $s \in L$ kezdő lokációból véges idő illetve lépés után eljutottunk T célhalmaz valamely elemébe. ▪

Elérés során még fontos szót ejteni a $\Diamond C\,U T$ feltételes elérésről, mely során úgy érjük el a T célhalmazt, hogy $C \subseteq L$ lokációhalmazt is érintjük közben. Ennek egy típusa $\Diamond \neg C\,U T$ amikor úgy érjük el a célhalmazt, hogy C lokációit érintjük. A későbbi levezetésekben, ha $C := \{l\};\ l \in L$ csak egy lokációt tartalmaz, a $\Diamond \{s\}\,U T$ helyett a $\Diamond s\,U T$ rövidítéssel élünk. Továbbá amennyiben egy halmazt $n \in \mathbb{N}$ sokszor érintve jutunk el a célhalmazba $\Diamond C^n T$-vel jelöljük. Emiatt ha úgy jutunk el a célhalmazba, hogy $n \in \mathbb{N}$ lokációváltás történik, akkor a $\Diamond L^n T$ jelölést használjuk.

2.4.2. Elérési tulajdonságok

Az a sztohasztikus modellekben az elérésnek számos aspektusát meg lehet vizsgálni. Ezek közül a gyakorlati alkalmazások szempontjából legfontosabb szempontokat mutatom be.

Definíció 10 (Elérési valószínűség). Egy adott μ sztochasztikus modellen $T \subseteq L$ célhalmaz elérésének $P^{\mu}(\Diamond T) \in \mathbb{R}^{+}$ valószínűsége megmutatja, hogy ha ezt a folyamatot folyamatosan újra és újra lefuttatnánk, akkor mekkora relatív gyakorisággal jutnánk el $s \in L$ kezdőlokációból T-be. ▪

Példa Elektronikai eszközeink vizsgálata során például fontos megvizsgálni, hogy a tervezett működés során mekkora valószínűséggel fogunk valamilyen hibaállapotba kerülni.

Habár ez a kérdés folytonos idejű Markov-láncoknál is releváns, mégis felmerül az igény az időkorlátos elérés valószínűségének meghatározására.

Definíció 11 (Időkorlátos elérési valószínűség). Egy adott μ sztochasztikus modellen $T \subseteq L$ célhalmaz elérésének $P^{\mu}(\Diamond T)_{<t} \in \mathbb{R}^{+}$ valószínűsége megmutatja, hogy ha ezt a folyamatot folyamatosan újra és újra lefuttatnánk, akkor mekkora relatív gyakorisággal jutnánk el **legfeljebb** $t \in \mathbb{R}^{+}$ **idő alatt** $s \in L$ kezdőlokációból T-be. ▪

Példa Ezen probléma gyakran felmerül például útvonaltervezés során is. Ekkor az egyes lokációk a megteendő útvonal szakaszait jelölik, és szeretnénk megállapítani milyen valószínűséggel jutunk el a célba az elvárt idő alatt.

2.4.3. Aggregált mérték és kapcsolódó tulajdonságok

Érdemes még megvizsgálni a modellek átlagos viselkedését is vagyis átlagosan mennyi idő alatt tudunk eljutni a célba. Viszont ekkor, ha van egy olyan útvonal egy célhalmaztól különböző nyelő végállapotba, vagyis van esély van rá, hogy nem érkezünk meg, akkor a célhalmaz elérésének várható ideje végtelen lesz. Ennek kiküszöbölése végett a feltesszük, hogy a modellben mindenképpen meg kell érkeznünk az célhalmazba.

Definíció 12 (Feltételes várható idő). Egy adott μ sztochasztikus modellen $T \subseteq L$ célhalmaz elérésének $\mathbb{E}^{\mu}(\Diamond T)_{t} \in \mathbb{R}^{+}$ várható ideje megmutatja, hogy ha ezt a folyamatot folyamatosan újra és újra lefuttatnánk, akkor mekkora annak a $t \in \mathbb{R}^{+}$ időnek az átlaga míg eljutunk $s \in L$ kezdőlokációból eljutunk a T-be. Ekkor feltesszük, hogy $P^{\mu}(\Diamond T) = 1$, vagyis a célhalmaz elérési valószínűsége 1. ▪

Az előzőekben úgy tekintettük, hogy a lokációkban időt töltünk, és vizsgáltuk az egyes lefutások alatt ennek összegyűjtött mértékét. Ehhez hasonlóan definiáljuk ennek egy "általánosítását", az aggregált mértéket, melyet a lefutás során folyamatosan gyűjtünk, minden egyes lokációba történő belépéskor.

Definíció 13 (Aggregált mérték). Az aggregált mérték $F_{reward} : L \to \mathbb{R}$ a μ sztochasztikus modellünk lokációhamazán értelmezett valós értékű függvény. Ez a lefutások alatt a lokációkba történő belépésekkor folyamatosan aggregálódik. ▪

Egy adott célhalmaz eléréséig gyűjtött aggregált mértéket $\blacklozenge T$-vel jelölöm, mely egy új temporális logikai operátor. A $\Diamond T$ elérés operátorhoz hasonlóan használjuk a feltételeket. Így $\blacklozenge C\, U T$, mely a T-ig C-n keresztül összegyűjtött aggregált mértéket jelöli [4].

Definíció 14 (Feltételes várható aggregál mérték). Egy adott μ sztochasztikus modellen $T \subseteq L$ célhalmaz elérésének $\mathbb{E}(\blacklozenge T | \Diamond T)^{\mu}_{t} \in \mathbb{R}^{+}$ várható aggregált értéke megmutatja, hogy ha ezt a folyamatot folyamatosan újra és újra lefuttatnánk, akkor mekkora annak a lefutások alatt összegyűjtött aggregált mérték átlaga míg eljutunk $s \in L$ kezdőlokációból eljutunk a T-be. Ekkor feltesszük, hogy $P^{\mu}(\Diamond T) = 1$, vagyis a célhalmaz elérési valószínűsége 1. ▪

2.5. Ütemező

A különböző tulajdonságok ismertével, magától jön a kérdés, hogy egy Markov-döntési folyamatban, hogyan kell optimálisan választani a lehetséges döntések közül a különböző

lokációkban hogy maximalizáljuk vagy épp minimalizáljuk ezen tulajdonságok bekövetkezési valószínűségét. Például hogyan lehet olyan útvonalat választani, hogy minimális legyen a különböző hibaállapotokba jutás valószínűsége. Ezen döntéseknek a modellezését ütemezőkkel végezzük. Ezen ütemezőfk optimális meghatározása egy rendkívül széles szakirodalommal rendelkezik [1][3] így ezek részletes ismertetése meghaladja dolgozatom kereteit.

Definíció 15 (Ütemező). Diszkrét idejű Markov döntési folyamat lokációin értelmezett $\omega = \langle \mu; prop \rangle$ ütemező [3] [1] egy Markov döntési folyamaton értelmezett függvény, mely minden egyes lokációhoz és az odavezető úthoz hozzárendeli a folyamatba saját *prop* teljesülése szempontjából optimális döntést. ▪

3. fejezet

Kapcsolódó munkák

Ezen fejezetben bemutatom az eddigi megoldásokat az általam megoldott problémára illetve összehasonlítsam őket az erőforrás igény, a skálázhatóság, az alkalmazhatóságát és pontosság szempontjából. Ezen kívül bemutatom az általam felhasznált szoftvereszközöket, különös tekintettel a Markov döntési folyamat megoldókra, továbbá bemutatom a Markov-lánc alapú modelleket és segítségükkel készített vezérlőprogramoknak milyen alkalmazásai vannak az iparban.

3.1. Algoritmusok

3.1.1. Quatmann-féle paraméter ellenőrzés

A paraméteres modellellenőrzésre jelenleg ez a leggyorsabb algoritmus [27]. Viszont gyorsasága ellenére egyedül az elérési valószínűséget képes kezelni kizárólag diszkrét idejű modellek esetén. Mivel ezen algoritmus működési elvét a saját megoldásaimban is hasznosítom, így a 4 fejezetben külön részben tárgyalom.

3.1.2. Lokáció elimináció

Diszkrét idejű paraméteres Markov-láncok esetén ezen algoritmus számít a legelterjedtebb. Ezt a megoldást használják a Param [17] és Prophesy [9] eszközökben is. Lényege, hogy a tranzíciók rátáit a paraméterek polinomjainak hányadosa határozza meg.
Ekkor ezen módszer segítségével egyesével eltávolítjuk (eliminájuk), miközben a többi tranzíciót úgy módosítjuk, hogy az elérési valószínűség ne változzon. Ezt a műveletet egészen addig végzik míg csak kettő (egy kezdeti és egy cél) állapot marad. Ennek segítségével ki lehet fejezni az elérési valószínűséget polinom per polinom alakban. Ezután egy speciális algebrai azonosságok automatizált megoldására szakosodott programmal, egy úgynevezett SMT oldóval [8] egy iteratív algoritmussal [17] határozzák a tulajdonságokat teljesítő paramétertartományokat. Ugyanis a paramétertart

Lokáció elimináció feltételes valószínűségekhez Ezen algoritmus az előbbi szekcióban bemutatott lokáció elimináció kiterjesztése, feltételes elérési valószínűség számolásához [9]. Vagyis képes meghatározni diszkrét idejű Markov-láncok esetében, hogy mekkora valószínűséggel képes elérni a célhalmazt úgy, hogy közben érint egy másik lokációhalmazt is.

3.1.3. Mohó algoritmus CTMDP-khez

A mohó folytonos idejű Markov döntési folyamatokon értelmezett mohó algoritmus [3] segítségével lehetséges megtalálni a legjobb ütemezőt időkorlátos elérési valószínűség számításához. Ezen módszer bár nem képes a teljes paramétertérre garanciát biztosítani, de amennyiben a paraméterterünkben nagy számú pontot veszünk fel és ezeknek értékeit döntésekhez rendeljük képesek vagyunk egy becslést adni, hogy milyen paraméterek teljesítik a vizsgált követelményeket.

3.1.4. Elosztott ütemező alapú lokáció eliminációl

Ezen megoldás egy a PARAM-hoz [17] [6] készített bővítés, melynek segítségével képesek vagyunk elosztott ütemezőkkel visszavezetni a paramméteres lépészám korlátos elérést lokáció eliminációra. Ekkor a mohó algoritmussal ellentétben itt a paramétertér folytonos, de az idő kvantált, így ez se ad pontos eredményt folytonos idejű problémákra.

3.1.5. Mote Carlo módszer

Az alkalmazások után történő kutatásom során tapasztaltam, hogy a gyakorlati megoldások során nagyon sok esetben [30] [19] [33] a Monte Carlo módszert alkalmazzák. Ekkor a vizsgálandó tulajdonságát úgy becslik, hogy nagy számú szimulációt végeznek, majd azok eredményét összesítik.

3.2. Toolok és egyéb szoftvercsomagok

3.2.1. PRISM Model Checker

Ha Markov-lánc alapú modellezés ről van szó, akkor PRISM Model Checker [24] az egyik legelterjedtebb megoldás. Használatát elősegíti a PRISM modellezési nyelv [25] [7], mellyel intuitív módon szövegesen le tudjuk írni Markovi modellünket, továbbá külön fájlban értelmezni tudunk rajta tulajdonságokat is. A használatot multiplatform grafikus kezelőfelület könnyíti, de szolgáltatásait parancssorból is igénybe vehetjük. Segítségével képesek vagyunk vizsgálni folytonos és diszkrét idejű Markov láncokat, diszkrét és folytonos idejű Markov döntési folyamatokat és egy kiterjesztésnek [2] köszönhetően sztohasztikus játékokat is. A vizsgálat szempontja lehet állapot elérés valószínűsége, várható aggregált mérték, vagy más szochasztikus temporális logikai tulajdonság [temp logika], de lehetséges kombinált tulajdonságokat is beállítani. Ám paraméteres modellek közvetlen kezelése nem lehetséges.

Rugalmas használata és egyszerű kezelhetősége miatt munkámban a prototípus implementáció elkészítésekor a PRISM Model Checkert használtam fel háttér megoldóként.

3.2.2. Storm Modellchecker

A Storm Modellchecker [10] program a legújabb modell ellenőrző eszköz, és habár nem rendelkezik grafikus kezelőfelülettel, de számos másik modellezőnyelvet is támogat mint például a Jani [5], GSPNs [11], de számos másik a dolgozat szempontjából lényegtelen gráfleíró nyelvet is támogat. Illetve szolgáltatásait Python [28] könyvtár segítségével is igénybe lehet venni. Viszont paraméterekkel történő közvetlen számolásra nem alkalmas.

3.2.3. PARAM

Ezt a szoftvert [17] kifejezetten diszkrét idejű paraméteres Markov-láncokhoz fejlesztették ki. Lényege, hogy a Hybrid engine-t és a hozzá tartozó PRISM nyelvet kiterjesszék paraméteres modellekre és lehetőséget biztosítsanak elérésre, időkorlátos elérésre illetve várható aggregált mértékre paraméteres ellenőrzést végezni. Működésének alapja az állapot elimináció illetve kiterjesztései, mely egyszerű felépítése ellenére rendkívül erőforrás-igényes és rosszul skálázódik. Továbbá ki kell emelni, hogy ezen szoftver nem képes folytonos idejű modelleket kezelni.

3.2.4. Prophesy

Ezen szoftver [9] nagy mértékben hasonlít a PARAM-ra [17] ugyanis szintén az állapot elimináción alapul és szintén csak diszkrét idejű modellek kezelésére alkalmas. Érdekessége, hogy képes feltételes valószínűségekkel számolni, vagyis képes olyan kérdéseket megválaszolni mint például mekkora valószínűséggel jutunk el egy lokációk adott T célhalmazába, úgy hogy közben egy másik C lokációhalmazt érintünk.

3.2.5. MDP Matlab toolbox

A MDP Matlab Toolbox [20] segítségével lehetővé válik Markov döntési folyamatokhoz adott célhalmaz elérése szempontjából optimális stratégiákat keresni. Habár Matlab környezetben egyszerű a használata, de nem támogatja tulajdonságok közvetlen vizsgálatát.

3.2.6. MRMC

Az MRMC (Markov Reward Model Checker) [23] egy diszkrét és folytonos idejő Markov-láncokhoz aggregált mérték számítására specializálódott programcsomag. Ez a várható aggregált mértéket és az időintervallum alatt gyűjtött aggregált mértéket képes számolni, de paraméterek kezelésére nem alkalmas.

3.2.7. Infamy

Az Infamy [16] egy érdekes kis szeglete a Markovi modell ellenőrző programoknak, ugyanis képes végtelen nagy folytonos idejű Markov-láncokat vizsgálni.

3.3. Szoftverek összehasonlítása

Ezen részben egy táblázat formájában hasonlítom össze az eddig bemutatott megoldásokat saját megoldásommal. Összehasonlítás folyamán döntő szempont volt, hogy milyen típusú tulajdonságokkal képesek számolni valamint képesek-e kezelni a folytonos időt.

Programok	Folytonos idő	$P(\Diamond T)$	$\mathbb{E}(\blacklozenge T)$	$\mathbb{E}t(\Diamond T)$
Megoldásom	van	van	van	van
Quatmann programja	nincs	van	nincs	nincs
Param	nincs	van	van	nincs
Prophesy	nincs	van	van	nincs

3.1. táblázat. Paraméteres megoldók

Mint az a 3.1 táblázatból is látszik, kutatásom alapján egyedül az én megoldásom képes folytonos időben időzített elérési valószínűség és várható idő paraméteres számolására, továbbá az eddigi megoldásokkal ellentétben [17] az általam fejlesztett programom jól skálázható és elosztott számítógépes rendszereken is megvalósítható. A programok tejes körű összehasonlítása a 3.2 táblázatban található

Programok	Idő	Modell típusok	Paraméterek	Tulajdonságok
Saját programom	Folytonos	Markov-láncok	Van	$\Diamond T; \blacklozenge T; \mathbb{E}_t$
Quatmann programja	Diszkrét	MC és MDP	Van	$\Diamond T$
Prizm Model Checker	Mind	MC és MDP	Nincs	$\Diamond T; \blacklozenge T$
Storm Model Checker	Mind	MC és MDP	Nincs	$\Diamond T; \blacklozenge T$
Param	Diszkrét	MC és MDP	Van	$\Diamond T; \blacklozenge T; \mathbb{E}_t$
Prophesy	Diszkrét	MC és MDP	Van	$\Diamond T \vert \Diamond C$
Infamy	Diszkrét	MC és MDP	Nincs	$\Diamond T \& \vert L \vert = \inf$
Matlab MDP Toolbox	Diszkrét	MDP	Nincs	ω
MRMC	Mind	MC	Nincs	$\blacklozenge T$

3.2. táblázat. Programok összehasonlítása

3.4. Alkalmazások

Ezen szakaszban mutatom be, hogy az iparban a Markovi modelleknek milyen alkalmazásterületei vannak. Mutatva, hogy az általam fejlesztett és ahhoz hasonló megoldások ipari potenciálját, és jövőbeli alkalmazási lehetőségeit.

3.4.1. Szélerőmű optimalizálás

A Markovi modelleket széles körben alkalmazzák a megújuló energia termelő és elosztó rendszerekben, különös tekintettel a szélerőmű telepekre [Wind Speed Behavioral Modeling for Economical Energy Generation using Windmills] [Stochastic modeling to represent wind power generation and demand in electric power system based on real data]. Hisz a hagyományos erőművekkel ellentétben, a termelt árammenyiség folyamatosan ingadozik. Az időjárástól függően hol teljesen megszűnik a termelés, hol a túltermelés túlterheli a hálózatot. Így ezen problémák megoldásához elengedhetetlenül szükséges a szél és ezáltal a termelés ingadozásainak pontos modellezése, és viselkedésének előrejelzése. Így a különböző tipikus állapotait a szélnek Markov-láncbeli lokációknak feletetik meg.

3.4.2. Hibrid

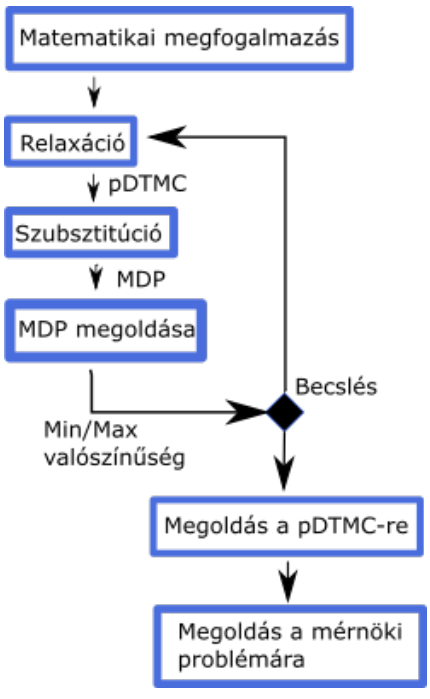
A hibrid elektronikákban, melyekben egyszerre több forrásból is jön energia (például hibrid autók esetén a benzines motor, valamint az akkumulátor), az optimális működés meghatározásához elengedhetetlenül szükséges az energiafelvétel előrejelzése. Ezen probléma megoldásához is a Markov-láncok egy rendkívül hatékony eszköznek bizonyulnak [31]. Az alábbi cikkben például a vizsgált jármű sebességét és gyorsulását diszkrét értékpárokra osztották, majd azokat Markov-láncbeli lokációknak feleltették meg.

4. fejezet

Új algoritmusok paraméteres verifikációra

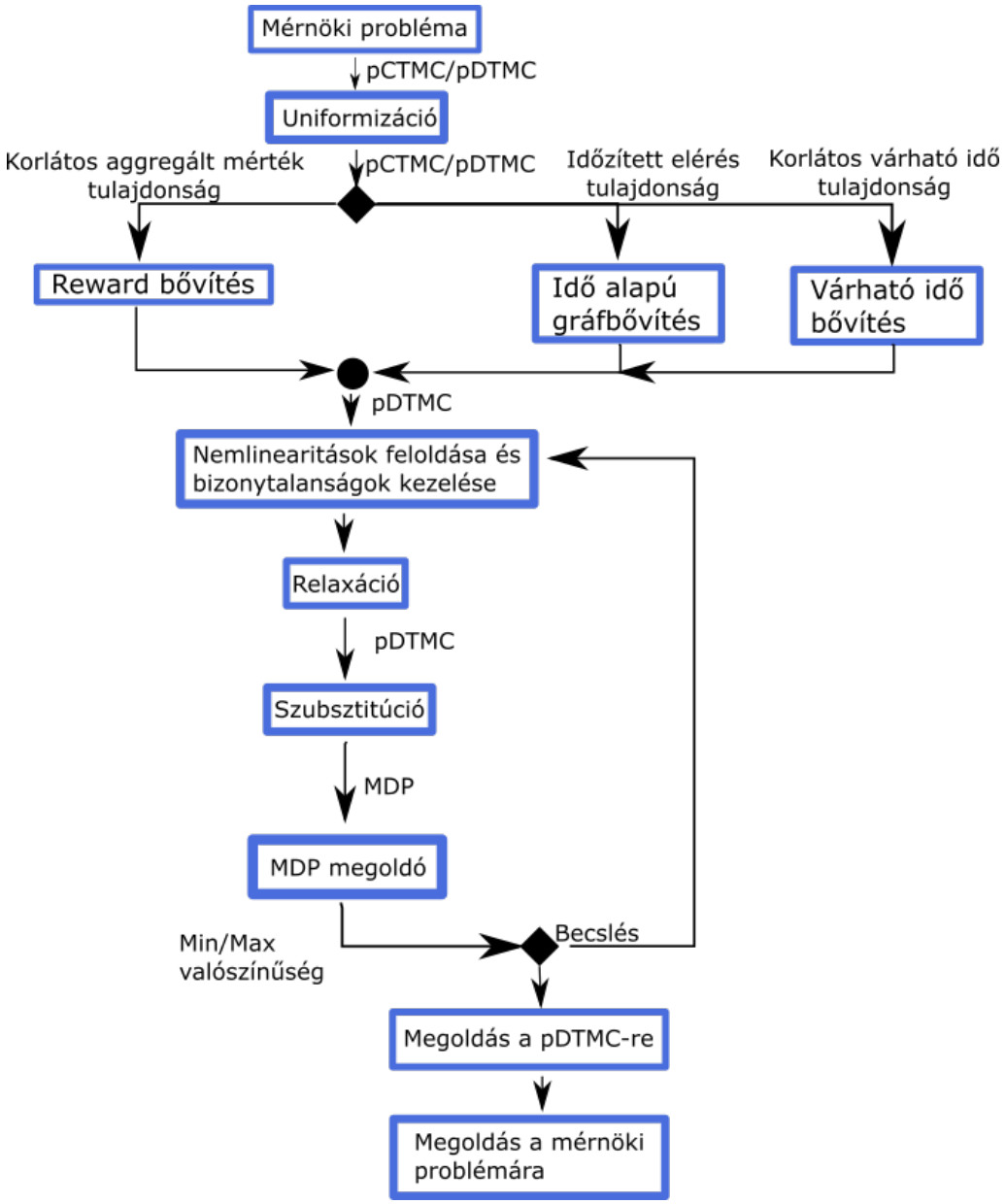
Ebben a fejezetben mutatom be kutatási munkám lényegét, az általam fejlesztett algoritmusokat és a működésük helyességét és optimalitását igazoló bizonyításokat. Mindezek előtt azonban bemutatom a Qatmann paraméteres ellenőrző algoritmusa és helyességét igazoló bizonyításokat, mely munkám elméleti alapját képezi.

Ma már szinte az összes paraméteres Markov-láncokon értelmezett tulajdonság kiértékeléséhez létezik explicit képlet és megoldóalgoritmus [Prizm Modellchecker,Storm Modellchecker]. Viszont ezek futási ideje és erőforrás igénye a bemenet méretétől exponenciálisan függnek. Továbbá elosztott rendszereken történő futtatás sem lehetséges így nem használhatóak nagy méretű modellek analizálásához. Ezen problémákra ad választ diszkrét időben Qatmann-féle paraméter ellenőrző algoritmus. Megoldásának újszerűsége, hogy a paraméteres Markov-láncok problémáját egy iteratív becslés segítségével visszavezeti a Markov döntési folyamatokra, melyekhez a paraméteres Markov-láncokkal ellentétben már számos hatékony megoldóalgoritmus létezik.



4.1. ábra. Quatmann-féle algoritmus működése

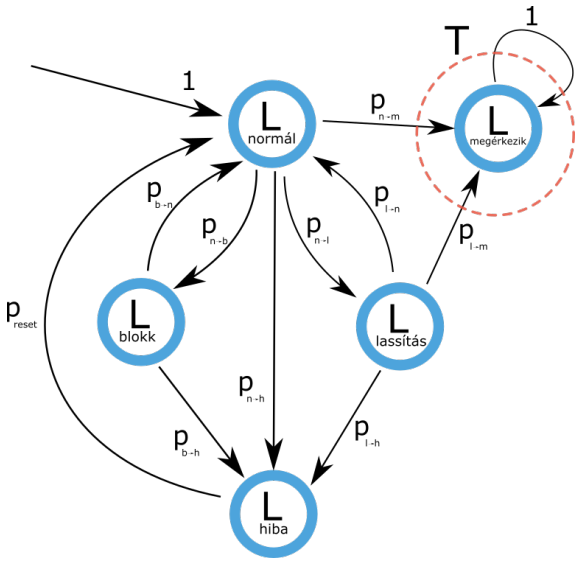
Viszont ezen algoritmus csak egy típusú tulajdonságot tud kezelni (elérési valószínűséget) valamint egyedül lineáris paraméter függvényeket írhatják le a paraméterek és a valószínűségek közti összefüggést, és végül nem képes a folytonos idővel számolni. Erre jelent megoldást az algoritmusom, mely a Quatmann algoritmusa továbbfejlesztve és kiterjesztve képes bármilyen korlátos paraméter függvényt kezelni és folytonos időtartománybeli tulajdonságok ellenőrzésére. Megoldásom lényege, hogy a nemlinearitások feloldása után a folytonos idejű paraméteres Markov-láncot uniformizálom, majd egy a vizsgált tulajdonságtól függő transzformáció után a Qatmann-féle gondolkodás alapján visszavezetem egy Markov döntési folyamatra, majd azt megoldva fokozatosan becsüljük az eredeti Markov-láncot.



4.2. ábra. Algoritmusom működése

4.1. Matematikai megfogalmazás

A mérnöki problémák pontos matematikai leírása rendkívül nehéz feladat, mint azt a 3.4 szakaszban bemutattam, számos különböző megközelítés létezik rá. Ezen lépésre adott megoldásom látható a 2.2.1 szakaszban, ahol a MoDeS3 projektbeli állapotgépet egy Markov-lánchoz rendeltük hozzá meg. Mivel ezen lépés a dolgozatban bemutatott összes algoritmus esetén megegyezik így ezen lépést már külön-külön nem tárgyalom. Továbbá könnyebb megértést elősegítendő a következő algoritmusokat a 4.3 ábrán átható, korábban említett MoDeS3 vonatvezérlő állapotgépéből készült Markov láncon keresztül fogom bemutatni.



4.3. ábra. A vonat állapotgépének Markov-lánca

4.2. Quatmann algoritmusa diszkrét idejű paraméteres Markov-láncokra

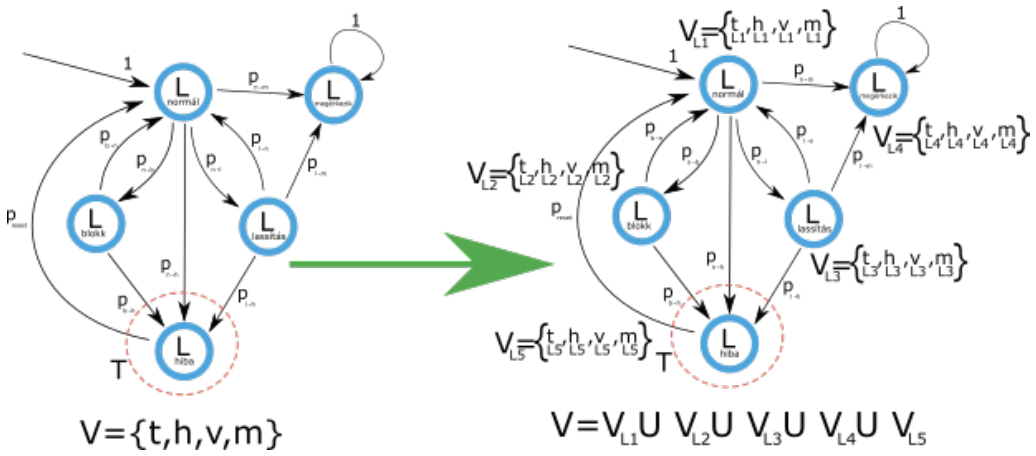
A paraméter tér ellenőrző algoritmusoknak egy nehéz problémát kell legyőzniük, a tranzíciók paraméterfüggőségeit. Ugyanis ha megváltoztatunk egy paraméter értékét akkor nagyon nehéz átlátni hatását a különböző tulajdonságok teljesülésére.

4.2.1. Relaxáció

Ez egy paraméteres Markov-láncok halmazán értelmezett függvény, mely feloldja a különböző lokációk paraméterfüggőségeit és minden egyes csúcshoz létrehoz lokális paramétereket. Ezáltal a különböző lokális paraméterek hatása sokkal könnyebben kiszámolható, ugyanis mint azt a Paramétermonotonitási tételben belátom a vizsgált tulajdonság teljesülési valószínűsége monotonan fog függni ezen paramétertől. Így habár a monotonitás irányát nem ismerjük, de elég csak megvizsgálni, a paraméter alsó és felső korlátját.

Definíció 16 (Relaxáció). A relaxáció $rel : \mu_1(L, s, T, V, R) \in pDTMC \rightarrow \mu_2(L', s, T, V', R') \in pDTMC$, mely μ_1 lokációit teljesen érintetlenül hagyja viszont $V' = x_i^l | x_i \in V; l \in L$ [1] és habár R és R' rátamátrixban lévő polinomok paraméterei vál-

[1] l a felső indexben az adott lokációhoz való tartozást jelenti.



4.4. ábra. Relaxáció alkalmazása a példa modellen

tozatlanok de paramétereik megváltoznak $f'_{(l1;l2)} \in R' : (l_1; l_2) \in (L \times L) \to f_(l1; l2) \in \mathcal{Q}_{V_l}$. Így minden egyes $f'_{(l1;l2)}$ R'-ben lévő polinom saját V_{l1} lokális paraméterén értelmezett.

$$f'_{(l1;l2)}(V_{l1}) = f_{(l1;l2)}(V) \tag{4.1}$$

És ekkor a relaxált paramétertér az eredeti pDTMC lokális paramétertereinek lokációjukkal vett Descartes szorzatuk uniója lesz, mint az a 4.2 egyenleten is látható.

$$V' = \bigcup_{l \in L} V_l \times l \tag{4.2}$$

.

Tétel 1 (Paramétermonotonitás). Ha $P^{rel(\mu)}(\Diamond T)$ valamilyen relaxált $\mu \in pDTMC$ diszkrét idejű paraméteres Markov-láncon értelmezett tulajdonság teljesülésének valószínűsége akkor ha $x \in V'$ ezen valószínűség monoton függ x^{li}-től ($l_i \in L$). .

4.2.1.1. Bizonyítás

A bizonyítás indirekt, vagyis feltesszük, hogy valamely relaxált $rel(\mu)$ diszkrét idejű paraméteres Markov-lánc, ahol $\mu \in pDTMC$ valamely $li \in L$ lokációjában létezik egy olyan $x^{li} \in V_{li}$ paraméter, melytől a T célhalamaz elérésének valószínűsége nem monotonan függ. Vizsgáljuk meg az li-ből a $(\Diamond T)$ tulajdonság teljesülésének a $P_{li}^{rel(\mu)}(\Diamond T)$ valószínűségét.

$$P_{li}^{rel(\mu)}(\Diamond T) = \sum_{l \in L} f_{(li;l)} \cdot P_l^{rel(\mu)}(\Diamond T) = \tag{4.3}$$

$$= \sum_{l \in L} f_{(li;l)} \cdot (P_l^{rel(\mu)}(\neg \Diamond T \, U li) \cdot P_{li}^{rel(\mu)}(\Diamond T) + P_{li}^{rel(\mu)}(\neg\{l_i\}\Diamond T)) = \tag{4.4}$$

$$= \sum_{l \in L} f_{(li;l)} \cdot P_l^{rel(\mu)}(\neg \Diamond T \, U li) \cdot P_{li}^{rel(\mu)}(\Diamond T) + \sum_{l \in L} f_{(li;l)} \cdot P_{li}^{rel(\mu)}(\neg\{l_i\}\Diamond T) \tag{4.5}$$

Ekkor nem csináltunk mást csak szétbontottuk az l lokációkból az elérési valószínűséget két részre:

- Visszajutunk az l_i-be T-t nem érintve
- Eljutunk T-be anélkül, hogy l_i-t érintenénk

Ezután az átláthatóság kedvéért vegyünk fel két ($x^l i$-től függő és paraméterintervallumán értelmezett) segédfüggvényt, f-t és g-t.

$$f(x^l i) = \sum_{l \in L} f_{(li;l)} \cdot P_l^{rel(\mu)}(\neg \Diamond T \, U li) \tag{4.6}$$

$$g(x^l i) = \sum_{l \in L} f_{(li;l)} \cdot P_{li}^{rel(\mu)}(\neg \{l_i\} \Diamond T) \tag{4.7}$$

Korábbiakban már kijelentettük, hogy mivel $\mathcal{Q}_{\mathcal{V}}$-ben kizárólag elsőfokú polinomokat (lineáris) engedünk meg és mivel első fokú polinomok összege is elsőfokú lesz így mind f, mind g lineárisan fog függeni x^{li}-től.

Így segítségükkel ezen elérési valószínűség már egyszerűen felírható.

$$P_{li}^{rel(\mu)}(\Diamond T) = f \cdot P_{li}^{rel(\mu)}(\Diamond T) + g \tag{4.8}$$

Így már a T elérési valószínűsége kifejezhető x^{li} függvényeként.

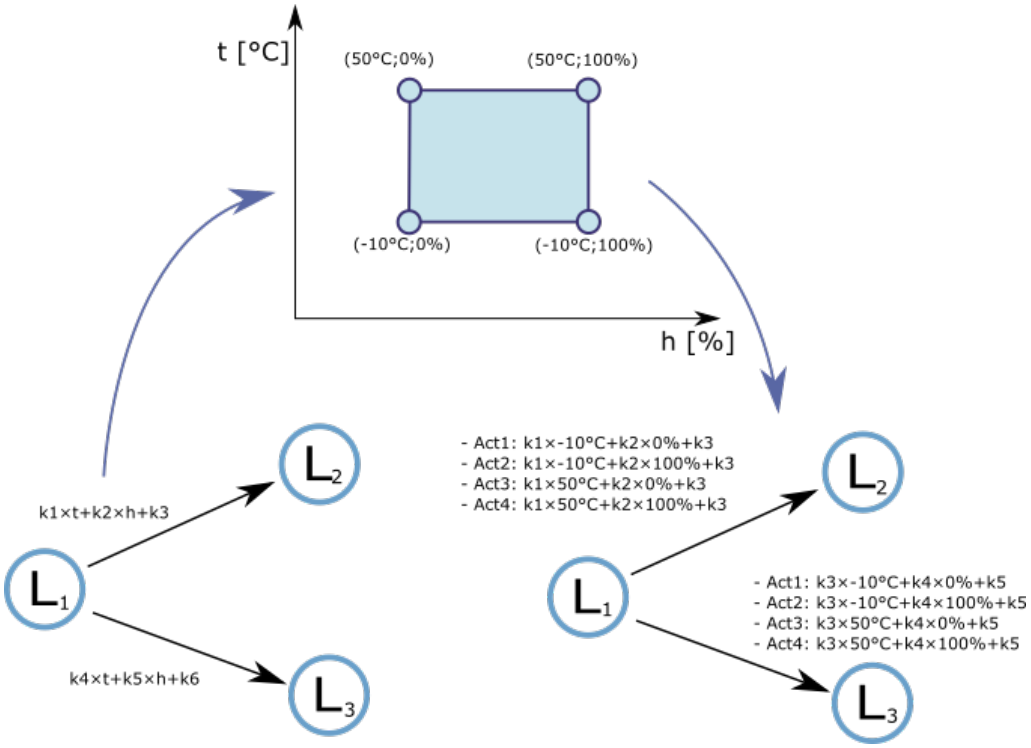
$$P_{li}^{rel(\mu)}(\Diamond T) = \frac{g}{1 - f} \tag{4.9}$$

Itt észre kell vennünk, hogy mind a számláló, mind a nevező lineáris függvény. Melyből következik, hogy deriváltja vagy csak pozitív vagy csak negatív. Továbbá ezen függvény nem tarthat a végtelenbe $1 - f(x^l i) \neq 0$, hisz a tartomány jól definiált és a valószínűség 0 és 1 közé esik. Így beláttuk hogy $P_{li}^{rel(\mu)}(\Diamond T)$ valószínűség monoton függ x^{li}-től vagyis ellentmondásra jutottunk és az indirekt bizonyítás teljes lett.

4.2.2. Szubsztitúció

Most, hogy a relaxáció segítségével képesek vagyunk feloldani a lokációk paraméterfüggőségeit, lehetőség nyílik már a tulajdonságok valószínűségeinek becslésére is a szubsztitúció segítségével. Ezen lépés lényege, hogy mivel egy adott lokációban a lokális paramétertérnek a határait kell megvizsgálni. Így a relaxált Markov-láncot megfeleltetjük egy Markov döntési folyamatnak, melyben a különböző döntések, a lokális paramétereknek a különböző szélső értékeikre történő állítását jelenti.

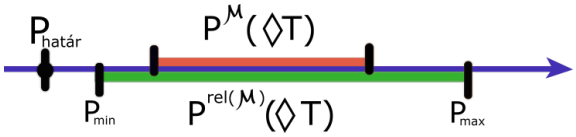
Például mint az a 4.5 ábrán is látszik, ha egy lokációban a tranzíciók valószínűségei a ($t \in [-10°C; 70°C]$) hőmérséklettől és a ($h \in [0\%; 100\%]$) relatív páratartalomtól függenek (és más a modellen értelmezett paraméter nem hat rájuk) akkor a szubsztitúció során négy döntés fog tartozni ezen lokációhoz, mely során a kimenő tranzíciók $\mathcal{Q}_{\mathcal{V}^{\updownarrow}}$ paraméteres polinomjaiba behelyettesítem a lokális paramétertartomány szélső értékeit. Ezt úgy érdemes szemléltetni, hogy amennyiben a lokális paraméterteret egy derékszögű koordinátarendszerben ábrázoljuk akkor ezen térrész egy téglalap lesz. Ennek csúcsait feleltetjük meg a döntéseknek és a koordinátáit helyettesítjük be a kimenő tranzíciók polinomjaiba. Így például a jobb felső sarokból a (50°C;0%) koordinátákon lévő pontból lesz az Act3-as döntés, mint az az ábrán is látható.



4.5. ábra. Szubsztítúció

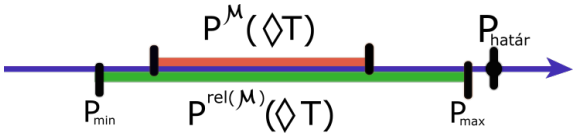
4.2.3. Az iteratív becslés

A szubsztitúció során kapott Markov döntési folyamat segítségével meg tudjuk vizsgálni a relaxált Markov-lánc esetén az elérés maximumát és minimumát, mely becsli az eredeti modellt. Így mint az a 4.6 ábrán is látható a relaxált Markov-lánc esetén a célhalmaz elérésének valószínűsége egy tágabb intervallumon helyezkedik el mint az eredeti modell esetében, így ha kisebb a megadott határ valószínűség mint relaxált intervallumhoz tartozó alsó határértek akkor biztosak lehetünk benne, hogy a vizsgált paramétertéren az elérési valószínűség biztosan nagyobb lesz mint a határ.



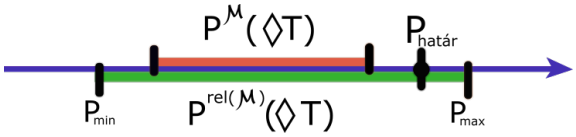
4.6. ábra

Ha a határ valószínűség nagyobb mint a relaxált modell elérési valószínűségének maximuma akkor pedig a helyzet hasonló, mint az a 4.7 ábrán is látható. Ekkor megállapíthatjuk a vizsgált paramétertéren az elérés valószínűsége biztosan kisebb lesz mint az adott határ.



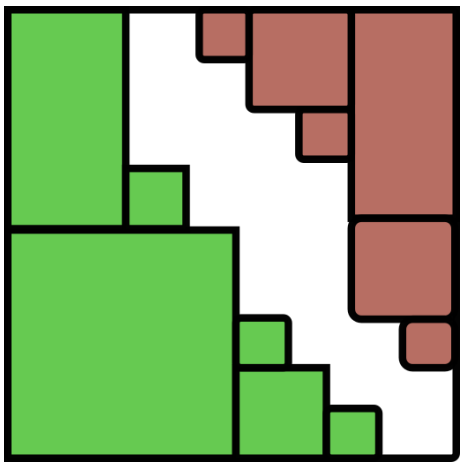
4.7. ábra

Természetesen ha a határ valószínűség a relaxált Markov-lánc intervallumán belül található valahol, mint az a 4.8 ábrán is látható, akkor nem tudunk eredménnyel szolgálni és a paramétertér pontosabb vizsgálata szűkséges.



4.8. ábra

Ebben az esetben a [Param]-ban bemutatott eljárást kell alkalmaznunk. Ekkor fogjuk a paraméterterünket és kettévágjuk valamely paraméter mentén és mindkét paramétertérre elvégezzük ezen becslő módszert iteratívan míg a paramétertér kívánt hányadát fel nem térképezzük. Ekkor egyre több egyre kisebb paraméterteret vizsgálunk folyamatosan növekvő lefedettséget elérve, mint az a 4.9 ábrán is látható.



4.9. ábra. Iteratív becslés az állapottérben

Az alábbi ábrán zöld színnel azokat a paramétertartományokat jelöltem, melyek esetén az elérési valószínűség kisebb míg pirossal azokat a tartományokat ahol nagyobb ezen valószínűség. Ezen módszer segítségével egyszerűbb sztochasztikus modelleket már néhány tíz iteráció után a paramétertér 95%-os lefedettségét tudjuk elérni [27].

4.3. Új algoritmusok paraméteres verifikációra

Dolgozatomnak ebben a részében mutatom meg, hogy az eddigi megoldásokat, hogyan fejlesztem tovább egyre komplexebb problémák megoldásához. Így ebben a részben bemutatom a kutatásom során kidolgozott algoritmusaimat, valamint a helyességüket igazoló bizonyításaimat. Az következőekben az algoritmus bemutatását a 4.2 ábra alapján végeztem.

4.3.1. Reward bővítés

Bár ebben a részben még kizárólag diszkrét időben dolgozok, de mégis egy rendkívül fontos problémát oldok meg, mely későbbi algoritmusaim alappillére, a korlátos aggregált mértékkel történő számítás. Ekkor a különböző lokációkhoz hozzárendelünk egy ú.n.

aggregált mértéket [2]

$$F_{reward} : L \to \mathbb{R} \tag{4.10}$$

Melyet a paraméteres Markov-láncon történő végighaladás során folyamatosan gyűjtünk és minden új lokációba történő belépés során az általunk gyűjtött mértékhez az új lokáció mértéke hozzáadódik (aggregálódik).Így a vizsgálandó probléma, hogy egy adott célhoz történő megérkezésig gyűjtött mérték várható értéke [3] nagyobb lesz-e mint egy adott határérték.

Ekkor feltételezzük, hogy $P(\Diamond T) = 1$, vagyis előbb vagy utóbb de biztosan megérkezünk a célba. Amelyből következik, hogy ha minden lokáció elérhető véges időn belül a kezdő lokációból, akkor minden lokációból is biztosan elérhető a célhalmaz.

$$(\forall l \in L; P_s(\Diamond\{l\}) = 1) \to P_l(\Diamond T) = 1 \tag{4.11}$$

Példa Ezekre az egyik legjobb példa az anyagok öregedése, fáradása [Anyagtudomány]. Hisz a folyamatos terhelés és különböző irányú erőhatások hatására a fémekben [4] az atomok fokozatosan elmozdulnak eredeti helyükről, a szilárdságot gyöngítve egészen a tönkremenetelig illetve az alkatrészek strapabíróságát a ciklusszámmal is jellemzik, mely az átlagos tönkremenetelig megtett mechanikai fordulatok és egyéb erőhatások száma.
Ezáltal a (MoDeS3 terepasztalon történő) útvonaltervezés során, mivel korábbi mérési adatokból rendelkezésre áll, hogy adott útvonal szakasz megtétele során hány darab ciklus következik be így, lehetséges megállapítani, hogy a tervezett út során a kívánt határérték alatt tudjuk-e tartani az alkatrészek öregedését.

Algoritmus Quatmann algoritmusa kiterjeszthető az aggregált mértékkel történő számolásra, hisz a lokációkban lévő mérték nem függ a paraméterektől így a paramétermonotonitási 1 tétel könnyedén kiterjeszthető. Így mind a reaxációs mind a szubsztitúciós lépés változatlan egyedül az iteratív becslés során a szubsztitóció segítségével generált Markov döntési folyamatokon az aggregált mértéket értelmezni. Így az ütemező segítségével azt kell egnézni, hogy az aggregált érték mikor lesz maximális és inimális. [5]

Tétel 2 (Paramétermonotonitási tétel aggregált mértékre). Ha $rel(\mu) \in pDTMC$ relaxált Markov lánc és F_{reward} μ lokációin értelmezett aggregált mérték akkor minden $l \in L$ lokáció minden $x^l \in V_l$ lokális paraméterére igaz, hogy $\mathbb{E}(\blacklozenge T)^{rel(\mu)}$ monoton függ x^{li}-től.

Bizonyítás Indirekten feltesszük, hogy létezik, olyan $l_i \in L$ lokáció amelynek létezik olyan $x^{li} \in V_{li}$ lokális paraméterére melytől a várható aggregált mérték nem monotonan fog függeni.
Ezután felírom l_i-ből a várható aggregált mértéket kimenő tranzíciók segítségével.

[2] Az angol szakirodalomban reward [Optimal computation for MC-s]

[3] Rövidítve várható aggregált érték.

[4] (és más mechanikai szerkezet kialakítására használt anyagokban)

[5] Ennek megvalósításához csak át kell állítani az MDP megoldót, hogy várható aggregált mérték alapján számoljon, mely ma már ezen programok alapfunkciói közé tartozik [Prizm],[Storm] [Matlab toolbox].

$$\mathbb{E}(\blacklozenge T|\lozenge T)_{li}^{rel(\mu)} = F_{reward}(l_i) + \sum_{l\in L} f_{(li;l)} \cdot \mathbb{E}(\blacklozenge T|\lozenge T)_{l}^{rel(\mu)} = \tag{4.12}$$

$$= F_{reward}(l_i) + \sum_{l\in L} f_{(li;l)} \cdot (P_{l}^{rel(\mu)}(\neg\lozenge T\, U li) \cdot P_{li}^{rel(\mu)}(\lozenge T) \cdot (\mathbb{E}(\blacklozenge\neg T\, U li|\lozenge T) + \mathbb{E}(\blacklozenge T|\lozenge T)_{li}^{rel(\mu)}) + \tag{4.13}$$

$$\cdots + P_{li}^{rel(\mu)}(\neg l_i\lozenge T) \cdot \mathbb{E}(\blacklozenge\neg l_i\, UT|\lozenge T)_{li}^{rel(\mu)})) = \ldots \tag{4.14}$$

Ekkor felbontottuk a várható aggregált mértéket két részre:

- Visszajutunk az l_i-be T-t nem érintve
- Eljutunk T-be anélkül, hogy l_i-t érintenénk

Így a 4.11 miatt $P_{li}^{rel(\mu)}(\lozenge T) = 1$ elhagyható és az egyenletet átrendezve:

$$\cdots = F_{reward}(l_i) + \sum_{l\in L} f_{(li;l)} \cdot (P_{l}^{rel(\mu)}(\neg\lozenge T\, U li) \cdot \mathbb{E}(\blacklozenge\neg T\, U li|\lozenge T) + \ldots \tag{4.15}$$

$$\cdots + P_{li}^{rel(\mu)}(\neg\{l_i\}\lozenge T) \cdot \mathbb{E}(\blacklozenge|\lozenge T\neg\{l_i\}\, UT)_{li}^{rel(\mu)})) + \ldots \tag{4.16}$$

$$\cdots + \sum_{l\in L} f_{(li;l)} \cdot P_{l}^{rel(\mu)}(\neg\lozenge T\, U li) \cdot P_{li}^{rel(\mu)}(\lozenge T) \cdot \mathbb{E}(\blacklozenge T|\lozenge T)_{li}^{rel(\mu)} \tag{4.17}$$

Így a korábbi logikát követve 4.8 mintájára felismerhető egy f és egy g függvény, melyek szintén lineárisan függenek x^{li}-től, hiszen ebben a hatalmas egyenletben egyedül $f_{(li;l)}$ függ tőle.

$$f(x^{li}) = F_{reward}(l_i) + \sum_{l\in L} f_{(li;l)} \cdot (P_{l}^{rel(\mu)}(\neg\lozenge T\, U li) \cdot \mathbb{E}(\blacklozenge\neg T\, U li|\lozenge T) + \ldots \tag{4.18}$$

$$\cdots + P_{li}^{rel(\mu)}(\neg\{l_i\}\lozenge T) \cdot \mathbb{E}(\blacklozenge\neg\{l_i\}\, UT|\lozenge T)_{li}^{rel(\mu)})) \tag{4.19}$$

$$g(x^{li}) = \sum_{l\in L} f_{(li;l)} \cdot P_{l}^{rel(\mu)}(\neg\lozenge T\, U li) \cdot P_{li}^{rel(\mu)}(\lozenge T) \tag{4.20}$$

Így $\mathbb{E}(\blacklozenge T|\lozenge T)_{li}^{rel(\mu)}$-t kifejezve f és g segítségével újra ki tudtam fejezni két elsőfokú polinom hányadosaként. ▪

Paraméterfüggő aggregált mérték Habár a legtöbb esetben az aggregált mérték egy állandó konstans a különböző lokációkban, mégis fontos megvizsgálni a helyzetet amikor a Markov-lánc paramétereitől függ. Ekkor a rátamátrix elemeihez hasonlóan az aggregált mérték is egy $\mathcal{Q}_V$-beli polinomot rendel a lokációhalamaz elemihez.

$$F_{reward}(l_i) : L \to \mathcal{Q}_V \tag{4.21}$$

Viszont szerencsénkre mivel a mérték egyedül 4.18 képletben és ott is csak első hatványon szerepel így f ugyanúgy lineáris marad, tehát a paramétermonotonitási tétel paraméterfüggő aggregált mérték esetén is igaz marad.
Ekkor azonban az algoritmus nagy mértékben módosul, ugyanis ekkor szubsztitúciót ki kell terjeszteni F_{reward}-ra is. Vagyis a különböző döntések határozzák meg az adott lokációbeli mérték nagyságát.

4.3.2. Várható idő bővítés

Algoritmusomban ezen lépés során történik meg a magasabb szintű problémák visszavezetése a Quatmann-féle algoritmus lépéseire. Viszont azt fontos még az elején leszögezni, hogy megoldásom bár a diszkrét Markov-láncokra vezeti vissza a problémát, de ez nem jelenti a hagyományos értelemben vett diszkretizálást vagyis azt, hogy egy folytonos időintervallumot diszkrét egyenlő hosszúságú időszakaszokra bontunk [15]. Ezekben a lépésekben már az időt egy absztrakt módon diszkrét időre leképezve fogjuk kezelni, vagyis a tranzíciókhoz tartozó időt a tranzíciók tüzelésének valószínűségével, a lokációhalmaz bővítésével és aggregált mértékekkel számoljuk habár diszkrét modelleket használnak, de mégis pontos eredményt adnak.

A reward bővítés során, arra a kérdésre keressük a választ, hogy egy adott célállapotok halmazába való megérkezés várható értéke nagyobb lesz-e mint egy adott határ. Ezt a módszert akkor célszerű alkalmazni ha a modellezni kívánt folyamat sokszor megy végbe, és ezekre rendszerként szeretnénk tekinteni és egy átlagos jósági tényezővel akarjuk jellemezni.

Példa Így ha a MoDeS3 terepasztalban tervezünk útvonalat akkor fontos megvizsgálni, hogy milyen paraméterek függvényében lesz az út során eltelt idő várható értéke a kívánt szint alatt. Ez akkor igazán érdekes ha nagyon sokszor küldünk vonatokat azon a bizonyos útvonalon és szeretnénk meghatározni, az átlagos menetidejüket, mely meghatározza mind a vonatok mind a sínszakaszok kihasználtságát. Viszont ezen algoritmus használata során mindig valamilyen sokszor előforduló jelenség vagy folyamat átlagos időbeli viselkedését határozzuk meg. Ilyen probléma még, hogy a terepasztalban lévő alkatrészeknek a hőmérséklet, áthaladó vonatok száma és egyéb paraméterek függvényében mekkora a várható élettartama és átlagosan milyen gyakorisággal kell őket.

Ezen algoritmus nagy mértékben épül a 25. oldalon bemutatott várható aggregált mérték algoritmusra, ugyanis visszavezetem a az egyes lokációban töltött időt egy aggregált mértékre. Így egy diszkrét idejű Markov-lánc és a hozzá kidolgozott segítségével tudom folytonos idejű modellemet számolni. [6]

A diszkrét idejű problémára történő visszavezetés lényege, hogy a 2.5-ban lévő levezetés miatt az az idő melyet egy adott lokációban töltünk exponenciális eloszlást mutat, melynek λ változója a kimenő tranzíciók λ változóinak összege, más néven az exit ráta lesz. Így ennek a reciproka lesz az adott lokációban töltött idő várható értéke, melyre mint lokációhoz rendelt aggregált mértékre tekintek.

$$F_{reward}(l_i) = \frac{1}{r(l_i)} \tag{4.22}$$

És ebből adódóan az adott célig várható aggregált mérték az út során eltelt idő várható értéke lesz.

$$\mathbb{E}(\blacklozenge T) = \mathbb{E}_t(\Diamond T) \tag{4.23}$$

Ekkor kihasználom, hogy két valószínűségi változó összegének várható értéke két várható érték összege lesz [Varga, Matematikai statisztika], illetve kihasználom még, hogy annak a valószínűsége, hogy egy adott lokációban egy adott kimenő tranzíció fog tüzelni az a

[6] Számolni és nem becsülni hiszen a kapott eredmény pontos.

tranzíció λ változójának és a lokáció exit rátájának a hányadosa lesz.

$$f_{(li;l)} = \frac{R_\lambda(l_i, l)}{r(l_i)} \tag{4.24}$$

Így a aggregált mértékre vett paramétermonotonitási tétel lényegében egy az egyben érvényes várható időre.

Viszont az exit ráta és ezáltal a lokációkhoz rendelt aggregált mérték lokációnként különbözni fog és a 4.23 egyenlőség miatt még nemlineárisan fog függeni a paraméterektől. Ennek kiküszöbülése végett még az algoritmus legelején elvégzek egy uniformizációt (3). Ezáltal minden exit ráta azonos lesz ezáltal lehetővé válik paraméterfüggtelen aggregált mértékkel számolni, mely nagy sebességbeli gyorsulást jelent.

4.3.3. Időkorlát alapú gráfbővítés

Míg sok hasonló esemény esetén a megérkezésig eltelt várható idő ad egy jó mérőszámot. Viszont ha az eseményeket és folyamatokat egyesével szeretnénk megvizsgálni akkor az időzített elérés tulajdonságát érdemes használnunk. Ekkor azt vizsgáljuk meg, hogy adott T célhalmazt el tudunk-e érni egy megfelelő valószínűséggel egy adott időtartamon belül.

Példa MoDeS3 terepasztalon rendkívül fontos, hogy a menetrendeket nagy pontossággal be tudjuk tartani, hisz a vasúti közlekedésben tapasztalhatjuk, hogy akár egyetlen késés is felboríthatja a teljes menetrendet. Így nélkülözhetetlen, hogy meg tudjuk állapítani, hogy egy adott út menetrend szerinti megtétele teljesíthető-e a kívánt mértékben (pédául 99,9% valószínűséggel).

Algoritmus működése Algoritmusom működése az uniformizációs tételen alapul [Optimal Time-Abstract Shedulers], mely kimondja, hogy folytonos idejű Markov-lánc esetén egy adott lokációban töltött idő független attól, hogy melyik kimenő tranzíció fog tüzelni. Ezáltal lehetővé válik a különböző utakat és a azok megtétele során eltelt időt. Ez azt jelenti, hogy minden egyes utat az út alatt tüzelt tranzíciók száma jellemez, vagyis hányszor kell lokációban várakozni. Ezáltal egy út során megtett idő valószínűségi változója az út során érintett lokációkban töltött idő valószínűségi változóinak összege. [7] Így uniformizált Markov-lánc esetén az út során eltelt idő valószínűségi változója azonos λ változójú exponenciális eloszlású valószínűségi változók összege, ezáltal n-ed rendű gamma eloszlást mutat [1] [3], ahol $n \in \mathbb{N}$ a megtett út során érintett lokciószám.

$$Distribution_n(t) = \gamma_n(t) \tag{4.25}$$

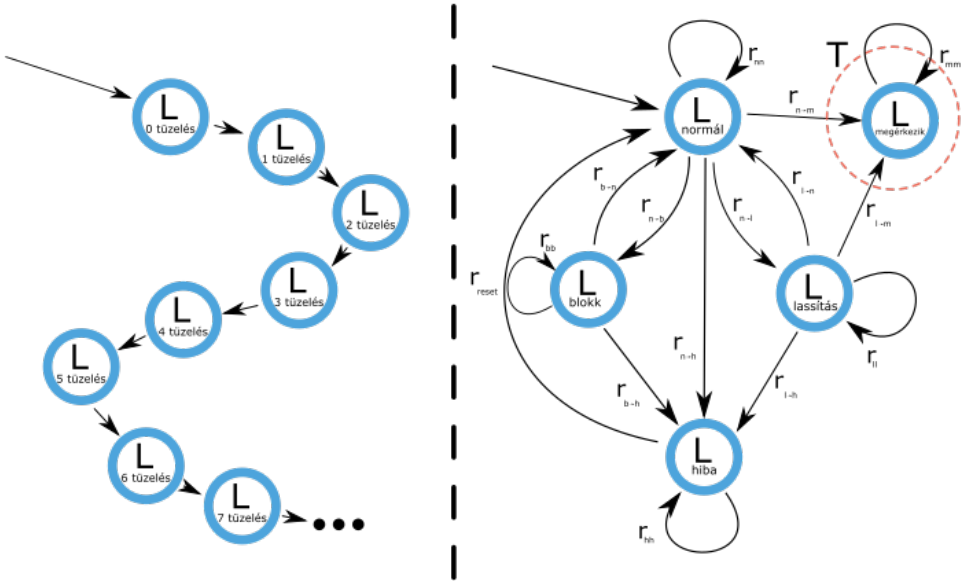
Ezáltal annak a valószínűsége, hogy egy adott $t_{limit} \in \mathbb{R}^+$ időn belül.

$$P^{\mu}_{tlimit} = \sum_{n=0}^{\inf} P(\Diamond L^n T) * \gamma_n(t_{limit}) \tag{4.26}$$

Ennek segítségével az időkorlátos elérés problémáját vissza tudjuk vezetni egy diszkrét idejű elérés problémájára, ha számon tudjuk tartani az út hosszát. Ehhez a 2.7-os ábrán látható módon párhuzamos állapotteret hozunk létre az alapján, hogy eddig hány tranzíció tüzelt. Így a kapott Markov-lánc lokációhalmaza 4.10-ben látható két halmaz Descartes

[7] Ha egy lokációt többször érintünk egy út során akkor az ott töltött idő valószínűségi változóját annyiszor kell hozzáadni ahányszor érintjük.

szorzata lesz. Ekkor kezdetben belekerülünk a ”Normál & 0 tranzíció tüzelt” állapotba, és ha ”Normál” → ”Lassítás” lokációátmenet bekövetkezik akkor ”Lassítás & 1 tranzíció tüzelt” lokációba fogok kerülni.



4.10. ábra

Az (4.26)-ben látható összegzést az általam használt megoldások [24] támogatják és a ezek segítséget nyújtanak a útvonal hosszának a számon tartására is.

4.3.4. Relaxáció és szubsztitúció

Mivel az általam kidolgozott becslő módszerek közös eleme, hogy a folytonos idejű sztochasztikus modelljeinket idő absztrakt módon [1] diszkrét idejű Markov-láncként vizsgáljuk, így ezen ezen konverzió után, ezek a lépések Qatmann algoritmusához képest változatlanok.

4.3.5. Nemlinearitások feloldása

A korábbi algoritmusok egyik legnagyobb hiányossága a nemlinearitások kezelése, hiszen a valóságban rendkívül kevés összefüggés modellezhető pontosan lineáris összefüggésekkel. Habár persze magától adódik a szakaszonkénti linearizálás, vagy más becslő módszerek alkalmazása, de ezekkel pont azt a pontosságot veszítenénk el, melyek pont a paraméteres Markov-lánc megoldóalgoritmusok erőssége. Erre jelent megoldást az algoritmusom, melynek segítségével képesek vagyunk bármilyen nemlineáris paraméterfüggőséget hatékonyan kezelni.

4.3.6. Helyettesítő paraméter módszere

Ha a modellünkben egy tranzíciót valamilyen bonyolult nemlineáris összefüggés határoz meg, akkor azt úgy kezeljük mint egy új paramétert, melynek értékeinek tartományát szélsőérték kereséssel határozhatjuk meg. Így lényegében egy extra relaxációs lépéssel minden nemlineáris összefüggést új változóval helyettesítünk, melyet úgy kezelünk mintha (értéktartományát leszámítva) teljesen független lenne a meghatározó paraméterektől.

Ezen módszer előnye, hogy különféle nemlinearitásokat, egyszerűen egy egységes rendszerben tudjuk kezelni. Viszont minden új nemlineáris függvény a tranzícióknál egy új helyettesítőparaméter behozatalát jelenti. Így sokféle nemlinearitás esetén a paraméterek száma nagy mértékben megemelkedik, mely a szubsztitúciónál, a Markov döntési folyamat generálása során a döntések számának drasztikus emelkedéséhez vezethet. De szerencsére ez a hatás csak abban az esetben jelentős, ha a lokációkhoz nagy számú kimenő tranzíció tartozik.

4.3.7. Paraméterfüggés megállapítása mérési adatok alapján

Eddig a dolgozatom során folyamatosan úgy tekintettem, hogy a modellemben lévő paraméteres összefüggések adottak. Pedig a mérnöki problémáknál az egzakt képletek és összefüggések meglehetősen ritkák, ezeket szinte mindig valamilyen mérési adatokból számoljuk. Így ha nem akarjuk az algoritmusunk pontosságát a mérési bizonytalanságok miatt elveszteni akkor szűkségünk van egy olyan számolási módszerre, mely garantálja az eredmény hitelességét még bizonytalan információk esetén is.

Bizonytalan paraméterek A modellezéshez használt Markov-láncainknak számos merőben különböző paramétere lehet, melyeket vagy szenzorok segítségével mérünk, vagy tervezési időben választhatunk meg. Viszont mindre igaz, hogy ezen értékek sohasem pontosak, mindig valamilyen tűréssel, bizonytalansággal határozhatóak meg.

Példa A különböző szenzorok, például hőmérő által gyűjtött adatokat minden esetben szórásukkal vagy standard bizonytalanságukkal jellemzik, melyet rendszerint a gyári adatlapon tüntetnek fel. Ennek eredményeként előállhat olyan szituáció, hogy például ha rendszerünk paraméteres Markov-lánc alapú modellünk elemzése során azt az eredményt kaptuk, hogy 39°C fölött már nem garantálható a biztonságos működés, és emiatt úgy programoztuk fel a vezérlőrendszert, hogy ezen határt túllépve állítsa le a működést, de szenzorunk 10%-os relatív hibával rendelkezik [8] , akkor előfordulhat olyan helyzet, hogy habár még csak 37°C-ot mérünk, de már jóval átléptük a valóságban ezen határt. Így rendszerünket a megszabott limitet meghaladó kockázatnak tettük ki. Természetesen még egy paraméter esetén ezen hiba általában csak kis mértékű, de sok paraméter esetén hibáik "worst case" összegződése [12] esetén komoly problémákat okozhatnak.

Paraméter bizonytalanság kiküszöbölése Ezen módszer lényege, hogy a szubsztitúció során nem a különböző akcióknál, a tranzíciókat meghatározó polinomokba nem az adott paraméter $I_p = [lower_{limit}; upper_{limit}]$ intervallumát használjuk, hogy annak határértékeit helyettesítjük be, hanem ezen intervallumot minden egyes szubsztitúciós lépés előtt még kibővítjük a lehetséges hibák értékével, ahol az intervallum alsó határát csökkentjük, míg a felső határát növeljük ezen értékkel.

Ennek a módszernek a használata során a Markov-lánc paraméterei mindig az észlelt megfigyelt értékek lesznek, például műszer által mutatott érték, vagy felhasznált alkatrészek paraméterinek névleges értékei, míg a szubsztitúciónál helyettesített értékek lesznek.

$$I'_p = [limit_{lower} - h(limit_{lower}); limit_{upper} + h(limit_{lower})] \tag{4.27}$$

Ekkor $h(x)$ az $x \in V$ paramétert meghatározó megfigyelt érték bizonytalanság függvénye, mely a legtöbb esetben x-től függ lineárisan (lásd. relatív hiba [Méréstechnika]). Így ebben az esetben akár új $h \in V$ paraméter felvételével is megoldható ezen probléma.

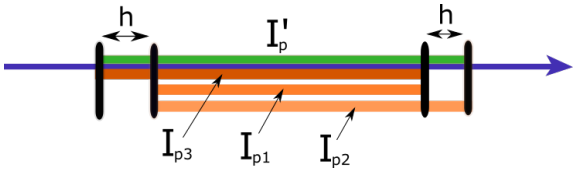
[8] Ilyen például a DHT22-es szenzor [dht]

$$x \to x + x \cdot h \mid x; h \in V \tag{4.28}$$

Habár ezzel a módszerrel egyszerűen kézben tarthatjuk a bizonytalanságot, ám a paraméterek száma növekedni fog és emiatt több bizonytalan paraméter esetén a számítási idő is nagy mértékben megemelkedhet, illetve meg kell jegyezni, hogy ekkor rengeteg felesleges számítást végzünk el hisz ekkor nemcsak (4.27)-t vizsgáljuk meg hanem 3 másikat is.

$$I_p^1 = [limit_{lower} - h(limit_{lower}); limit_{upper} - h(limit_{upper})] \tag{4.29}$$
$$I_p^2 = [limit_{lower} + h(limit_{lower}); limit_{upper} + h(limit_{upper})] \tag{4.30}$$
$$I_p^3 = [limit_{lower} + h(limit_{lower}); limit_{upper} - h(limit_{upper})] \tag{4.31}$$

Ezen intervallumok a 4.11 ábrán is láthatóak.



4.11. ábra. Bizonytalan intervallumok

Összefüggések megállapítása mérések alapján Sokkal, nehezebb a helyzet viszont akkor, ha nincs exakt képlet a mért mennyiség és a tranzíciók rátái között, hanem ezt is mérési adatok alapján kell meghatároznunk. Ezeknek a meghatározásához viszont nagy mennyiségű mérési adatra van szükségünk, de megfelelő szenzorokkal ezen adatok, gyorsan összegyűjthetőek.
Kielemzésükhöz megalkotott módszerem nagyban hasonlít a 4.3.6 -ben bemutatott módszerhez. Vagyis minden egyes mérési adatok alapján meghatározott tranzícióhoz rendelünk egy új paramétert, melynek értékét a rá ható paraméterek csak közvetetten határozzák meg.
Először meghatározom a tarnzíció rátájának eloszlását ezen paramétertérben, majd pedig minden egyes ciklusban megállapítom, hogy a paramétertér éppen vizsgált szakaszában a ráta, mely lehetséges értékeket veheti fel, így ezen tranzíció rátájához rendelt új paraméter intervalluma ezen lehetséges értékek alsó és felső határa közti valós számok halmaza. Ekkor lényegében egy extra relaxációs lépést építek be, mely képes a mérési adatok feloldására. Így mérési adatok alapján kapott összefüggések esetén is pontos számítások végezhetőek.

5. fejezet

Értékelés

Ezen fejezben mutatom be, hogy a kutatásom során kidolgozott elméleteknek és azokat felhasználó algoritmusoknak matematikai jelentőségükön túl számos gyakorlati haszna van. A fejezet első szakaszában algoritmusaim futását vizsgálom az általam fejlesztett tesztprogramok segítségével. Ezután a második részében a gyakorlati alkalmazásokra térek rá és megmutatom a MoDeS3 projekthez kapcsolódó esettanulmány segítségével, hogy hogyan használhatóak fejlesztéseim az üzemi hibák megelőzésére és a működés optimalizálására.

5.1. Mérések

Ezen alfejezetben algoritmusom hatékonyságát fogom megvizsgálni a gyakorlati alkalmazás szempontjából legfontosabb típusproblémákon keresztül.

5.1.1. Metrológia és mérési eljárások

Ezen a szakaszban bemutatom a teszteléshez használt mérési módszereket. A szakasz során ismertetem a teszteléshez a felhasznált programokat valamint azok bemeneteit. Továbbá megfogalmazom algoritmusom felé támasztott követelményeket is.

Tesztkörnyezet A mérések elvégzése során fontos volt algoritmusom működésének hatékonyságát és skálázhatóságát a lehető legtöbb szempont alapján megvizsgálnom, különös tekintettel a lehetséges alkalmazások során leggyakrabban előforduló helyzetekre. Így két tipikus problémán keresztül fogom megvizsgálni algoritmusom skálázhatóságát a lokációk számától és a paraméterek mennyiségétől és a modell bonyolultságától (lokációnkénti átlagos kimenő tranzíciószám) függően.

Ekkor vizsgálom a lokációkhoz tartozó akciók maximális számát, a paramétertartomány 95%-os lefedéséhez szűkséges iterációk számát, illetve a paramétertartomány lefedettségének alakulását az iterációk számának függvényében.

A mérések elvégzéséhez az 4.3 "Új algoritmusok" részben bemutatott várható idő algoritmust 4.3.2 leimplementáltam java programozási nyelven. Melyeknek segítségével figyelem, hogy a folytonos idejű Markov-lánc paramétertartományán, mely értékek esetén lesz a várható érték biztosan nagyobb vagy kisebb mint egy adott határ. Ekkor nemcsak, a számításaim numerikus eredményeit figyelem, hanem a programomba beépítettem speciális monitorozó függvényeket, melyeknek segítségével lehetővé válik a program futása közben a korábban említett adatokat kigyűjteni és a megfelelő statisztikákat elkészíteni. Így ezen függvények segítségével lehetővé válik futás közben:

- Iterációs szint számontartása, vagyis hogy eddig minimum hányszor vágtuk félbe a paramétertartományt.
- Az iterációs lépésszám, vagyis hogy eddig hány becslő ciklus (hány darab szubsztitúció történt) futott le.
- A paramétertartomány lefedettségi szintje, vagyis hogy a paramétertartomány hány százalékára sikerült eddig sikeres becslést mondanunk, hogy teljesül-e az adott tulajdonság vagy sem.
- A szubsztitúciós lépések idejének mérése is lehetővé vált.
- Két paraméteres modellek esetén azokat a tartományokat, melyekre a becslés megállapítja, hogy biztosan nagyobbak vagy kisebbek egy határnál automatikusan megjeleníti egy ".svg" fájlban.

Ekkor a paramétertartomány lefedettsége során paraméterteret Euklideszi térnek tekintve, minden egyes becslés után kiszámolom a sikeresen becsült tartomány Jacobi mértékét és összehasonlítom az eredeti tartomány Jacobi mértékével.

Programom megírása során a szubsztitúciós lépés megvalósításához felhasználtam a Prizm Model Checker [24] programot, melynek számos funkciója közül a nemdeterminizmusok modellezését és Markov döntési folyamatok elemzésének lehetőségét (aggregált mérték és elérés szempontjából) használom ki. A programomban a PRIZM számára megfelelő bemenetek generálása után meghívom ezen programot, majd a kapott eredmények beolvasásával és értelmezésével lehetővé válik a Markov döntési folyamatok számítása a külső programon keresztül.

Algoritmusom számára a legfontosabb követelmény a gyors futás és a jó skálázódás. Ennek megvizsgálásához elengedhetetlen megvizsgálni, hogy a futási idővel, vagyis az iterációs lépések számával milyen ütemben tudjuk a vizsgált paramétertartományt leellenőrizni. De ezen kívül a bemenetek futási időre nézve gyakorolt hatásának vizsgálata is elengedhetetlen.

A tesztelés során több különböző folytonos idejű Markov-láncot. Ezek lánc felépítésűek vagyis a kezdeti lokációból egy változó egyszerű út topolódiájú gráfon keresztül kell eljutni a céllokációba. Ekkor algoritmusom skálázódásának vizsgálatához, változtatom a lánc hosszát, a paraméterek számát illetve a struktúra összetetségének elemzéséhez új tranzíciókat is felvettem.

Futási idő mérése A futási idő mérése egy rendkívül összetett feladat, lévén azt mind a szoftveres implementáció módja, mind az azt futtató számítógép tulajdonságai erősen befolyásolja. Ezenkívül a felhasznált Markov döntési folyamat megoldó hatékonysága is meghatározó tényező. Viszont a futás idejének vizsgálatához az iterációs szubsztitúciós becslő ciklusok száma egy nagyon jó mérték. Ugyanis a szubsztitúció során generált Markov döntési folyamat mérete a futás során nem változik. Így a megoldásához hozzávetőlegesen mindig ugyanannyi idő szűkséges. Így a futási idő becslésére a 5.1 képlet használható, ahol c_{step} a lépésszámot, míg $\Delta\bar{t}_{solver}$ a Markov döntési folyamat megoldásához szűkséges idő átlagát jelöli.

$$t_{run} = c_{step} \cdot \Delta\bar{t}_{solver} \tag{5.1}$$

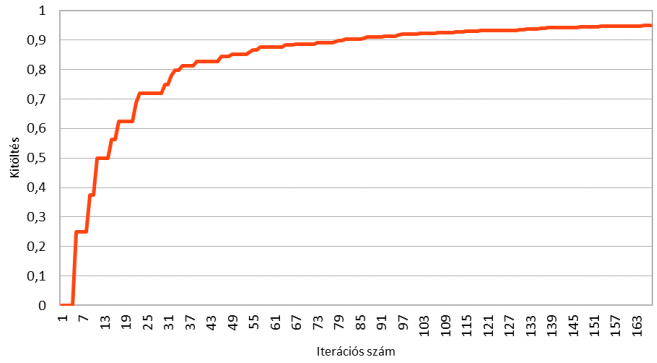
5.1.2. Mérési eredmények

Ezen szakszban mutatom be mérési eredményeimet illetve azok értékelését.

Lokációk számának hatása Méréseim rendkívül meglepő eredménnyel szolgáltak, ugyanis az iterációs lépésszámra a lokációk számának hatása elenyésző volt. A teszteléshez használt Markov-láncokban a lokációk számát kétszeresére emelve a lépésszámban nem történt számottevő változás. Viszont ez nem jelenti azt, hogy a lokációk száma nem befolyásolná a futási időt, ugyanis a Markov döntési folyamat megoldók számára lényeges kérdés a lokációk száma. Viszont mint azt tanulmány [10] is alátámasztja a lokációk számával a megoldáshoz igényelt idő közel lineárisan arányos.

A tranzíciók számának hatása Mérési eredményeim alapján a tranzíciók hatása a futási időre közel azonos a lokációk hatásával. Így ezen tényező lépésszámra gyakorolt hatása elenyésző és a Markov döntési folyamat megoldásához szűkséges időt növeli.

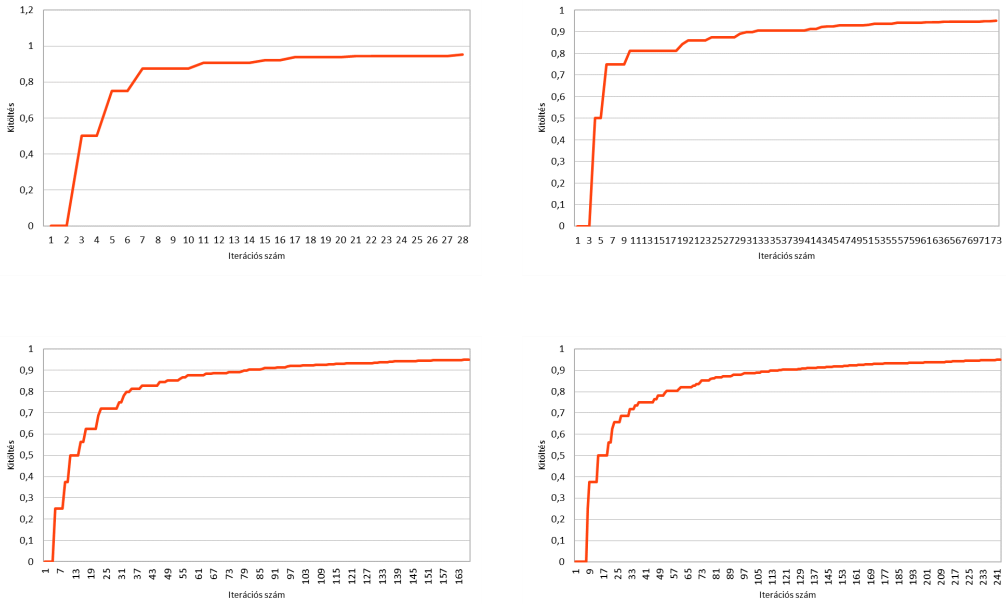
A tartomány lefedettsége Fontos megvizsgálni, hogy a program futása során, hogyan változik a paramétertartomány lefedettsége a iterációs számmal. Az általam vizsgált tesztbemenetek esetén ahogy haladt előre a program a le nem fedett paramétertartomány (azon területtek, melyekre az iterációs becslés során nem kaptunk eredményt) aránya exponenciálisan tart a 0-ba, mint az a 5.1 ábrán is látható. A mérések során minden esetben a 5.1



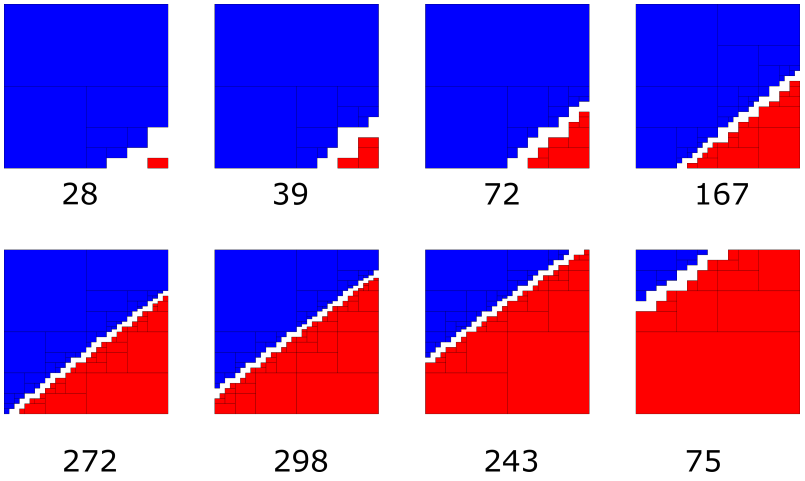
5.1. ábra. A paramétertartomány lefedettsége (kitöltése) az iterációs számmal tart az 1-be

ábrához hasonló eredményt kaptam. A különböző teszt bemenetek eredményeikben csak a konvergálás sebessége különbözött. Ezen exponenciális konvergencia a 5.2 ábrán is látszik ahol három különböző bemenet esetén hasonlítom össze a konvergálás gyorsaságát.

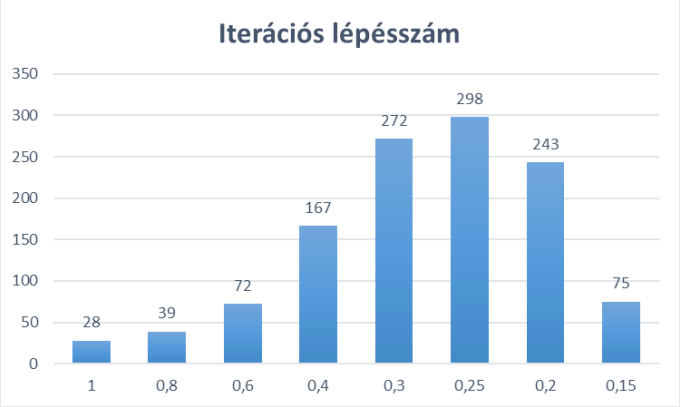
A paraméterek hatása A paraméterek hatása már egy sokkal érdekesebb kérdés, hiszen vele együtt a paramétertér dimenziója is megemelkedik, vagyis a taranzíciók rátái jóval rugalmasabban változhatnak. Ennek következményeként, szubsztitúciós lepésszám drasztikusan megemelkedik. Hiszen a paraméterteret kizárólag téglatestekkel fedhetjük le ezen algoritmussal, és a bonyolult, sok dimenziós tartományok lefedése csak sok lépésben valósítható meg a segítségükkel. Továbbá ezen tartományok lefedését az határozza meg, hogy a határvonal, mely elválasztja a azokat a tartományokat, ahol a várható érték biztosan kisebb lesz a határnál azoktól, melyekben a várható érték biztosan nagyobb lesz. Ezt a méréseim is alátámasztották és a paraméterek. Ezen jelenség a 5.3 ábrán is megfigyelhető, melyen a programom segítségével generált tartományok láthatóak. Az ábrában a képek alatt a 95%-os lefedettség eléréséhez szűkséges iterációs lépésszám látható.



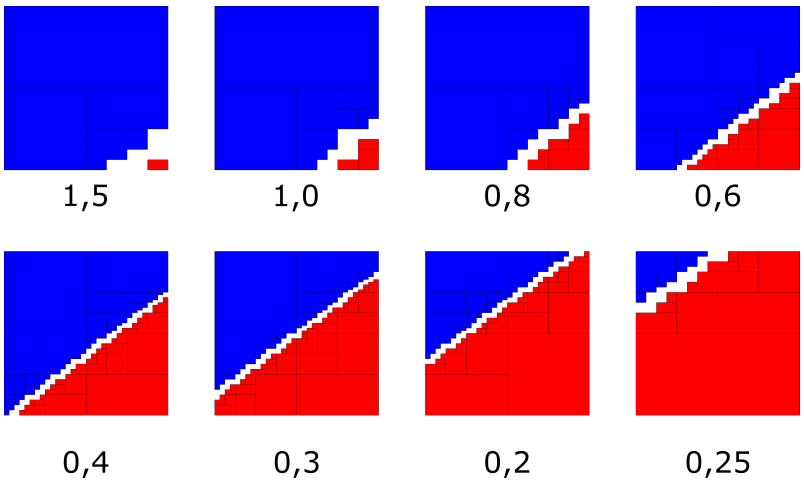
5.2. ábra



5.3. ábra. Az iterációs szám alakulása a tartományokat elválasztó határvonal változásai esetén



5.4. ábra. Határ változásainak hatása az iterációs számra



5.5. ábra. A tartományok közti határ eltolódása a határ függvényében.

A vizsgált határ hatása Méréseim során azt tapasztaltam, hogy egy adott folytonos idejű paraméteres Markov lánc esetén a futási idő (az iterációs szám) több nagyságrenddel is megnőhet attól függően, hogy a szubsztitúciós, becslő ciklusok során milyen határt állítok be. Ezen változtatás hatását egy három lokáció hosszú láncon vizsgáltam, melynek eredménye a 5.4 ábrán látható. Ekkor a határt 0.15-től egészen 1-ig változtattuk. Ezen jelenség magyarázata, hogy amikor a határt folyamatosan állítjuk, a határvonal (mely elválasztja a azokat a tartományokat, ahol a várható érték biztosan kisebb lesz a határnál azoktól, melyekben a várható érték biztosan nagyobb lesz) fokozatosan eltolódik, s ezáltal az előző 5.1.2 részben bemutatott okok miatt a lefedéshez szűkséges lépésszám is változik. Ezen jelenség látható a 5.5 ábrán is, ahol a kapott tartományok változása jelenik meg a határ függvényében. Ezen határ a tartományok alatt található.

5.2. MoDeS3 esettanulmány

Ebben a részben mutatom be algoritmusom gyakorlati hasznosítását a MoDeS3 projekt segítségével, mintegy mintapéldaként a későbbi hasznosításokhoz. Így megmutatom, hogy fejlesztéseim, hogyan használhatóak csapatommal közösen kiépített MoDeS3 projekthez tartozó IoT szenzorrendszer által gyűjtött adataim magas szintű feldolgozására, mely során az adatokból olyan magas szintű információkat tudok kinyerni, mely garantálja a

terepasztal optimális működését.
Ezen szakaszban be fogom mutatni, hogy algoritmusom, hogyan alkalmazható a menetrendek betartásához, az eszközök amortizációjának minimalizálásához és az üzemzavarok elkerüléséhez.

5.2.1. Projekt ismertetése

A MoDeS3 projekt vagyis (Modell Based Demonstrator for Smart and Safe Systems) még 5 évvel ezelőtt kezdődött a MIT tanszéken, célja a tudomány népszerűsítése és egy könnyen kezelhető tesztkörnyezet biztosítása az ott dolgozó hallgatók számára. Ezen projektben fejlesztett eszköz egy okos terepasztalt, melyen mind a vonatok, mind a váltók központilag irányíthatóak egy elosztott vezérlőrendszeren keresztül.

5.2.2. Terepasztal felépítése

Mechanikai felépítés A terepasztal külső felépítése átlagosnak mondható. Összesen egy dupla körpályából, egy mellékvágányból és egy leálló holtvágányból áll. A szakaszok közti zavartalan átjárást öt hagyományos és egy angol váltó biztosítja, míg a digitális vonatokat a Roco vezérlő egységével lehet irányítani.

Elektronikai rendszer A terepasztal egyszerű külseje mögött egy elosztott informatikai rendszer van. A vasúti pálya hat részre van osztva, amikért egy-egy Beaglebone [beaglebone] alapú beágyazott vezérlőegység felel.

Informatikai keretrendszer A rendszer vezérléséért és funkció eléréséért egy Java és Xtend [13] nyelven írt framework felel. Így lehetőség nyílik a terepasztal működésének teljes vezérlésére.

- Sínszakaszok lekapcsolhatóak, így az adott sínszakaszra rámodulált jel segítségével ha egy vonat arra a sínszakaszra érkezik azonnal megáll.
- Váltók központilag kapcsolhatóak.
- Lehetőség nyílik az útvonaltervezésre [Khaled program], a projektben kifejlesztett útvonaltervező-programnak köszönhetően. Ennek segítségével lehetséges két sínszakasz között a legrövidebb útvonal meghatározása és a vonat végigirányítása rajta.
- Vonatok helyének meghatározása [vonathelymeghatarozo program] is lehetséges, a vonatokon elhelyetett markerek, a terepasztal fölött elhelyezett kamera és optikai képfelismerés segítségével.

5.2.3. IoT szenzor réteg

MoDeS3-projektem végzett munkám során csapatommal ezt a rendszert egy internet alapú IoT szenzorréteggel bővítettem, hogy a lokális beavatkozás és megfigyelés eszköztárát radikálisan kibővítsem, és lehetővé váljon komplex hibahelyzetek felismerése, ezek alapján a gyors lokális beavatkozás és a mérési adatok központosított adatbázisba történő mentése. Ezáltal szenzoraim segítségével képes vagyok mérni:

- vonatok gyorsulását és rezgését, a saját tervezésű vonatra szerelhető WiFi-s gyorsulásmérő segítségével
- vonatok elhaladását és a sínek rezgését, a saját tervezésű sín mellé szerelhető WiFi-s szenzor modulom segítségével

- időjárást, vagyis a hőmérsékletet, a páratartalmat, a légnyomást és a fényerőt az asztalra felhelyezhető WiFi-s Wemos szenzor modulok segítségével

Ezen kívül a csapatommal közös munkám során okos képfelismerő kamerákat helyeztünk el mind a vonatokra, mind a sínek mentén. Képeiket webes felületen tudjuk megjeleníteni és segítségükkel képesek vagyunk meghatározni ha ember vagy más veszélyt jelentő tárgy kerül a sínek közé és ekkor lekapcsolni ezen veszélyes sínszakaszt.
Az észlelt anomáliákat kigyűjtjük és központi adatbázisban tároljuk, illetve fejlesztés alatt áll mind az észlelt hibák, mind a szenzorok adatainak helyhez történő kötése.
Továbbá az elosztott tervezési elveket szem előtt tartva, az adatbázis és a szenzorok közé egy ú.n. lokális edge feldolgozót [29] helyeztünk, mely képes az adatok klaszterezésére, anomáliák észlelésére és lokális beavatkozásra.

5.2.4. Modellezés Markov-láncokkal

A modellezés során nagy nehézséget jelentett, hogy sok különböző egymástól nagy mértékben eltérő mennyiséget kell egyszerre figyelembe venni. Erre az egyik legjobb megoldás a valószínűség alapú modellezés, mely során egyszerre összegezni tudjuk ezen jelenségek hatásait. Így például mind a hőmérséklet, mind az emberi mulasztás okozta hibák valószínűségeit modellezhetjük. Továbbá lehetőség nyílik ezek kombinált hatásainak elemzésére is, például nagy melegben az emberek nagyobb valószínűséggel követnek el hibákat.

Ezek alapján alkottam meg a terepasztal Markov-lánc alapú modelljét is. Ekkor a vezérlőprogram, állapotgépét képeztem le egy Markov-láncra. Az 5.6 ábrán [1] látható ezen vezérlő állapotai illetve milyen külső események hatására következik be. A program az alábbi módon működik, alapból a "Normál" állapotban van, mely maximális megengedett sebességgel történő haladást jelent a cél felé, ha a vonaton elhelyezett képfelismerő kamera embert vagy állatot észlel a sínek között akkor azonnal leállítja a vonatot és egészen addig vár ameddig el nem megy onnan, ha a vonatra felszerelt gyorsulásérzékelő jelzi, hogy a vonat rázkódása túlságosan megnő akkor lelassítja a vonatot és csak akkor engedi újra gyorsan haladni, ha a rázkódás megfelelően visszacsökkent a normális szintre illetve súlyos hiba esetén (kisiklás, ütközés stb...) létezik egy vészleállítás állapot is, melyből csak manuálisan lehet a rendszert visszaállítani.

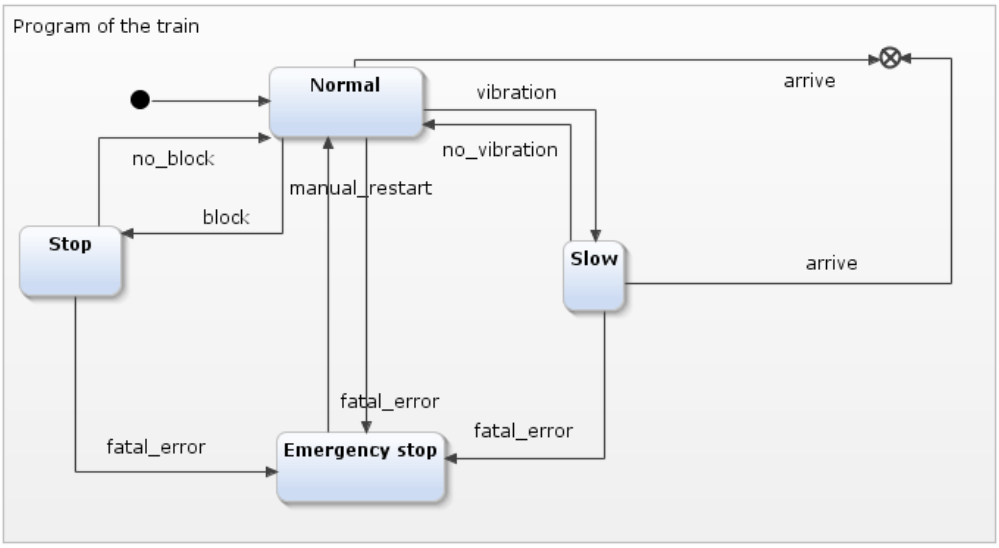
Ezen állapotgépből úgy kapunk DTMC-t ha szenzorok segítségével gyűjtött nagy mennyiségű naplóból meghatározzuk, hogy egy átlagos út során milyen relatív gyakorisággal következnek be az állapot átmenetek. A pontos eljárást a 4.3.7 szakaszban található.

Természetesen egy jóval precízebb modellt kapunk, ha a modellvasútpályát különböző sínszakaszokra bontjuk és a vonat által éppen megtenni kívánt út minden egyes sínszakaszában értelmezzük az vezérlő program állapotait, mint azt az alábbi 5.7 ábrán is láthatjuk.

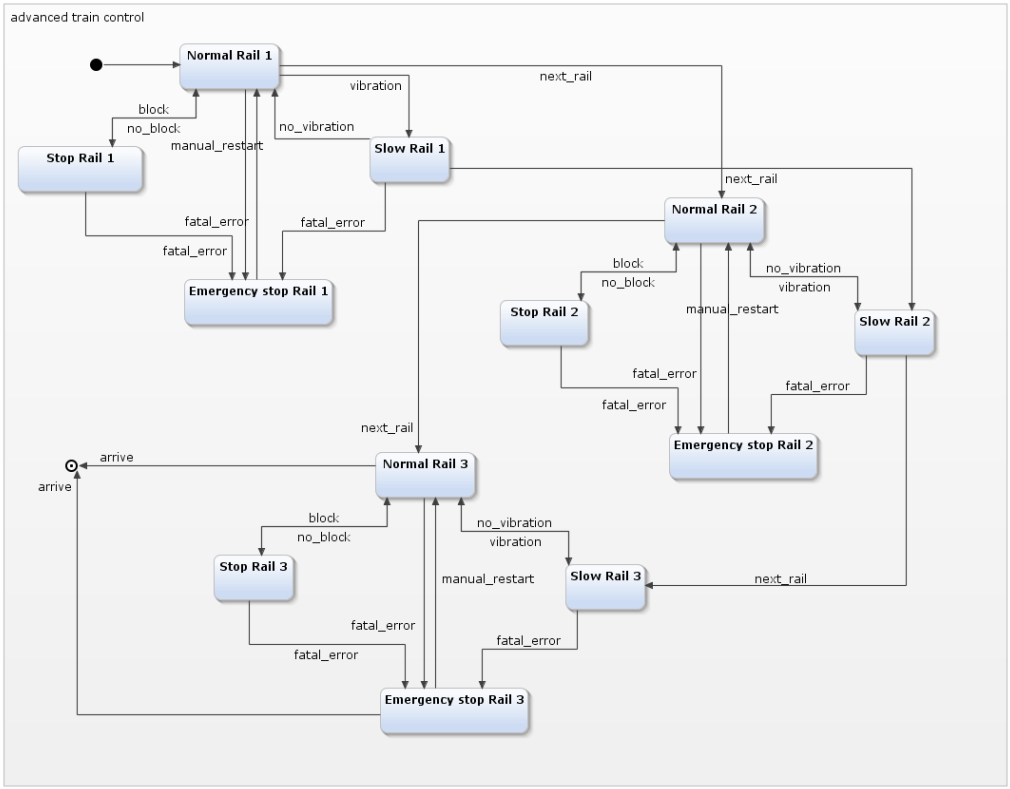
5.2.5. Útvonaltervezés a mechanikai igénybevétel optimalizálásával

Az IoT szenzor rendszerek kiépítésének elsődleges célja a karbantartások optimális ütemezése és az üzemi meghibásodások számának minimalizálása. Ennek eléréséhez elengedhetetlen, hogy az elöregedő alkatrészeket, még azelőtt lecseréljük mielőtt meghibásodást okoznak. Illetve a kiadások csökkenését eredményezi, ha még működés közben a fizikai terhelést optimalizálni tudjuk.

[1] Az alábbi ábra a Yakindu [32] grafikus modellezőprogrammal készült.



5.6. ábra. Vonat állapotgépe



5.7. ábra. Vonat állapotgépe a sínszakaszokon

Ennek eléréséhez nyújt segítséget a 4.3.1-ban bemutatott algoritmusom, melynek segítségével a lehetséges útvonalak közül kiválaszthatóak azok, melyeknél a terhelés az adott szint alatt tarthatóak, illetve ha ez nem lehetséges egyszerűen megtalálhatóak a rendszer gyenge pontjai, mely a megkívánt követelmény teljesülésében akadályoznak minket.

Ehhez elengedhetetlenül szűkséges, hogy a 5.2.4 részben megalkotott Markov láncunkat, hogyan lehet kibővíteni aggregált mértékkel, mely az amortizációt jellemzi. Ennek

megalkotásához a vonatokon elhelyezett gyorsulásmérő adatait használom fel, melynek segítségével képes vagyok vonatot érő rázkódás mérésére. Ugyanis az négyzetes középértéke a vonat súlyának függvényében megadja a rendszert érő terhelés teljesítményét [12].

A méréseim folyamán egy 2m hosszú sínszakaszon ment végig a vonat különböző feltételek mellett. Először nem alkalmaztam semmilyen extra körülményt, illetve gondosan ügyeltem a síndarabok pontos illeszkedésére és tisztaságukra. Utána a vonat egyik kerekét szándékosan kisiklattam, majd végül kicseréltem az egyik síndarabot egy speciális sínszakaszra melyet előre lereszeltem[2]. Minden mérést háromszor megismételtem három különböző sebesség mellett. Ezek alapján az elnyelt energiára a következő 5.2 képlet adódik.

$$E = a_{\text{közép}} \cdot m \cdot \mathbb{E}(t) \tag{5.2}$$

Ekkor viszont szűkség van a 4 fejezetben bemutatott 4.3.1 szakaszban található aggregált mérték számolására képes és 4.3.2 szakaszban bemutatott várható idő számítására képes algoritmus kombinálására. Ez egyszerűen megvalósítható hisz mindkét módszer aggregált mérték alapú.

Így az energia alapú aggregált mérték segítségével lehetővé válik egy út megtervezése során megállapítani, hogy a vonat által elnyelt rázkódás energiájának várható értéke a kívánt szint alatt van-e.

Méréseimből látható, hogy a rezgést egyik legfontosabban meghatározó paraméter a sebesség, de mind a vonat mind a sín állapota jelentősen befolyásolja azt. Továbbá a vonatot terhelő erőket még $\vec{F} = m \cdot \vec{a}$ a vonat súlya is lineárisan befolyásolja Newton második törvénye alapján [26]. Viszont mivel itt munkavégzést vizsgáljuk így a mechanikai szerkezet által elnyelt energia a $W = p \cdot \Delta t$ képlet alapján arányos az erőhatás idejével. Így a várható idő is egy paramétere a terhelésnek.

Végül még ki kell térnünk arra, hogy Markov-láncunkban a célhalmaztól eltérő nyelők vagy nyelő struktúrák vannak mint például vészleállások. Ekkor az aggregált mérték tartana a végtelenbe, mely komoly számításbeli problémát okoz. Ekkor érdemes közte és a célhalmaz eleme között egy nagy várható értékű tranzíciót felvenni, mely garantálja a megérkezést, de ezen az úton a várható érték biztosan nagyobb lesz mint a határérték. Ennek a szemléletes jelentése, hogy miután megérkezik az operátor akkor miután a problémát megoldja bekerül a célhelyére.

5.2.6. Sebesség-optimalizált útvonaltervezés

A menetrend betartása során már jóval egyszerűbb dolgunk van. Ugyanis a Markov-lánccal történő modellezés után már a korlátos idő alapú elérési algoritmus egy az egyben elvégzi a feladatot. Csupán arra van szűkség, hogy a begyűjtött naplók alapján meghatározni a 4.3.7 szakaszban bemutatott módszer segítségével, hogy a megfigyelhető paraméterek függvényében mekkora a sínszakaszokban töltött idő elosztásának rátája. Így ezekkel lehetővé válik, hogy a kíválasztott útvonalon meghatározzuk hogy a kívánt időtartam segítségével történő elérés lehetséges-e a kívánt valószínűséggel.

[2] Ekkor a vont kereke nem volt kisikolva.

6. fejezet

Összefoglalás és továbbfejlesztési lehetőségek

Összegzés A dolgozatomban bemutattam a kutatásom során kifejlesztett új algoritmusokat, melyeknek segítségével lehetővé válik a Markovi modelleknek egy sokkal tágabb halmazát a folytonos idejű paraméteres Markov-láncokat vizsgálni, és olyan tulajdonságokat megvizsgálni mint a várható megérkezési idő (4.3.2) vagy a időkorláton belüli megérkezés valószínűsége (4.3.3). Így lehetővé válik megállapítani, hogy a vizsgált tulajdonság a paramétertartomány, mely részintervallumain lesz kisebb mint egy adott határérték. Ezáltal lehetővé válik a paraméteres-modellellenőrzés felhasználhatóságának a nagy mértékű bővítése, mint azt egy esettanulmányban, a 5.2 szakaszban be is mutatom a MoDeS3 projekt kapcsán. Így Ezen kívül megvizsgáltam programom futását különböző Markov láncokon. Megállapítottam, hogy a tesztelt algoritmusom jól skálázódik a bemenet növekedéséve, valamint a még megvizsgálatlan része a paramétertartománynak exponenciálisan csökken a futási idővel. Így megállapítható, hogy az általam kifejlesztett algoritmusok használatának a nincs technológiai akadálya.

Továbbfejlesztési lehetőségek Kutatásom természetesen nem ért véget, számos tervem van a munkám folytatásáról. Elsődlegesen szeretnék a vizsgált tulajdonságok körét bővíteni, valamint a meglévő algoritmusokat kiterjeszteni folytonos idejű paraméteres Markov döntési folyamatokra valamint sztohasztikus játékokra [27]. Emellett szeretném a 5.1.1 részben bemutatott, tesztelésre használt tesztkörnyezetet és algoritmusaim implementációit egy önálló segédszoftverré fejleszteni, mely segíthet megbízhatósági problémáik elemzésében.

Viszont hosszabb távú tervem egy Markovi modelleken alapuló biztonsági vezérlőrendszer megalkotása. Ugyanis eddigi rendszerem legnagyobb hiányossága, hogy a rendszert modellező Markov-lánc topológiáját egyedül a vezérlő szoftver struktúráját jellemzi. Célom hogy egy olyan elosztott megfigyelő rendszer megalkotása, amelyben rendszerben elhelyezett elosztott vezérlő központok képesek felismerni a mechanikai rendszerben lévő rejtett Markov-láncokat és paraméterfüggéseit. Ezeket elküldve pedig a központi vezérlő egység a sok kis részlet alapján összeállítja a rendszer teljes modelljét, melynek paraméteres ellenőrzése során lehetővé válik a rendszerünk vezérlésének és környezetének együttes modellezésére és a megfelelő követelményeket teljesítő vezérlő stratégiák megalkotása.

Irodalomjegyzék

[1] Christel Baier – Holger Hermanns – Joost-Pieter Katoen – Boudewijn R Haverkort: Efficient computation of time-bounded reachability probabilities in uniform continuous-time markov decision processes. *Theoretical Computer Science*, 345. évf. (2005) 1. sz., 2–26. p.

[2] Nicolas Basset – Marta Z. Kwiatkowska – Ufuk Topcu – Clemens Wiltsche: Strategy synthesis for stochastic games with multiple long-run objectives. In *Tools and Algorithms for the Construction and Analysis of Systems - 21st International Conference, TACAS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015. Proceedings* (konferenciaanyag). 2015, 256–271. p.
URL `https://doi.org/10.1007/978-3-662-46681-0_22`.

[3] Peter Buchholz – Ernst Moritz Hahn – Holger Hermanns – Lijun Zhang: Model checking algorithms for ctmdps. In *Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings* (konferenciaanyag). 2011, 225–242. p.
URL `https://doi.org/10.1007/978-3-642-22110-1_19`.

[4] Peter Buchholz – Ernst Moritz Hahn – Holger Hermanns – Lijun Zhang: Model checking algorithms for ctmdps. In *International Conference on Computer Aided Verification* (konferenciaanyag). 2011, Springer, 225–242. p.

[5] Carlos E. Budde – Christian Dehnert – Ernst Moritz Hahn – Arnd Hartmanns – Sebastian Junges – Andrea Turrini: JANI: quantitative model and tool interaction. In *Tools and Algorithms for the Construction and Analysis of Systems - 23rd International Conference, TACAS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings, Part II* (konferenciaanyag). 2017, 151–168. p. URL `https://doi.org/10.1007/978-3-662-54580-5_9`.

[6] Georgel Calin – Pepijn Crouzen – Pedro R. D'Argenio – Ernst Moritz Hahn – Lijun Zhang: Time-bounded reachability in distributed input/output interactive probabilistic chains. In *Model Checking Software - 17th International SPIN Workshop, Enschede, The Netherlands, September 27-29, 2010. Proceedings* (konferenciaanyag). 2010, 193–211. p. URL `https://doi.org/10.1007/978-3-642-16164-3_15`.

[7] Dave Parker. *PRISM Modelling Language Manual*. 2010. 12. `http://www.prismmodelchecker.org/manual/ThePRISMLanguage/Introduction`.

[8] Leonardo Mendonça de Moura – Nikolaj Bjørner: Z3: an efficient SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008.*

Proceedings (konferenciaanyag). 2008, 337–340. p.
URL https://doi.org/10.1007/978-3-540-78800-3_24.

[9] Christian Dehnert – Sebastian Junges – Nils Jansen – Florian Corzilius – Matthias Volk – Harold Bruintjes – Joost-Pieter Katoen – Erika Ábrahám: Prophesy: A probabilistic parameter synthesis tool. In *Computer Aided Verification - 27th International Conference, CAV 2015, San Francisco, CA, USA, July 18-24, 2015, Proceedings, Part I* (konferenciaanyag). 2015, 214–231. p.
URL https://doi.org/10.1007/978-3-319-21690-4_13.

[10] Christian Dehnert – Sebastian Junges – Joost-Pieter Katoen – Matthias Volk: A storm is coming: A modern probabilistic model checker. In *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part II* (konferenciaanyag). 2017, 592–600. p.
URL https://doi.org/10.1007/978-3-319-63390-9_31.

[11] Susanna Donatelli – Giuliana Franceschinis: Modelling and analysis of distributed software using gspns. In *Lectures on Petri Nets II: Applications, Advances in Petri Nets, the volumes are based on the Advanced Course on Petri Nets, held in Dagstuhl, September 1996* (konferenciaanyag). 1996, 438–476. p.
URL https://doi.org/10.1007/3-540-65307-4_54.

[12] dr. Schnell László: *Jelek és rendszerek méréstechnikája*. 1998, Műszaki Könyvkiadó.

[13] Eclipce. *Xtend User Guide*. 2014. 9. https://www.eclipse.org/xtend/documentation/2.7.0/Xtend%20User%20Guide.pdf.

[14] Cheng Feng – Jane Hillston: Speed-up of stochastic simulation of PCTMC models by statistical model reduction. In *Computer Performance Engineering - 12th European Workshop, EPEW 2015, Madrid, Spain, August 31 - September 1, 2015, Proceedings* (konferenciaanyag). 2015, 291–305. p.
URL https://doi.org/10.1007/978-3-319-23267-6_19.

[15] Fodor György: *Hálózatok és rendszerek I-II.* 2014, BME Printer kft.

[16] Ernst Moritz Hahn – Holger Hermanns – Björn Wachter – Lijun Zhang: INFAMY: an infinite-state markov model checker. In *Computer Aided Verification, 21st International Conference, CAV 2009, Grenoble, France, June 26 - July 2, 2009. Proceedings* (konferenciaanyag). 2009, 641–647. p.
URL https://doi.org/10.1007/978-3-642-02658-4_49.

[17] Ernst Moritz Hahn – Holger Hermanns – Björn Wachter – Lijun Zhang: PARAM: A model checker for parametric markov models. In *Computer Aided Verification, 22nd International Conference, CAV 2010, Edinburgh, UK, July 15-19, 2010. Proceedings* (konferenciaanyag). 2010, 660–664. p.
URL https://doi.org/10.1007/978-3-642-14295-6_56.

[18] Holger Hermanns – Joost-Pieter Katoen – Joachim Meyer-Kayser – Markus Siegle: A markov chain model checker. In *Tools and Algorithms for Construction and Analysis of Systems, 6th International Conference, TACAS 2000, Held as Part of the European Joint Conferences on the Theory and Practice of Software, ETAPS 2000, Berlin, Germany, March 25 - April 2, 2000, Proceedings* (konferenciaanyag). 2000, 347–362. p. URL https://doi.org/10.1007/3-540-46419-0_24.

[19] David M Higdon: Auxiliary variable methods for markov chain monte carlo with applications. *Journal of the American Statistical Association*, 93. évf. (1998) 442. sz., 585–595. p.

[20] Iadine Chadès∗, Guillaume Chapron†, Marie-Josée Cros‡, Frédérick Garcia‡, Régis Sabbadin‡. *Quick Start: Resolving a Markov decision process problem using the MDP-toolbox in Matlab*. 2014. 1. `www7.inra.fr/mia/T/MDPtoolbox/QuickStart.pdf`.

[21] M. Usman Iftikhar – Danny Weyns: Activforms: A runtime environment for architecture-based adaptation with guarantees. In *2017 IEEE International Conference on Software Architecture Workshops, ICSA Workshops 2017, Gothenburg, Sweden, April 5-7, 2017* (konferenciaanyag). 2017, 278–281. p.
URL `https://doi.org/10.1109/ICSAW.2017.21`.

[22] J.-P. Katoen – M. Kwiatkowska – G. Norman – D. Parker: Faster and symbolic CTMC model checking. In L. de Alfaro – S. Gilmore (szerk.): *Proc. 1st Joint International Workshop on Process Algebra and Probabilistic Methods, Performance Modeling and Verification (PAPM/PROBMIV'01)*, LNCS konferenciasorozat, 2165. köt. 2001, Springer, 23–38. p.

[23] Joost-Pieter Katoen – Maneesh Khattri – Ivan S. Zapreev: A markov reward model checker. In *Second International Conference on the Quantitative Evaluaiton of Systems (QEST 2005), 19-22 September 2005, Torino, Italy* (konferenciaanyag). 2005, 243–244. p. URL `https://doi.org/10.1109/QEST.2005.2`.

[24] Marta Z. Kwiatkowska – Gethin Norman – David Parker: PRISM: probabilistic symbolic model checker. In *Computer Performance Evaluation, Modelling Techniques and Tools 12th International Conference, TOOLS 2002, London, UK, April 14-17, 2002, Proceedings* (konferenciaanyag). 2002, 200–204. p.
URL `https://doi.org/10.1007/3-540-46029-2_13`.

[25] Marta Z. Kwiatkowska – Gethin Norman – David Parker: PRISM 4.0: Verification of probabilistic real-time systems. In *Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings* (konferenciaanyag). 2011, 585–591. p.
URL `https://doi.org/10.1007/978-3-642-22110-1_47`.

[26] Alvin Hudson Rex Nelson: *Útban a modern fizikához*. 1994, LSI OMAK ALAPÍTVÁNY.

[27] Tim Quatmann – Christian Dehnert – Nils Jansen – Sebastian Junges – Joost-Pieter Katoen: Parameter synthesis for markov models: Faster than ever. In *Automated Technology for Verification and Analysis - 14th International Symposium, ATVA 2016, Chiba, Japan, October 17-20, 2016, Proceedings* (konferenciaanyag). 2016, 50–67. p. URL `https://doi.org/10.1007/978-3-319-46520-3_4`.

[28] RWTH Aachen. *PRISM Modelling Language Manual*. 2018. `http://https://moves-rwth.github.io/stormpy/`.

[29] Farzad Samie – Vasileios Tsoutsouras – Lars Bauer – Sotirios Xydis – Dimitrios Soudris – Jörg Henkel: Computation offloading and resource allocation for low-power iot edge devices. In *3rd IEEE World Forum on Internet of Things, WF-IoT 2016, Reston, VA, USA, December 12-14, 2016* (konferenciaanyag). 2016, 7–12. p.
URL `https://doi.org/10.1109/WF-IoT.2016.7845499`.

[30] Adrian FM Smith – Gareth O Roberts: Bayesian computation via the gibbs sampler and related markov chain monte carlo methods. *Journal of the Royal Statistical Society. Series B (Methodological)*, 1993., 3–23. p.

[31] Chao Sun – Xiaosong Hu – Scott J. Moura – Fengchun Sun: Velocity predictors for predictive energy management in hybrid electric vehicles. *IEEE Trans. Contr. Sys. Techn.*, 23. évf. (2015) 3. sz., 1197–1204. p.
URL `https://doi.org/10.1109/TCST.2014.2359176`.

[32] Reinhard von Hanxleden – Dipl-Inf Christian Motika: A synccharts editor based on yakindu set. 2013.

[33] Jasper A Vrugt – Cajo JF ter Braak – Cees GH Diks – Gerrit Schoups: Hydrologic data assimilation using particle markov chain monte carlo simulation: Theory, concepts and applications. *Advances in Water Resources*, 51. évf. (2013), 457–478. p.