BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM
VILLAMOSMÉRNÖKI ÉS INFORMATIKAI KAR
MÉRÉSTECHNIKA ÉS INFORMÁCIÓS RENDSZEREK TANSZÉK

PETRI-HÁLÓN ALAPULÓ MODELLEK ANALÍZISE ÉS ALKALMAZÁSA A REAKCIÓKINETIKÁBAN

Papp Dávid

KONZULENSEK:
Varró-Gyapay Szilvia
BME VIK, Méréstechnika és Információs Rendszerek Tanszék
Dr. Tóth János
BME TTK, Analízis Tanszék (külső konzulens)

2005. július 14.

Nyilatkozat

Alulírott, Papp Dávid, a Budapesti Műszaki és Gazdaságtudományi Egyetem hallgatója kijelentem, hogy ezt a diplomatervet meg nem engedett segítség nélkül, saját magam készítettem, és a diplomatervben csak a megadott forrásokat használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával jelöltem.

. . . . . .

Papp Dávid

Petri-hálón alapuló modellek analízise és alkalmazása a reakciókinetikában

A Petri-háló a hatvanas évek vége óta a mérnöki rendszerek modellezésének és analízisének széles körben használt eszköze, különböző változatainak számtalan alkalmazása van az informatikán belül és kívül egyaránt. Ennek egyik oka, hogy a vizsgálatukra kifejlesztett formális, matematikai elemző módszereken kívül igen praktikus grafikus megjelenés is kapcsolódik hozzájuk, így számítógépi modellező eszközökben a modellezést körülményessé tehető formalizmus megkerülésével, hatékonyan alkalmazhatók. A Petri-hálóhoz tartozó grafikus modell reakciókinetikai rendszerek leírására is használható, azonban az informatikai vizsgálódások során használt absztrakt, strukturális elemző módszerek általában nem ültethetők át közvetlenül a reakciókinetika nyelvére.

Jelen dolgozat célja a Petri-hálón alapuló reakciókinetikai modellek struktúrájának és időbeli fejlődésének vizsgálata az informatikai rendszerek analízise során alkalmazott technikák felhasználásával és kiegészítésével, valamint az ezen vizsgálatok számítógépi támogatásához szükséges reakciókinetikai programcsomag készítése.

A megoldandó strukturális problémák többsége lineáris diofantoszi egyenletrendszerek nemnegatív egészek feletti megoldásainak megkeresésére vezet. A dolgozat ismertet és összehasonlít több korábban publikált megoldó algoritmust, valamint bemutat egy új algoritmust is, amely nagy megoldásokkal rendelkező egyenletek esetében hatékonyabb az elterjedt algoritmusoknál.

A dolgozat áttekinti a Petri-hálók dinamikus viselkedésének leírására kidolgozott módszereket, ezen belül a sztochasztikus szimulációt, valamint a Petri-hálóhoz rendelt differenciálegyenletek megoldásainak kvalitatív és kvantitatív jellemzésének (elsősorban diszkrét) matematikai módszereit.

Bemutatja a megszerzett ismeretek felhasználásával készített programot, amely reakciókinetikai vizsgálatok számítógépi támogatására szolgál. A programcsomaggal megoldható feladatok: bruttó reakciók felbontása elemi reakciólépésekre, kinetikai differenciálegyenletek felírása és megoldása, a megoldások kvalitatív jellemzése (azok előállítása nélkül), valamint a reakciók sztochasztikus szimulációja.

Analysis of Petri net-based Models and their Applications in Reaction Kinetics

Petri nets are widely used tools of systems modelling and engineering since the late sixties; its several variants have countless applications, not only in computer science, but also in other fields of engineering. Not only they have well-established mathematical background, but they are also graphically appealing, which makes them easy to be used efficiently in computer-aided modelling, even without deep knowledge of the formal theory behind them. Their graphical notation can also be used to describe reaction kinetic models, but the translation of the abstract, structural analysis methods of computer science to the language of reaction kinetics is far from straightforward.

The aim of this thesis is to study the structure and temporal evolution of Petri net-based reaction kinetic models using (and developing further) the techniques originally developed during the investigation of engineering models, and to describe and implement a computer application which supports the investigation of these reaction kinetic models.

Most of the underlying structural problems lead to the solution of linear Diophantine systems of equations over nonnegative integers. In the thesis several previously published algorithms are presented and compared to each other, and a new algorithm is proposed, which is more efficient in solving systems with large solutions than the previously known methods.

Methods of describing the dynamical behaviour of Petri nets are also reviewed, such as simulating stochastic Petri nets and identifying the qualitative properties of the solutions of the differential equations which are assigned to Petri nets of reaction kinetic systems. These methods are primarily based on a discrete mathematical approach.

Finally, a computer program based on the above ideas is presented, which supports reaction kinetic investigations. The problems which can be solved with the program include: decomposition of overall reactions into elementary steps, stochastic simulation of chemical reactions, generation and solution of kinetic differential equations, and the identification of certain qualitative properties of the solutions (without solving the equations).

Tartalomjegyzék

Matematikai jelölések jegyzéke

A vektorokat és mátrixokat félkövér, a skalárokat normál betűk jelölik. A vektorok komponensei minden esetben a megfelelő alsó indexelt normál betűk.

$\mathbb{N}_0$	a természetes (nemnegatív egész) számok halmaza
$\mathbb{Z}$	az egész számok halmaza
$\mathbb{Q}_0^+$	a nemnegatív racionális számok halmaza
$\mathbb{R}$, $\mathbb{R}_0^+$	a valós, illetve nemnegatív valós számok halmaza
2^S	az S halmaz hatványhalmaza
$\lvert S\rvert$	az S halmaz elemszáma
$\lVert\boldsymbol{x}\rVert_1$	az $\boldsymbol{x} = (x_1, \ldots, x_n)^T$ vektor hossza: $\sum_{i=1}^{n} \lvert x_i\rvert$
$\lVert\boldsymbol{x}\rVert_\infty$	az $\boldsymbol{x} = (x_1, \ldots, x_n)^T$ vektor maximum normája: $\max_i \lvert x_i\rvert$
$\lVert\boldsymbol{x}\rVert$	az $\boldsymbol{x} = (x_1, \ldots, x_n)^T$ vektor euklideszi normája: $\left(\sum_{i=1}^{n} x_i^2\right)^{\frac{1}{2}}$
$\lVert\boldsymbol{M}\rVert$	az $\boldsymbol{M} = [m_{i,j}]$ mátrix elemei abszolút értékének összege: $\sum_{i,j} \lvert m_{i,j}\rvert$
$\langle\boldsymbol{v}, \boldsymbol{w}\rangle$ vagy $\boldsymbol{v}^T\boldsymbol{w}$	a $\boldsymbol{v}$ és $\boldsymbol{w}$ vektor skaláris szorzata
$\boldsymbol{v} \leq \boldsymbol{w}$	a $\boldsymbol{v}$ vektor komponensenként kisebb vagy egyenlő a $\boldsymbol{w}$ vektornál
$\boldsymbol{v} \lneqq \boldsymbol{w}$	$\boldsymbol{v} \leq \boldsymbol{w}$, de $\boldsymbol{v} \neq \boldsymbol{w}$
$\mathbf{I}_n$	az n-edrendű egységmátrix
$\mathbf{0}$ vagy $\mathbf{0}^n$	az (n dimenziós) nullvektor
$\boldsymbol{e}^{(1)}, \ldots, \boldsymbol{e}^{(n)}$	$\mathbb{Z}^n$ kanonikus bázisa
$\mathcal{S}(\boldsymbol{M})$	az $\boldsymbol{M}$ mátrix oszlopvektorai által kifeszített lineáris altér
$\operatorname{rank}\boldsymbol{M}$	az $\boldsymbol{M}$ mátrix rangja
$(a)_k$	az a szám k-adik leszálló faktoriális hatványa: $\prod_{i=0}^{k-1}(a-i)$
$\arg\min_k f_k$	az unimodális $j \to f_j$ függvény minimumhelye (k: $f_k = \min_j f_j$)
$\mathbf{P}(A)$, $\mathbf{P}(A \mid B)$	az A esemény valószínűsége, illetve feltételes valószínűsége
$\mathbf{E}\,X$	az X valószínűségi változó várható értéke

1. fejezet

Bevezetés

A Petri-háló a hatvanas évek vége óta a mérnöki rendszerek modellezésének és analízisének széles körben használatos eszköze. Különböző változatait használják konkurens vagy elosztott informatikai rendszerek leírására és nemdeterminisztikus vagy sztochasztikus rendszerek modellezésére éppúgy, mint formális logikai következtető rendszerek megjelenítésére, illetve kémiai és vegyipari folyamatok leírására. A különféle Petri-hálós modellek közös előnye, hogy a definiáló formalizmushoz és a hálók vizsgálatához szükséges matematikai módszerekhez szemléletes grafikus megjelenés is kapcsolódik, így a Petri-hálók számítógépes modellező eszközökben a modellezést körülményessé tevő formalizmus megkerülésével, hatékonyan alkalmazhatók.

A Petri-hálóhoz tartozó grafikus modell reakciókinetikai és vegyipari rendszerek leírására is használható – utóbbi környezetben a Petri-hálóhoz igen hasonló P-gráfokat (más néven processz- vagy folyamatgráfokat) használjuk inkább –, azonban az informatikai vizsgálódások során használt absztrakt, strukturális elemző módszerek nem minden esetben ültethetők át közvetlenül a reakciókinetika nyelvére.

E dolgozat célja a Petri-hálón alapuló reakciókinetikai modellek struktúrájának és időbeli fejlődésének vizsgálata az informatikai rendszerek analízise során alkalmazott technikák felhasználásával és kiegészítésével, valamint ezen vizsgálatok számítógépi támogatásához szükséges reakciókinetikai programcsomag készítése. Megvizsgálom a Petri-hálók invariánsait előállító algoritmusokat, és bemutatok egy saját algoritmust, amely nagy invariánsokkal rendelkező, illetve sűrű hálók esetében hatékonyabb az elterjedt algoritmusoknál. Áttekintem a Petri-hálók dinamikus viselkedésének leírására kidolgozott módszereket, ezen belül a sztochasztikus szimulációt, valamint a Petri-hálóhoz rendelt differenciálegyenletek megoldásainak kvalitatív és kvantitatív jellemzésének (elsősorban diszkrét) matematikai módszereit. Végül bemutatom a megszerzett ismeretek felhasználásával készített programcsomagot, amely reakciókinetikai vizsgálatok számítógépes támogatására szolgál.

A 2. fejezet a Petri-hálós modellezés alapfogalmait ismerteti. Bemutatja a Petri-hálós modellek típusait és ezek alkalmazását az informatika és a reakciókinetika területén. Ebből kiderül, hogy a strukturális kérdések jelentős része lineáris diofantoszi egyenletrendszerek nemnegatív egészek feletti megoldásainak keresésére vezet.

A 3. fejezet ennek a problémának a matematikai vizsgálatával foglalkozik, alkalmazásorientált, algoritmikus megközelítésben. Különböző Petri-háló alosztályok esetén hatékony algoritmusokat mutat be, amelyeket – a szerző tudomása szerint – ilyen formában mindeddig nem foglaltak össze. A fejezet egy új algoritmust is tartalmaz, amely reakciókinetikai problémákból származó egyenletek megoldásában gyakran hatékonyabb, mint a korábban publikált algoritmusok.

A 4. fejezet a reakciókinetikai modellek dinamikájával kapcsolatos eredményeket mutat be. Ismerteti a kémiai reakciók egy determinisztikus és egy sztochasztikus modelljét, melyek közül az utóbbi az elméleti számítástudomány területéről ismert sztochasztikus Petri-hálóval ekvivalens. A fejezetet a sztochasztikus szimuláció módszereinek, valamint a determinisztikus és sztochasztikus modell kapcsolatát leíró tételeknek az áttekintése zárja.

Az 5. fejezet mutatja be a *Mathematicában* megvalósított általános célú reakciókinetikai programcsomagot. A tervezési és implementálási kérdések ismertetése után az elkészült program funkcióinak bemutatása következik.

A 6. fejezet már e programcsomaggal elért eredményeket tartalmaz, melyek közül a legérdekesebb egy szervetlen reakció számos felbontásának előállítása.

A dolgozat utolsó, 7. fejezete az eredmények összefoglalása; ezenkívül a diplomatervezés közben megfogalmazódott, de végül nyitva maradt kérdéseket is ismertet.

2. fejezet

Petri-hálók

Ez a fejezet a Petri-hálón alapuló modellezés alapjait foglalja össze. Rövid fogalmi áttekintés után ismerteti a Petri-hálós modellek típusait és a legfontosabb, a diplomatervben is érintett strukturális problémákat. E feladatok megoldásának algoritmikus módszereivel a 3. fejezet, a modellek dinamikai vizsgálatával a dolgozat 4. fejezete foglalkozik.

A részletes formális definíció ismertetése helyett feltételezem, hogy az olvasó ismeri a Petri-háló fogalmát, ezzel kapcsolatosan a [22] tankönyv elnevezéseit használom; a legfontosabb fogalmakat azonban vázlatosan itt is összefoglalom, illetve egy egyszerű példán (2.1. ábra) is bemutatom. A reakciókinetikában jártas olvasót segítheti a 2.1. táblázat, amely a Petri-hálók és formális kémiai mechanizmusok (vagy összetett kémiai reakciók) közötti analógia megvilágítását szolgálja.

2.1. Jelölések és alapfogalmak

A **Petri-hálót** egy $(P, T, w, \mathbf{m}_0)$ rendezett négyes reprezentálja, ahol a **helyek** P halmaza egy tetszőleges véges halmaz, $T \subseteq 2^P \times 2^P$ az **átmenetek** (vagy **tranzíciók**) halmaza, $w : (P \times T) \cup (T \times P) \to \mathbb{N}_0$ a Petri-háló **súlyfüggvénye**, végül az $\mathbf{m}_0 \in \mathbb{N}_0^{|P|}$ vektor a háló **kezdeti jelölése**, ami a Petri-háló automataszerű működésének kezdeti állapota. Működése során a Petri-háló **jelölése** (ami szintén egy $\mathbb{N}_0^{|P|}$-beli vektor[1]) az átmenetek **tüzelései** során változik, a szomszédossági mátrixának (lásd alább) megfelelően. A jelölés vektorának komponenseit a helyekhez tartozó **tokenek** számának nevezzük. A Petri-háló **szomszédossági mátrix**a a súlyfüggvény értékeiből képzett

$$\mathbb{Z}^{|P|\times|T|} \ni \boldsymbol{W} = [w_{p_i,t_j}] := [w(p_i, t_j) - w(t_j, p_i)] \qquad (p_i \in P, t_j \in T)$$

mátrix, ami megmutatja, hogy az egyes átmenetek tüzelése során mennyivel változik a Petri-háló jelölése. (A tüzelésről még lásd később.) Szemléletesebben fogalmazva: a

[1] Időnként kényelmesebb, ezért szokás a jelölést egy $P \to \mathbb{N}_0$ függvényként is definiálni.

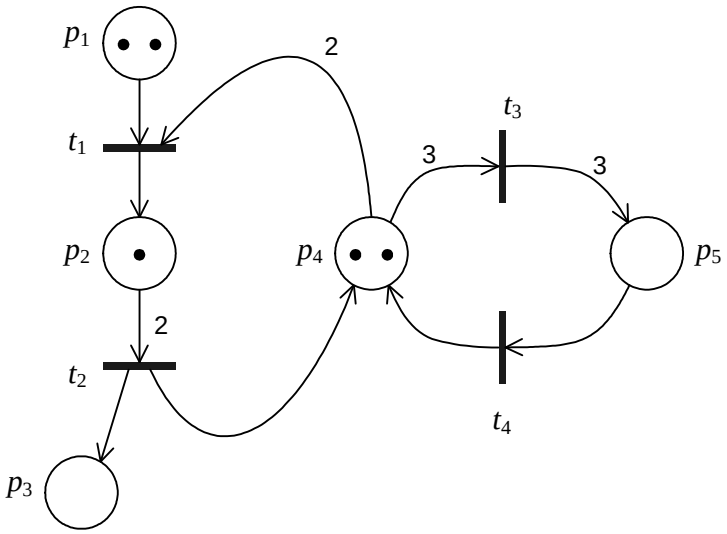
jelölés nélküli Petri-hálóra, mint diszkrét struktúrára tekinthetünk úgy, mint egy irányított páros gráfra, melynek két pontosztálya a helyek illetve az átmenetek halmaza, szomszédossági mátrixa pedig a $\boldsymbol{W}$ mátrix. Ez a kép azonban semmit nem mond a Petri-háló automataszerű működéséről. Egy átmenet tüzelésekor a tokenek az átmenet felé mutató, illetve az átmenetből induló éleken keresztül vándorolnak egyik helyről a másikra, tűnnek el, illetve jelennek meg. A szomszédossági mátrixot szokás két mátrix különbségeként is definiálni, a $\boldsymbol{W}^+$-szal és $\boldsymbol{W}^-$-szal jelölt mátrixok rendre a $w(p_i, t_j)$ és $w(t_j, p_i)$ súlyfüggvény-értékekből álló mátrixok.

Grafikusan a Petri-hálók helyeit körökkel, az átmeneteket vastag, egyenes szakasszal ábrázoljuk, a súlyfüggvény értékeit a megfelelő átmenet és hely között futó nyíl mellé írjuk. (Az alapértelmezett egységnyi súlyt azonban általában nem tüntetjük fel.) A tokeneket a körökbe rajzolt fekete pöttyökkel, vagy – sok token esetén – a helyet jelölő körbe írt számmal jelezzük.

Példa Egy Petri-hálót mutat a 2.1. ábra. Öt helye és négy átmenete van, a szomszédossági mátrixa:

$$\boldsymbol{W} = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 1 & -2 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -2 & 1 & -3 & 1 \\ 0 & 0 & 3 & -1 \end{pmatrix},$$

a kezdeti jelölés vektora $\mathbf{m}_0 = (2, 1, 0, 2, 0)^T$.



2.1. ábra. Petri-háló rajza egy lehetséges kezdeti jelöléssel

A grafikus jelölés és a formalizmus is mutatja, hogy a Petri-hálók fogalma közel áll a vegyipari folyamatok modellezésére használt P-gráfok (*process graphs*), és ezen keresztül a folyamathálózat-szintézis (PNS) fogalmához. A leglényegesebb különbség a P-gráfok és a Petri-hálók között, hogy a P-gráfok jelölése folytonos, a Petri-hálók diszkrét jelölésének megfelelő fogalomra a PNS feladatában nincs szükség [10, 11].

A Petri-háló tekinthető egy nemdeterminisztikus véges automatának is, eltekintve attól, hogy az automataelméletben szokásos automatáktól eltérően a Petri-háló nem végez számítási feladatot, és nincsenek megkülönböztetett elfogadó vagy elutasító állapotai. Egy Petri-hálóval leírt rendszer működését a **tüzelési sorozat**ok írják le, az átmenetek tüzelése felel meg az automata állapotváltásainak. Egy átmenet **engedélyezett**, ha minden bemenő helyén legalább annyi token van, mint amennyi a bemenő helytől az átmenethez vezető él súlya. Egy engedélyezett átmenet tüzelése során minden bemenő helyről elveszünk az átmenethez vezető él súlyával egyező számú tokent, és minden kimenő helyre az odavezető él súlyával egyező számú tokent helyezünk az esetleg már ott található tokenek mellé. Egyszerre csak egy engedélyezett átmenet tüzelhet, nincsen előírás arra, melyik tüzeljen, ha több lehetőség is van. Tüzelések egy sorozata során a tokenszámban bekövetkező változást a háló **állapotegyenlet**e írja le, amely a következőképpen definiálható. Az n darab átmenetet tartalmazó Petri-háló egy tüzelési sorozatához egy $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \ldots, \sigma_n)^T$ vektort rendelhetünk, ahol σ_i mutatja, hogy az i-edik átmenet hányszor tüzel a sorozatban. Ha a tüzelési sorozat előtti jelölés $\mathbf{m}_0$, akkor a tüzelési sorozat utáni jelölés:

$$\mathbf{m}_1 = \mathbf{m}_0 + \boldsymbol{W}\boldsymbol{\sigma}. \tag{2.1}$$

Az előbbi példában t_1 az egyetlen engedélyezett átmenet, minden tüzelési sorozat ennek a tüzelésével kezdődik. A t_1 átmenet tüzelése után az új jelölés $(1, 2, 0, 0, 0)^T$ lesz, amivel a t_2 átmenet is engedélyezetté válik.

2.2. Petri-hálón alapuló modellek típusai

Informatikai rendszerek modellezése során kulcskérdés a megalkotott modell helyességének vizsgálata. A helyesen működő rendszer sajátossága, hogy állapottere a kezdeti jelölés függvényében csak korlátozottan változhat. Az állapottér elérhető állapotaira vonatkozó, megmaradó mennyiségek formájában kifejezhető korlátok a háló *invariáns mennyiségei*. Ezek keresése során a valós időbeli működést nem vizsgáljuk, csupán azt, hogy a Petri-háló átmeneteinek tüzelése során hogyan változhat a háló jelölése (tokeneloszlása). Csak a Petri-háló (diszkrét) struktúrájára koncentrálunk, sem a jelölésnek, sem a tüzelések nemdeterminisztikus jellegének nincsen jelentősége. Ilyenkor tehát egy *diszkrét idejű, diszkrét állapotterű, determinisztikus modellt* vizsgálunk, jellemzően kombinatorikai és lineáris algebrai eszközökkel.

Kémiai rendszerek modellezésekor a rendszer kvalitatív és kvantitatív vizsgálata a célunk. A kvantitatív jellemzéshez a kémiai folyamatban résztvevő anyagok mennyiségének időbeli változását kell leírnunk a Petri-hálóhoz rendelt differenciálegyenletek megoldásával. Sok esetben kézenfekvő, hogy a kémiai reakcióban résztvevő anyagok mennyiségét folytonosnak tekinthetjük, így a kvantitatív vizsgálat során *folytonos idejű, folytonos állapotterű, determinisztikus modellben* dolgozunk.

Mind az informatikai, mind pedig a kémiai modellezésben fontos szerepük van a *sztochasztikus modelleknek*, melyeket időzített átmenetekkel is rendelkező sztochasztikus Petri-hálókkal írhatunk le. Informatikai rendszerek modellezésénél rendszerint ilyen hálók szimulációjával vizsgáljuk a rendszer hosszú távú viselkedését, például hibatűrő rendszerek rendelkezésreállásának számításakor. (Közismert, hogy különféle alkatrészek következő meghibásodásának ideje jól modellezhető exponenciális eloszlású valószínűségi változóval.) Kémiai rendszerek esetén a diszkrét állapotterű modellek alkalmazhatóságára jó példa a sejten belüli folyamatok modellezése, ahol egyes résztvevő enzimek molekuláinak száma legfeljebb tízes nagyságrendű. További, a kémiai reakciók sztochasztikus modellje mellett szóló érveket ismertet pl. [3, 3. fejezet]. Ilyen esetekben tehát egy *folytonos idejű, diszkrét állapotterű, sztochasztikus modellt* vizsgálunk.

A kvalitatív reakciókinetikai vizsgálatok során számos probléma motiválja a „természetes" folytonos idejű modellek mellett a teljesen diszkrét modellek vizsgálatát (ez nem összekeverendő az előző bekezdésben említett diszkrét állapotterű sztochasztikus modellel); elsősorban az, hogy a kémiai folyamatokhoz rendelt differenciálegyenletek megoldása általában szimbolikusan és numerikusan is igen nehéz. Ezért különösen fontosak a folyamat kvalitatív jellemzését szolgáló eljárások, melyek során a keresett koncentráció–idő függvények meghatározása nélkül állapítjuk meg a megoldások jellegzetes tulajdonságait (egyértelműség, nemnegativitás, periodicitás stb). Vegyipari folyamatok tervezésénél a lehetséges reakcióutak kiválasztása, illetve optimalizálása vezet el a Petri-háló-analízishez igen hasonló folyamathálózat-szintézis feladatához, amelyet szintén tisztán kombinatorikai módszerekkel szokás vizsgálni [10, 11]. Ezért a felsorolt modellek kapcsolatának is igen kiterjedt irodalma van [8, 33]. Az informatikai módszerek kutatói ezzel szemben csak a kilencvenes évektől kezdve foglalkoznak a folytonos idejű modellekkel (ilyenek például a hibrid Petri-hálók vagy a fluid sztochasztikus Petri-hálók a formális modellellenőrzésben), ezen vizsgálatok célja és eszközei – a hálókhoz rendelt differenciálegyenletek eltérő jellege miatt – teljesen különböznek a reakciókinetikai vizsgálatokétól. A kémiai vizsgálatok során évtizedek óta használt formalizmus informatikai alkalmazásának lehetősége csak az utóbbi években bukkant fel az irodalomban [12, 23].

2.3. Petri-hálók T- és P-invariánsai

Bármilyen rendszert is modellezünk, a strukturális vizsgálatok alapvető eszköze a rendszer működése során megmaradó mennyiségek keresése. A Petri-háló állapotterének elérhető állapotaira vonatkozó, megmaradó mennyiségek formájában kifejezhető korlátokat a háló invariáns tulajdonságainak nevezzük. Ezek közül a két legalapvetőbb a T- és a P-invariáns.

Egy ciklikus működésű, Petri-hálóval leírt rendszert az jellemez, hogy a működése során bizonyos átmenetek tüzelése után visszaáll az eredeti (azaz a tüzelési sorozat végrehajtása előtti) állapot, minden helyen ugyanannyi token található, mint kezdetben. Formálisan, a (2.1) állapotegyenletből az ilyen tulajdonságú tüzelési sorozatokra, illetve a hozzájuk rendelt $\boldsymbol{\sigma}$ vektorokra

$$\boldsymbol{W\sigma} = 0.$$

Az ezt az egyenletet kielégítő $\boldsymbol{\sigma} \neq 0$ vektorokat a Petri-háló **T-invariáns**ainak nevezzük. (A T betű a *transition* szó kezdőbetűje.) Informatikai rendszerekben általában az erőforrások modellezése vezet el a Petri-háló **P-invariáns**ának fogalmához. (A helyek – *place* – angol megnevezéséből.) Ez a helyek egy olyan nemüres részhalmazának felel meg, ahol a tokenek súlyozott összege a háló működése során (tetszőleges tüzelési sorozat esetén) állandó. Az egyes helyekhez az előbbi súlyt rendelve egy p dimenziós vektort kapunk. Könnyen látható, hogy az így kapott $\boldsymbol{\rho}$ vektorok kielégítik a

$$\boldsymbol{W}^{\mathrm{T}}\boldsymbol{\rho} = 0$$

egyenlőséget. Ezúttal is kikötjük, hogy $\boldsymbol{\rho} \neq 0$ legyen. A fogalmak reakciókinetikai értelmezését lásd a 2.4. szakaszban.

Példa A 2.1. ábrán látható Petri-háló egyetlen T-invariánsa a $(0,0,1,3)^{\mathrm{T}}$ vektor. Az ábráról is azonnal leolvasható, hogy bármilyen jelölés esetén ha a t_3 és a t_4 átmenetek tetszőleges sorrendben egyszer, illetve háromszor tüzelnek, visszakapjuk a négy tüzelést megelőző jelölést. (Feltételezve, hogy a tüzelési sorozat minden lépésében engedélyezve van az éppen sorra kerülő átmenet.)

A Petri-hálónak két független P-invariánsa is van, ezek a $(0,2,3,1,1)^{\mathrm{T}}$ és $(1,1,2,0,0)^{\mathrm{T}}$ vektorok. Ezek az ábrán is könnyen ellenőrizhetőek, például a p_1, p_2, p_3 helyeken lévő tokenek számának az $1,1,2$ súlyokkal képzett összege valóban mind a négy átmenet tüzelése során változatlan marad.

Mindkét invariánsszámítási feladat tehát homogén, lineáris, egész együtthatós egyenletrendszerek megoldásaként formalizálható (az egyenletrendszer mátrixa a Petri-háló szomszédossági mátrixa, vagy annak transzponáltja), azonban eltérő fogalmakat kaphatunk, ha eltérő halmaz feletti megoldásokat keresünk [22, 2.11.3 szakasz]. Minthogy az

átmenetek nemnegatív egész számszor tüzelhetnek, közvetlen gyakorlati, kombinatorikai jelentést a (2.1) állapotegyenlet nemnegatív egészek feletti megoldásainak tulajdoníthatunk. A feladatunk matematikai megfogalmazásban tehát *homogén lineáris diofantoszi egyenletrendszerek megoldása a természetes számok felett*.

Könnyen látható, hogy ha egyáltalán van megoldás, akkor végtelen sok van, ezért jellemzően a megoldások egy *véges generátorát* keressük, máskor szükség lehet minden megoldás felsorolására egy bizonyos korlátig. A generátorok közül *a* $\leq$ *relációra nézve minimális* megoldások, valamint a *minimális tartójú* megoldások vizsgálata szokásos. (Egy $\boldsymbol{\nu}$ vektor **tartó**jának az $\{i \mid \nu_i \neq 0\}$ halmazt nevezzük.) A dolgozat mindegyik feladat megoldására ismertet algoritmust. A 3. fejezet az ezekkel kapcsolatos irodalmi áttekintést és a saját eredményeimet is tartalmazza.

2.4. Petri-hálók a reakciókinetikában

Az előző alfejezetben definiált invariánsoknak kémiai jelentésük is van. Hasonló feladatokra vezet ezenkívül a reakciók felbontásának feladata is. Ezeket a feladatokat ismerteti ez az alfejezet.

A kémiai mechanizmusok is jól modellezhetők Petri-hálókkal. Az egyes **anyagfajták**nak helyeket, a **reakció**knak átmeneteket feleltetünk meg. A súlyfüggvény megmutatja, az egyes anyagfajtákból mennyi fogy, illetve termelődik a reakciókban. A Petri-háló szomszédossági mátrixát reakciókinetikai szövegkörnyezetben is szokás két mátrix különbségeként definiálni: a szokásosan α-val jelölt mátrix tartalmazza a helyektől átmenetek felé mutató élek súlyát (ez mutatja, mennyi anyag fogy a reakciókban), míg a β mátrix az átmenetektől a helyek felé menő élek súlyaiból áll (ez a reakciókban termelődő anyagok mennyiségét mutatja). Az ezekből kapható $\gamma := \beta - \alpha$ mátrix, a **sztöchiometriai együtthatók** mátrixa, azonos a korábbiakban $\boldsymbol{W}$-vel jelölt szomszédossági mátrixszal.

A T-invariánsok a reakciókhoz rendelnek nemnegatív egész együtthatókat úgy, hogy a reakciók ezen együtthatókkal súlyozott eredője zérus. Tehát a T-invariánsok a kémiai mechanizmus *körfolyamatainak* felelnek meg. A P-invariánsok az anyagokhoz rendelnek súlyt úgy, hogy minden reakció két oldalán található anyagok összsúlya egyenlő. Tehát a P-invariánsok léte a *tömegmegmaradásnak* felel meg.

Kémiai reakciók modellezésekor felmerülő további alapfeladat a *bruttó reakciók felbontásának* feladata, melynek során azt kell meghatároznunk, hogy egy tetszőleges reakció – ez az ún. **bruttó reakció** – milyen elemi lépések sorozataként hajtódhat végre. Egy reakciólépés **elemi**, ha legfeljebb két atom, molekula stb. vesz benne részt. Ezt a definíciót az indokolja, hogy ahhoz, hogy egy reakció végbemenjen, a résztvevő részecskéknek térben és időben találkozniuk kell, három anyag találkozásának a valószínűsége azonban gyakorlatilag zérus. Ha adva vannak az elemi lépések, akkor ezek hálózata modellezhető

egyetlen Petri-hálóval, a feladatunk pedig az, hogy ebben a Petri-hálóban olyan átmeneteket keressünk, melyek hatásának eredője a felbontandó bruttó reakciónak megfelelő. Pontosabban fogalmazva, m anyagfajta esetén minden reakcióhoz egy m dimenziós vektort rendelhetünk, amelynek i-edik komponense ($1 \leq i \leq m$) megmutatja, hogy az i-edik anyagfajtából mennyi fogy, illetve termelődik az adott reakcióban. (Ez tehát az elemi reakciók esetén nem más, mint a γ mátrix i-edik oszlopvektora, a bruttó reakció esetén egy további vektor.) A feladatunk a bruttó reakcióhoz rendelt vektor előállítása a γ mátrix oszlopvektorainak nemnegatív egész együtthatós lineáris kombinációjaként az összes lehetséges módon, azaz egy *inhomogén lineáris diofantoszi egyenletrendszer megoldása a természetes számok felett*.

Az előző feladatban adva voltak az elemi reakciólépések, de ez nem szükségképpen van így. Ha csak a bruttó reakcióban esetleg résztvevő anyagok ismertek, az elemi reakciólépéseket algoritmikusan is előállíthatjuk, figyelembe véve az atom- és töltésmegmaradás törvényét, melyek szerint egy reakció két oldalán minden atomból ugyanannyi szerepel, és a töltések összege is megegyezik. Ez a következőképpen formalizálható: ha a résztvevő anyagokat összesen k-féle atom építi fel, úgy minden anyaghoz egy $(k+1)$ dimenziós vektort rendelhetünk, amelynek i-edik komponense ($1 \leq i \leq k$) megmutatja, hogy az i-edik atomból mennyi található az anyagban, az utolsó komponens pedig a töltés (elemi töltés egységben kifejezve). Az atom- és töltésmegmaradás törvénye szerint egy reakció két oldalán található anyagok vektorainak a reakcióbéli együtthatóikkal súlyozott összege egyenlő. Ezt összevetve az elemi reakciók előbbi definíciójával (mely szerint legfeljebb két anyag lehet a reakcióegyenlet bal oldalán) a feladatunk az, hogy állítsuk elő minden anyag vektorát, azok kétszeresét, továbbá bármely két anyag vektorának összegét a többi anyag vektorának nemnegatív egész együtthatós lineáris kombinációjaként. Ez ismét inhomogén lineáris diofantoszi egyenletrendszerek megoldását jelenti a természetes számok felett.

A két utolsó feladat matematikai modellje az előbbiek alapján teljesen azonos, a feladatokból adódó egyenletek eltérő jellege miatt a megoldásuk mégis eltérő módszereket igényel. Az elemi reakciók előállítása során sok kisebb egyenlet megoldására van szükség, melyek mindig véges sok megoldással bírnak, ráadásul ezek a megoldások elég rövidek, azaz a komponenseik összege elég kicsiny. Ezek a tulajdonságok nagyon jól kihasználhatók, ilyen algoritmusokat ismertet a 3.1.1. és a 3.3.1. szakasz. A bruttó reakció elemi lépésekre bontása során mindezek a tulajdonságok elvesznek. Egyetlen nagy méretű egyenletrendszert kell megoldanunk (a változók és az egyenletek száma százas vagy akár ezres nagyságrendű is lehet), amelynek gyakran végtelen sok megoldása van (ilyenkor minimális megoldásokat keresünk), és a legkisebb megoldások mérete általában jóval meghaladja az elemi reakciók előállításának feladatában kapható megoldások méretét.

PN terminológia	kémiai interpretáció	matematikai fogalom
háló	összetett kémiai reakció, kémiai mechanizmus	irányított páros gráf
helyek	anyagfajták	P pontosztály
átmenetek	reakciólépések	T pontosztály
bemenő hely	reaktáns	—
kimenő hely	termék	—
jelölés	anyagmennyiségek	$M : P \to \mathbb{N}_0$ függvény
token	molekula	M értékei
súlyfüggvény	—	élek multiplicitása
szomszédossági mátrix	sztöchiometriai együtthatók mátrixa	szomszédossági mátrix
—	reakciósebesség	$w : T \to \mathbb{R}^+$ függvény
engedélyezett átmenet	potenciálisan bekövetkező reakciólépés	—
tüzelés	reakciólépés bekövetkezése	—

2.1. táblázat. Szótár Petri-hálós modellek értelmezéséhez

A dolgozatban mind az informatikai, mind a reakciókinetikai alkalmazásokat szem előtt tartva vizsgálom a Petri-hálók analízisének módszereit. Mivel az egyes területeken érdekes feladatoknak nem minden esetben létezik természetes analogonja a másik területen, mindig a vizsgált terület terminológiáját, vagy a megfelelő matematikai fogalmakat használom. Az érintett tudományterületek közötti váltást elősegítendő az összetartozó fogalmakat, megnevezéseket táblázatosan is összefoglaltam (2.1. táblázat).

3. fejezet

Lineáris diofantoszi egyenletrendszerek megoldása természetes számok felett

Az előző fejezetben írottak alapján a Petri-hálók invariánsainak keresése, a bruttó reakciók felbontása elemi reakciólépésekre, illetve a megadott anyagfajták között végbemenő elemi reakciólépések előállítása egyaránt lineáris diofantoszi egyenletrendszerek nemnegatív egész számok feletti megoldására vezető feladat. E problémák mégis igen különbözők: a homogén és az inhomogén egyenletrendszer megoldása; az összes megoldás, a minimális megoldások, illetve a minimális tartójú megoldások keresése egyaránt szerepel a megoldandó feladatok között. Ez a fejezet ezt a problémacsaládot járja körül, az irodalmi áttekintés mellett korábban publikált algoritmusok kiegészítéseit, illetve egy saját algoritmust is tartalmaz. A fejezetet alkalmazási példák ismertetése zárja. Formálisan a következő a feladatunk.

Feladat. Legyen adva az egészekből álló $\mathbf{b} \in \mathbb{Z}^d$ **felbontandó vektor**, továbbá a **particionáló vektorok** egy $\{\boldsymbol{\nu}_1, \boldsymbol{\nu}_2, \ldots, \boldsymbol{\nu}_n\}$ halmaza, $\boldsymbol{\nu}_i \in \mathbb{Z}^d$. Keressük mindazokat az $(\alpha_1, \alpha_2, \ldots, \alpha_n) \in \mathbb{N}_0^n$ természetes szám n-eseket, amelyekre

$$\sum_{i=1}^{n} \alpha_i \boldsymbol{\nu}_i = \mathbf{b}. \tag{3.1}$$

A vizsgált feladat szemléletesen szoros rokonságban áll a ládapakolás, részhalmazösszeg problémájával és más hasonló problémákkal [25], így nem várhatjuk, hogy a megoldására gyors algoritmusttalálunk. A teljesség kedvéért pontosan is kimondjuk és bizonyítjuk a feladat bonyolultságára vonatkozó tételt.

3.1. tétel *A lineáris diofantoszi egyenletrendszerek nemnegatív egészek feletti megoldhatóságának eldöntése* **NP***-nehéz.*

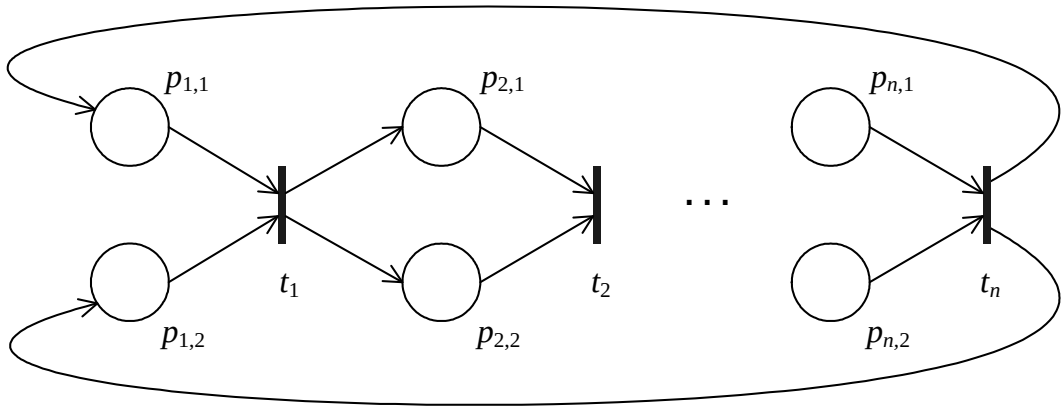
BIZONYÍTÁS Megadunk egy Karp-redukciót (polinomidejű visszavezetést) a közismerten **NP**-teljes részhalmazösszeg (RH) problémáról [25, 8.7. alfejezet] a vizsgált (LDE) problémára. Legyen RH egy példánya $(s_1, \ldots, s_n; b)$, ahol tehát egy olyan $I \subseteq \{1, \ldots, n\}$ indexhalmazt keresünk, amelyre $\sum_{i \in I} s_i = b$ teljesül. Képezzük ebből LDE egy példányát a következő módon: az egyenletrendszer mátrixa legyen az $(n+1) \times (2n)$ dimenziós

$$\begin{pmatrix} s_1 & \cdots & s_n & 0 & \cdots & 0 \\ & \mathbf{I}_n & & & \mathbf{I}_n & \end{pmatrix}$$

mátrix, ennek oszlopvektorai a particionáló vektorok. A felbontandó vektor legyen az $(n+1)$-dimenziós $(b, 1, \ldots, 1)^T$ vektor. Azonnal látható, hogy ennek az egyenletrendszernek minden nemnegatív egészek feletti megoldása csak 0 és 1 értékű komponensekből állhat, mégpedig úgy, hogy az i-edik ($1 \leq i \leq n$) vektor együtthatója pontosan akkor 1, ha az $(i+n)$-edik vektoré zérus. Az egyenletrendszer első egyenletéből ezek után az is világosan látszik, hogy az RH feladatnak pontosan akkor megoldása egy I indexhalmaz, ha a hozzárendelt LDE problémapéldánynak megoldása az a vektor, amelyet I karakterisztikus vektorának, illetve annak komplemensének egymás után fűzésével kapunk. Tehát az RH$\prec$LDE Karp-redukciónk helyes, így az LDE probléma az RH-hoz hasonlóan **NP**-nehéz. ∎

A megoldások felsorolása exponenciális tárkomplexitású feladat, mert csupán a minimális tartójú megoldások száma is exponenciálisan nagy lehet az egyenletrendszer méretéhez képest [20]. Tekintsünk például egy homogén egyenletet, melynek egyetlen minimális (és egyben minimális tartójú) megoldása az $(\alpha_1, \alpha_2, \ldots, \alpha_n) = (1, \ldots, 1)$ vektor. Az egyenletben minden ν_i megkettőzésével egy továbbra is n egyenletből álló, de már $2n$ változós egyenletrendszert kapunk, melynek 2^n minimális tartójú megoldása van. A konstrukciót szemlélteti a 3.1. ábra. Ezen egy olyan Petri-háló látható, melynek $2n$ számú helye, n átmenete és $4n$ éle van, míg minimális tartójú P-invariánsainak száma 2^n.

A diofantoszi egyenletekkel foglalkozó irodalom a lineáris egyenletrendszerekkel csupán a magasabbfokú egyenletek megoldásának segédeszközként foglakozik, az összes nemnegatív megoldás előállításának hatékony módszereiről azonban a nagyobb összefoglaló művekben [27,34] sem olvashatunk. Valójában a megoldhatóság kérdése (illetve a megoldások algebrai jellemzése) is igen nehéz [18]. A *Maple* matematikai programcsomag sem tartalmaz olyan függvényt, amivel a megoldások előállíthatók. A *Mathematica* legú-



3.1. ábra. Exponenciális számú minimális tartójú P-invariánssal rendelkező Petri-háló

jabb (5-ös) verziója már igen, azonban ez sem elég hatékony nagy egyenletrendszerek kezeléséhez. (A *Mathematica* által alkalmazott algoritmusról lásd a 3.1.1. szakaszt.)

Homogén egyenletrendszerek esetén ha van megoldás, akkor végtelen sok van. Ilyenkor a megoldások felsorolásának igénye értelmetlen, de bizonyos alkalmazások esetén hasznos lehet az összes megoldás felsorolása egy bizonyos méretkorlátig. Ilyen korlátozó feltétel reakciókinetikai feladatok megoldásánál is adódhat, például ha egy bruttó reakció végtelen sok felbontása közül csak korlátos számú lépésből állókat keresünk. Máskor a megoldások egy „érdekes" részhalmazát keressük; szokásosan a $\leq$ relációra nézve minimális, vagy a minimális tartójú megoldások halmazát. A minimális megoldásoknak kémiai jelentést is tulajdoníthatunk: ezek a kört (azaz zéró eredőjű lépéssorozatot) nem tartalmazó felbontások.

A következő segédtételek [7, 20, 24] mutatják, hogy a feladatunk így is értelmes, valamint hogy az említett „bázisok" valóban véges módon reprezentálják az összes megoldást.

3.2. lemma *Ha a* (3.1) *egyenletrendszer homogén, azaz* $\mathbf{b} = \mathbf{0}$*, akkor a megoldásaira teljesülnek az alábbiak.*

1. *A minimális megoldások száma véges.*
2. *A megoldások száma akkor és csak akkor véges, ha minden megoldás minimális.*
3. *Minden megoldás előáll minimális megoldások nemnegatív együtthatós lineáris kombinációjaként, speciálisan nemnegatív egész, illetve konvex kombinációként is.*
4. *Minden minimális megoldás (és így minden megoldás) előáll minimális tartójú megoldások nemnegatív együtthatós lineáris kombinációjaként.*

Érdemes megjegyezni, hogy az első két állítás inhomogén egyenletrendszerekre is teljesül, tehát általában is értelmes feladat a megoldások helyett csupán a minimális megoldásokat keresnünk.

3.1. A homogén egyenlet megoldása

Ebben az alfejezetben az invariánsok meghatározásakor megoldandó homogén egyenletrendszereket megoldó algoritmusok olvashatók, formálisan tehát a megoldandó egyenlet:

$$\sum_{i=1}^{n} \alpha_i \nu_i = 0, \tag{3.2}$$

amelynek most csak *a nullvektortól különböző* megoldásait keressük. A nullvektort nem tekintjük semmilyen Petri-háló T- vagy P-invariánsának, ezenkívül a minimális, illetve minimális tartójú megoldások értelmezése is így teljes.

Elsőként a szakirodalomban leggyakrabban hivatkozott Contejean–Devie-algoritmust és egy kiegészítését mutatom be, amely (3.2) minimális megoldásait adja meg. Ezután a minimális tartójú megoldásokat előállító Martínez–Silva-algoritmus ismertetése következik, amely az invariánsok kiszámításának másik gyakran alkalmazott módszere. Mindkét algoritmus adaptálható inhomogén egyenletrendszerekre is. Végül egy saját algoritmust mutatok be, amely minden hosszkorlátos megoldást megad, és módosítás nélkül alkalmazható inhomogén egyenletrendszerek esetén is. (Inhomogén egyenletrendszerek esetén még hosszkorlátra sincs szükség, ha véges sok megoldás van.)

A minimális megoldások méretének becslésével kapcsolatos eredményeket a 3.2. alfejezet foglalja össze, az inhomogén egyenletekkel pedig a 3.3. alfejezet foglalkozik.

3.1.1. Contejean és Devie algoritmusa

Ebben a szakaszban Contejean és Devie algoritmusát [6] és annak egy általam javasolt javítását [21] mutatom be. Ez az algoritmus, amely a homogén egyenletrendszer minimális megoldásait sorolja fel, az irodalomban a lineáris diofantoszi egyenletrendszerek nemnegatív egészek feletti megoldásának legtöbbet idézett algoritmusa. Ezen alapul a *Mathematica* 5.0 verziójának megoldórutinja is [35], de az implementáció részletei nem nyilvánosak.

Ahogy az inhomogén egyenletrendszer megoldásait előállító naiv algoritmus (lásd a 3.3.1. szakaszban) felfogható úgy, mint a keresési tér mélységi bejárása, a Contejean–Devie-algoritmus a szélességi keresés (*breadth-first search*, BFS) ötletes módosításának tekinthető, amely nem csak gyorsabb és tárhatékonyabb, mint az eredeti BFS eljárás, hanem garantáltan be is fejeződik. A homogén egyenletrendszer megoldásait előállító BFS eljárás alapváltozata a 3.1. algoritmus.

Az eljárás működési elve, hogy az origóból kiindulva egyesével növeljük az éppen vizsgált vektor koordinátáit. Ha megoldást kaptunk, akkor azt megjegyezzük, és a meg-

3.1. algoritmus: A (3.2) egyenlet megoldása szélességi kereséssel

Bemenet: $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n$
Kimenet: A (3.2) egyenletet kielégítő $(\alpha_1, \alpha_2, \ldots, \alpha_n)$ vektorok.

1 $A \leftarrow \{\mathbf{0}^n\}$, $M \leftarrow \emptyset$;
2 $A \leftarrow \{\boldsymbol{a} + \boldsymbol{e}^{(k)} \mid \boldsymbol{a} \in A, 1 \leq k \leq n\}$; `// BFS lépés`
3 $M \leftarrow M \cup \{\boldsymbol{a} \in A \mid \boldsymbol{a}$ megoldása (3.2)-nek$\}$; `// új megoldások`
4 $A \leftarrow A \setminus \{\boldsymbol{a} \mid \exists \boldsymbol{m} \in M : \boldsymbol{m} \leq \boldsymbol{a}\}$; `// rossz ágak levágása`
5 Ha $A = \emptyset$, megállunk, a megoldáshalmaz M. Különben ugorjunk a 2. lépésre.

oldásvektorból nem lépünk tovább. Hasonlóan levágjuk a keresési fa minden olyan ágát, amely egy már megtalált megoldásnál komponensenként nagyobb vagy egyenlő vektorok felé vezet, így garantáltan csak minimális megoldásokat kapunk. Az eljárás az M halmazban gyűjti a minimális megoldásokat, az A halmazban a keresési fa még megvizsgálandó pontjai találhatók.

Könnyen meggondolható, hogy az eljárás minden minimális megoldást megtalál, de nem biztos, hogy véges sok lépésben befejeződik. Ennek orvoslására a [6] publikációban javasolt módosítás a következő: a 2. lépésben az $\boldsymbol{a}$ vektort csak akkor növeljük $\boldsymbol{e}^{(k)}$-val, ha a $\langle \sum_{i=1}^{n} a_i \boldsymbol{v}_i, \boldsymbol{v}_k \rangle$ skalárszorzat negatív; szemléletesen: az eljárás mindig úgy lép tovább, hogy a megoldásvektor-jelöltet a megoldandó (3.2) egyenlet bal oldalába helyettesítve kapható lineáris kombináció a vektor k-adik komponensének növelésével „az origó irányában" változzon. Tehát a Contejean–Devie-algoritmus 2. lépése a következő:

3.2. algoritmus: A Contejean–Devie-algoritmus

1 Azonos a 3.1. algoritmussal, kivéve annak 2. lépését, amely most:
2 $A \leftarrow \{\boldsymbol{a} + \boldsymbol{e}^{(k)} \mid \boldsymbol{a} \in A,\ 1 \leq k \leq n, \langle \sum_{i=1}^{n} a_i \boldsymbol{v}_i, \boldsymbol{v}_k \rangle < 0\}$; `// új BFS lépés`

Könnyen belátható, hogy az eljárás így is teljes, a [6] cikk fő eredménye, hogy így már véges sok lépésben be is fejeződik, tehát az algoritmus valóban helyes és teljes is.

Az algoritmus további előnyös tulajdonsága, hogy rendkívül tárhatékonyan is implementálható annak ellenére, hogy nem mélységi, hanem szélességi keresésen alapul: az A halmaz, melyben a részeredményeket gyűjtjük, egy n mélységű veremként is megvalósítható, azaz hatékony implementáció esetén n-nél több részeredmény tárolására nincs szükség.

Ha a minimális megoldások hosszára felső korlátunk is van, az A halmazba nem kell ennél hosszabb vektorokat tennünk. (Emlékeztetőül, a vektorok hosszán a dolgozatban a komponensek abszolút értékeinek összegét értjük, nem az euklideszi normáját.) Önálló eredményem, hogy az eljárás ebben az esetben tovább javítható, hogy még jobban csökkenjen a szükséges lépések száma.

3.3. tétel *A* (3.2) *egyenlet legfeljebb* m *hosszúságú minimális megoldásait a Contejean–Devie-algoritmus következő módosításával is megkapjuk: a 2. lépésben csak azokat az* $\mathbf{a} \in A$ *vektorokat növeljük* $\mathbf{e}^{(k)}$*-val, amelyekre* $m > \|\mathbf{a}\|_1$*, és*

$$\langle \sum_{i=1}^{n} a_i \boldsymbol{\nu}_i,\ \boldsymbol{\nu}_k \rangle \leq -\frac{\| \sum_{i=1}^{n} a_i \boldsymbol{\nu}_i \|^2}{m - \|\mathbf{a}\|_1} < 0. \tag{3.3}$$

BIZONYÍTÁS A módosított algoritmus végessége a keresett megoldásokra vonatkozó méretkorlátból azonnal következik, csak a teljességet kell igazolnunk, azaz azt, hogy továbbra is megkapunk minden minimális megoldást, amelynek hossza legfeljebb m.

Legyen $\mathbf{a}^* = (a_1^*, a_2^*, \ldots, a_n^*)$ minimális megoldás, $\mathbf{a} \lneqq \mathbf{a}^*$, és legyen $a_i^* = a_i + r_i$ $(i = 1, ..., n)$. Ekkor

$$\begin{aligned} 0 = \| \sum_i a_i^* \boldsymbol{\nu}_i \|^2 &= \| \sum_i a_i \boldsymbol{\nu}_i \|^2 + \| \sum_i r_i \boldsymbol{\nu}_i \|^2 + 2\langle \sum_i a_i \boldsymbol{\nu}_i,\ \sum_i r_i \boldsymbol{\nu}_i \rangle \\ &= 2\| \sum_i a_i \boldsymbol{\nu}_i \|^2 + 2\langle \sum_i a_i \boldsymbol{\nu}_i,\ \sum_i r_i \boldsymbol{\nu}_i \rangle. \end{aligned}$$

Az utolsó egyenlőség annak következménye, hogy $\mathbf{a}^*$ megoldás, azaz $\sum_i (a_i + r_i)\boldsymbol{\nu}_i = \mathbf{0}$. A kapott egyenletet átrendezve:

$$\begin{aligned} -\| \sum_i a_i \boldsymbol{\nu}_i \|^2 &= \langle \sum_i a_i \boldsymbol{\nu}_i,\ \sum_j r_j \boldsymbol{\nu}_j \rangle \\ &= \sum_j r_j \langle \sum_i a_i \boldsymbol{\nu}_i,\ \boldsymbol{\nu}_j \rangle \\ &\geq (\sum_j r_j) \min_k \langle \sum_i a_i \boldsymbol{\nu}_i,\ \boldsymbol{\nu}_k \rangle \\ &= (m - \|\mathbf{a}\|_1) \min_k \langle \sum_i a_i \boldsymbol{\nu}_i,\ \boldsymbol{\nu}_k \rangle. \end{aligned}$$

Az egyenlőtlenség mindkét oldalát a pozitív $(m - \|\mathbf{a}\|_1)$ számmal osztva kapjuk, hogy a jobb oldalon álló minimumot adó k indexhez tartozó $\langle \sum_i a_i \boldsymbol{\nu}_i,\ \boldsymbol{\nu}_k \rangle$ kifejezés kielégíti a (3.3) egyenlőtlenséget. Így az algoritmus a javasolt 2. lépéssel is tovább tud lépni minden $\mathbf{a} \in A$ vektorból minden $\mathbf{a}^* \geq \mathbf{a}$-t teljesítő $\mathbf{a}^*$ minimális megoldás irányába, tehát megtalál minden minimális megoldást. ∎

Az előbbi tétel $m \to \infty$ határeseteként az idézett [6]-beli feltétel adódik (a bizonyításunk tehát az eredeti Contejean–Devie-algoritmus helyességére is jó bizonyítás), a lépésszám annál jobban csökken az eredeti algoritmus lépésszámához képest, minél kisebb az m korlát. Sajnos a Contejean–Devie-algoritmus helyességének bizonyításából (sem az eredetiből, sem a fentiből) semmilyen korlát nem adódik annak lépésszámára. Ilyen korlá-

tokat más, algebrai megfontolásokkal kaphatunk – néhány eredményt a 3.2. alfejezetben mutatunk be –, illetve maga az alkalmazás is adhat. Mind informatikai, mind pedig reakciókinetikai vizsgálódásoknál használható ez a megfigyelés, az informatikai modellek ugyanis gyakran igen kicsi invariánsokkal bírnak, és kémiai reakciók felbontása során is előnyben részesíthetjük a kevesebb lépésből álló, egyszerűbb felbontásokat.

3.1.2. Martínez és Silva algoritmusa

Az invariánsok meghatározására az egyik legegyszerűbb megoldás egy 1982-ben J. Martínez és M. Silva által publikált algoritmus [20], amely a minimális tartójú megoldások előállítására szolgál. Az algoritmus alapváltozata – a Contejean–Devie-algoritmussal szemben – valójában számos nemkívánt invariánst is előállít (nem csupán nem minimális tartójúakat, hanem akár a $\leq$ részbenrendezésre sem minimálisakat is), csupán azt garantálja, hogy minden minimális tartójú megoldás szerepel az előállított invariánsok között. Az algoritmus pszeudokódja a 3.3. algoritmus.

3.3. algoritmus: A Martínez–Silva-algoritmus

Bemenet: A Petri-háló $\boldsymbol{W} \in \mathbb{Z}^{m \times n}$ szomszédossági mátrixa
Kimenet: A Petri-háló P-invariánsaiból (mint sorvektorokból) álló $\mathbf{P}$ mátrix

Jelölés: A mindenkori $\mathbf{M}$ mátrix sorainak száma s, k-adik sora $\mathbf{m}_k$.

```
1 M ← [W I_n];
2 for i = 1 to m do
3     for j = 1 to s − 1 do
4         for k = j + 1 to s do
5             Ha m_j és m_k i-edik komponense különböző előjelű (és nem nulla),
              akkor adjuk M-hez a Kombinál(i, m_j, m_k) vektort.
6     Töröljük M mindazon sorát, amelynek i-edik eleme nem zérus.
7 Legyen P az M mátrix első m oszlopának törlésével kapott mátrix.
```

Az algoritmus egy segédfüggvényt használ. A `Kombinál`$(i, \boldsymbol{v}, \boldsymbol{w})$ függvény akkor van értelmezve, ha $v_i w_i < 0$, és értéke ilyenkor az az általában egyértelműen meghatározott $\lambda_1 \boldsymbol{v} + \lambda_2 \boldsymbol{w}$ $(\lambda_1, \lambda_2 \in \mathbb{Q}^+)$ egészekből álló vektor, melynek i-edik komponense nulla, és amelynek komponensei relatív prímek. Ezt a vektort úgy állíthatjuk elő, hogy a $w_i \boldsymbol{v} + v_i \boldsymbol{w}$ vektort – ha az nem a nullvektor – elosztjuk a komponensei legnagyobb közös osztójával. Ha $w_i \boldsymbol{v} + v_i \boldsymbol{w} = \mathbf{0}$, akkor ez az előbbi definíció pontatlan, helyette `Kombinál`$(i, \boldsymbol{v}, \boldsymbol{w}) := \mathbf{0}$.

Az algoritmus gondolatmenete a szemléletesebb Petri-hálós megfogalmazásban az, hogy sorra vesszük a háló átmeneteit, és minden átmenetre meghatározzuk a bemenő és kimenő helyekből álló helypárok jelölésének minden olyan nemnegatív egész együtthatós lineáris kombinációját, amelyet a kérdéses átmenet nem változtat meg. Egy átmenet

sorra vétele után (azaz a későbbi átmenetek vizsgálata során) az összekombinált helyekre már mint „összevont" helyekre tekintünk, amelyeken ezután állandónak tekintett (a már sorra vett átmenetek tüzelése során nem változó) számú token van, az eredeti bemenő és kimenő helyeket pedig töröljük. Az eljárás végén csak összevont helyek maradnak. Az ezeket alkotó eredeti Petri-háló helyek indexei az invariánsok tartóhalmazai. Ez alapján az algoritmus helyessége – amin most tehát csak annyit értünk, hogy a felsorolt megoldások között minden minimális tartójú megoldás szerepel – teljes indukcióval megmutatható.

Az algoritmus ismertetett változatának végén a minimális tartójú megoldásokat ki kell válogatnunk az összes kapott megoldás közül. Ez algoritmikus szempontból semmilyen nehézséggel nem jár, azonban az algoritmus idő- és tárigényét csökkentendő érdemes a nem kívánt megoldásokat már futás közben kiszűrni. Jelölje a (3.2) egyenlet mátrixának (azaz a $\boldsymbol{W}$ mátrixnak) i-edik sorát $\boldsymbol{w}_i^T$. Egyszerűen igazolható a következő állítás [7,20]:

3.4. tétel *Az egyenlet egy* $\{j_1, j_2, \ldots, j_q\}$ *tartóhalmazú megoldása akkor és csak akkor minimális tartójú, ha*

$$\operatorname{rank}\left(\boldsymbol{w}_{j_1}\ \boldsymbol{w}_{j_2}\ \cdots\ \boldsymbol{w}_{j_q}\right) = q - 1.$$

Mivel az algoritmus során az összevont helyhalmazok folyamatosan bővülnek, a nem minimális tartójú invariánsok menet közbeni törlése nem változtat az algoritmus teljességén. Így a fenti tételt beépíthetjük az eljárásba úgy, hogy ha egy állapotösszevonás után egy invariánst kaptunk (az $\boldsymbol{M}$ mátrixban egy csupa nulla sor keletkezett), akkor az előbbi rangfeltétellel ellenőrizzük annak minimalitását. Ez a megoldás a gyakorlatban sokszor nem hatékony, mert a rangfeltétel ellenőrzése lassú. Ilyenkor a szükséges és elégséges feltétel gyengítésével gyorsíthatunk. A 3.4. tételben szereplő mátrix rangját a nemnulla sorainak számával becsülhetjük felülről, ezzel egy szükséges feltételt kapunk a megoldás minimalitására, melynek segítségével már a külső for-ciklus futása közben kiszűrhető a nem kívánt megoldások egy része. Természetesen ez után az egyszerűsítés után ismételten össze kell hasonlítanunk a megoldásokat, hogy törölhessük a nem minimális tartójúakat.

A Martínez–Silva-algoritmus komoly hátránya, hogy általában szükségtelenül nagy tárigényű. Szemben a rendkívül tárhatékonyan implementálható Contejean–Devie-algoritmussal, amely – megfelelő implementáció esetén – a minimális megoldások megjegyzéséhez szükséges táron kívül csak lineáris méretű tárat használ, a Martínez–Silva-algoritmus futása során a megoldásokhoz nem vezető részeredmények mérete (az $\boldsymbol{M}$ mátrix sorainak száma) exponenciálisan nagyra nőhet.

3.1.3. LP-alapú leszámlálás

Contejean és Devie algoritmusa – csakúgy, mint az inhomogén egyenletrendszerre alkalmazható egyszerű rekurzív algoritmus (3.3.1. szakasz) – a keresési tér fabejárásán alapul, melyekben az l hosszú megoldások egy keresési fa l-edik szintjén találhatók, és ezek mindegyikéhez csak akkor jutunk el, ha a fa alsó $(l-1)$ szintjének nagy részét már bejártuk. Minden ilyen jellegű megoldási módszer eleve kudarcra van ítélve, ha a fa elágazási tényezője – azaz a vektorok száma – nagyon nagy. Ilyen esetekben, ha a háló nem elég ritka ahhoz, hogy a Martínez–Silva-algoritmust használjuk, egy olyan algoritmust kell találnunk, amely nem gráfbejáráson alapul.

A problémára általam javasolt algoritmus ezért egy egészértékű programozási [32] és rokon feladatoknál gyakran alkalmazott ötleten, az egészértékű feladat valós (illetve racionális) relaxációján alapul. Az alapelv – legyen szó akár homogén, akár inhomogén egyenletről – a következő: meghatározzuk a (3.1) egyenletrendszer racionális számok fölötti általános megoldását, mint egy partikuláris megoldás és a megfelelő homogén egyenletrendszer általános megoldásának összegét, majd a kapott általános megoldásvektor minden komponensére felírjuk a nemnegativitási feltételt. Az így kapott egyenlőtlenségrendszer megoldásai közül kiválasztjuk azokat, amelyekre a megoldásvektor komponensei egészek.

A megoldás pontosabb leírásához, illetve a helyesség bizonyításához szükség lesz az alábbi, egyszerűen igazolható állításra.

3.5. lemma *A* (3.2) *egyenlet megoldásainak létezik olyan* $\{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_k\}$ *bázisa, amelyre a vektorokból képzett* $(\mathbf{b}_1\ \mathbf{b}_2\ \cdots\ \mathbf{b}_k)$ *mátrixnak van egy* $k \times k$ *méretű, a főátlójában pozitív egész elemeket tartalmazó diagonális részmátrixa.*

BIZONYÍTÁS Feltéve, hogy már meghatároztunk egy racionális vektorokból álló bázist, a bázisvektorokból (mint sorvektorokból) álló mátrixot Gauss-eliminálva kaphatunk diagonális mátrixot. Ez az elemek racionalitásán nem változtat, csupa nulla sor pedig nem keletkezik, hiszen a mátrix teljes rangú. (A bázisvektorok függetlenek.) Ha az elimináció után a főátlóban negatív elemek is találhatók, úgy a megfelelő sorvektorokat -1-gyel szorozzuk. Végül a racionális bázisból megkapható az egész vektorokból álló bázis, ha szorzunk a nevezők legkisebb közös többszörösével. ∎

Az állítás szemléletesebben úgy is fogalmazható, hogy a homogén egyenletrendszer megoldásainak létezik olyan $\{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_k\}$ egész vektorokból álló bázisa, amelyre teljesül, hogy minden $\mathbf{b}_i$-nek van egy pozitív egész komponense úgy, hogy eközben az összes többi bázisvektor ugyanezen indexű komponense nulla.

Ha az inhomogén egyenlet egy partikuláris megoldása $\mathbf{p} = (p_1, p_2, \ldots, p_n)^T$, $(p_i \in \mathbb{Q})$, akkor az inhomogén egyenlet megoldásának általános alakja a 3.5. lemmában definiált

bázisvektorok segítségével így írható:

$$\boldsymbol{\nu} = \mathbf{p} + \mathbf{b}_1 x_1 + \cdots + \mathbf{b}_k x_k \quad (x_i \in \mathbb{R}).$$

Meg kell keresnünk azokat az $\mathbf{x} = (x_1, x_2, \ldots, x_k)$ vektorokat, amelyekre $\boldsymbol{\nu}$ minden komponense nemnegatív egész. Ehhez használhatjuk a $\mathbf{b}_i$-kre vonatkozó előbbi feltételünket. Ha ugyanis $b_{i_l} \in \mathbb{Z}^+$ jelöli a $\mathbf{b}_i$ vektornak az előbbi megjegyzésben említett pozitív egész komponensét (ez a vektor l-edik komponense), akkor a $\boldsymbol{\nu}$ vektor l-edik komponense $p_l + b_{i_l} x_i$ egész, tehát x_i szükségképpen egy $\frac{1}{b_{i_l}}(t - p_l)$, $(t \in \mathbb{Z})$ alakban írható racionális szám. Ezek után ha találunk alsó és felső korlátot $\mathbf{x}$ komponenseire, akkor az összes (immáron véges sok) esetet megvizsgálva a határok között megkereshetjük az összes megoldást. (Ez azt is mutatja, hogy a partikuláris megoldás keresésekor érdemes lehet a racionális megoldás helyett egész megoldást keresnünk. Ez a feladat inhomogén egyenlet esetén lényegesen számításigényesebb, de csak egyszer kell lefuttatnunk, a nagy számú lineáris programozási feladat megoldása előtt, homogén egyenletek esetén pedig semmilyen nehézséget nem jelent. Ezért cserébe a leszámlálandó x_i racionális együtthatók száma csökken.)

A határokat lineáris programozással kereshetjük meg. A $\boldsymbol{\nu} \geq \mathbf{0}$ egyenlőtlenségrendszer, mint korlátozó feltételek mellett keressük az egyes x_i-k minimumait és maximumait. (Ez $2k$ számú lineáris programozási feladat.) A legegyszerűbb (de általában korántsem a leghatékonyabb) megoldás, ha ezen határok között végigszaladunk minden lehetséges $(x_1, \ldots, x_k)$ kombináción, ellenőrizve, hogy csakugyan minden kombináció nemnegatív egészekből álló $\boldsymbol{\nu}$ vektort ad-e. Ez azonban nagyon időigényes lehet, ha a lineáris programozással kapott keresési tér (azaz a leszámlálandó vektorok száma) nagy.

Egy másik lehetőség, hogy kezdetben csupán az egyik – például az x_1 – változóra számítjuk ki a korlátokat, majd x_1 minden szóba jövő értékére külön-külön meghatározzuk az x_2-re adható alsó és felső korlátokat és így tovább. E módszer előnye, hogy így csak olyan szám n-eseket kapunk, amelyeket a megoldás általános alakjába helyettesítve nemnegatív megoldásokat kapunk. Mivel pedig a határok között csak a megfelelő alakú racionális számokat vizsgáljuk meg, az így kapott megoldások garantáltan egészek is – végső soron tehát nincs szükség semmiféle ellenőrzésre, csupán a tartománybeli szám n-esek visszahelyettesítésére, és a megoldások felsorolására. Látszólag nagy többletköltséget jelenthet, hogy az előbbinél lényegesen több lineáris programot kell megoldanunk – valójában ez a változat gyakran nagyságrendekkel gyorsabbnak bizonyul, ami azzal magyarázható, hogy a viszonylag gyors lineáris programozási rutin minden meghívásával exponenciálisan sok rossz (negatív számokat eredményező) megoldásjelöltet metszhetünk le a keresési térből.

A leszámlálást még hatékonyabban implementálhatjuk Land és Doig módszerét alkalmazva [16, 3.6. alfejezet]. E módszer lényege, hogy ha $x_1, \ldots, x_{i-1}$ rögzített, és az x_{i+1}-re adható alsó és felső korlátot mint x_i függvényét tekintjük, akkor e két függvény olyan lineáris szakaszokból áll, melyek meredeksége legfeljebb egyszer vált előjelet. Továbbá, ha x_i (bármilyen irányú) változtatásával x_{i+1} megengedett értékeinek halmaza üressé válik, akkor az is marad, ha x_i-t tovább változtatjuk ugyanabban az irányban.

Fontos megemlíteni, hogy az algoritmus mindhárom lényeges része, a partikuláris megoldás megkeresése, a homogén egyenlet általános megoldása és a lineáris programozási feladatok megoldása egyaránt tartalmaz numerikus buktatókat. Az algoritmust *Mathematicában* implementáltam, ennek felhasznált rutinjai mind futtathatók egész típusú számokkal, amivel kicsivel lassabb, de mindig pontos (egész, illetve racionális típusú) eredményeket adó eljárásokat kapunk. Ugyanakkor az eljárások dupla pontosságú valós számokként megadott egészekkel nehezebb feladatok esetén numerikus hibák miatt hibás megoldásokat adtak. A hibakeresés során úgy tapasztaltam, hogy ezek a hibák jellemzően a lineáris programok megoldása során keletkeztek.

Eddig nem foglalkoztunk azzal, mit tegyünk akkor, ha valamelyik x_i változóra nem kapunk felső korlátot. (Alsót nyilvánvalóan kapunk, ez az előbb felhasznált lemmából azonnal következik.) Ez akkor következik be, ha az egyenlet (racionális) megoldásának valamelyik komponense tetszőlegesen nagy lehet. Meggondolható, hogy ilyenkor a megfelelő komponens az egészek halmazán is tetszőlegesen nagy lehet. Ez a probléma tehát csak akkor léphet fel, ha végtelen sok megoldás van, melyek felsorolása amúgy is reménytelen. Ilyenkor a Contejean–Devie-algoritmushoz hasonlóan kereshetnénk a minimális megoldásokat, erre ez az algoritmus azonban közvetlenül nem használható, hiszen közvetlenül nem ellenőrizhető, hogy a megoldások minimálisak-e. (Az egyes x_i-k növelésével a $\boldsymbol{\nu}$ megoldásvektor különböző komponensei nőhetnek és csökkenhetnek is.) Azt azonban ezzel a módszerrel is megtehetjük, hogy felsoroljuk az összes (nem feltétlenül minimális) megoldást egy bizonyos korlátig: a megoldásokra adható hosszkorlát csupán egy további lineáris egyenlőtlenséget jelent a megoldandó lineáris programokban. Ha becsülni tudjuk a minimális megoldások maximális méretét, akkor egy ilyen becslésből származó korlátig felsorolva a megoldásokat, azok között minden minimális megoldás szerepel. Ebből a megoldáslistából a minimálisok kiválogatása már triviális feladat.

3.2. Felső becslések a minimális megoldások méretére

Domenjoud [7] cikkében olyan algoritmust javasolt a minimális megoldások kiszámítására, amely felső korlátot is ad azok méretére. Az algoritmus része az egyenletrendszer mátrixának minden $r \times r$-es minorának kiszámítása (ahol r a mátrix rangja), ami hatalmas számítási munka nagy egyenletrendszerek esetén; ilyenkor ez az algoritmus a megoldá-

sok számától és méretétől függetlenül alkalmazhatatlan. Később további, lineáris algebrai módszereken alapuló korlátok születtek a minimális megoldások méretére.

Pottier [24] cikke a legteljesebb összefoglalása az eddig ismert becsléseknek, amelyek Domenjoud algoritmusából, illetve egyéb (a megoldások meghatározását nem eredményező) lineáris algebrai megfontolásokkal kaphatók. Az eredményeket a következő tétel foglalja össze [7, 24].

3.6. tétel *Legyen a* (3.2) *egyenletrendszer mátrixa* $\mathbf{A}$*, ennek oszlopait jelölje* $\boldsymbol{\nu}_1, \ldots, \boldsymbol{\nu}_n$*, és legyen* $r = \operatorname{rank} \mathbf{A}$*. Legyen továbbá* D_r *az a legnagyobb szám, amely előáll* $\mathbf{A}$ *egyik* r*-edrendű minormátrixa determinánsának abszolút értékeként; hasonlóan a* D'_r *szám az* $\binom{1\cdots 1}{\mathbf{A}}$ *mátrix* $(r+1)$*-edrendű minorai determinánsának abszolút értékei közül a legnagyobb. Ekkor az egyenletrendszer minden* $\mathbf{m}$ *minimális megoldására fennállnak az alábbi egyenlőtlenségek:*

$$\|\mathbf{m}\|_1 \leq (1 + \max_i \|\boldsymbol{\nu}_i\|_1)^r, \tag{3.4}$$

$$\|\mathbf{m}\|_1 \leq (n-r)D'_r, \tag{3.5}$$

$$\|\mathbf{m}\|_\infty \leq (n-r)\left(\frac{\|\mathbf{A}\|}{r}\right)^r, \tag{3.6}$$

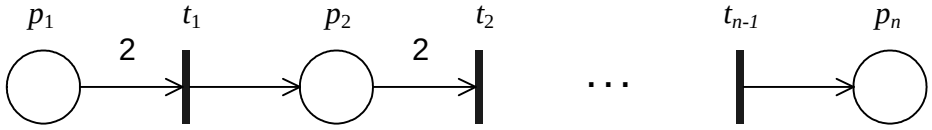
$$\|\mathbf{m}\|_\infty \leq (n-r)D_r. \tag{3.7}$$

A (3.7) becslés élesebb (3.6)-nál (valójában (3.6) becslés a (3.7) egyenlőtlenség közvetlen következménye), de előbbi inkább csak elvi jelentőségű, mert a (3.5) és (3.7) egyenlőtlenségek jobb oldalán álló kifejezések nem számíthatók ki hatékonyan, így nagyobb egyenletrendszerek esetén csak az első és a harmadik becslés használható. Láthatóan mindegyik becslés exponenciális, de (3.7) – legalábbis bizonyos mátrixosztályokon – éles, és (3.5) valamint (3.6) ennél gyakran nem sokkal rosszabb, tehát e becslések általában lényegesen nem javíthatók. Ezért az alkalmazások adta korlátok gyakran lényegesen kisebbek, mint a 3.6. tételben szereplő elvi korlátok.

Példa Tekintsük a 3.2. ábrán látható Petri-hálót. Láthatóan egyetlen minimális, illetve minimális tartójú P-invariánsa van, ez az $\mathbf{i} = (1, 2, \ldots, 2^{n-1})^T$ vektor. A Petri-háló szomszédossági mátrixa az $n \times (n-1)$-es

$$\boldsymbol{W} = \begin{pmatrix} -2 & 0 & \cdots & 0 \\ 1 & -2 & & \vdots \\ 0 & 1 & \ddots & 0 \\ \vdots & \vdots & \ddots & -2 \\ 0 & 0 & \cdots & 1 \end{pmatrix}$$

mátrix, így a $\boldsymbol{W}^{\mathrm{T}}\mathbf{x} = 0$ egyenlet egyetlen minimális megoldása $\mathbf{i}$. A 3.6. tételben most $r = \operatorname{rank} \boldsymbol{W} = n - 1$ és $D_r = 2^{n-1} = \|\mathbf{i}\|_\infty$, tehát a (3.7) egyenlőtlenség éles.



3.2. ábra. Petri-háló, amelyre a (3.7) becslés éles

3.3. Inhomogén egyenletek megoldása

Ebben az alfejezetben visszatérünk az eredeti, (3.1) inhomogén egyenletrendszer megoldásához. Az első szakasz egy olyan rekurzív algoritmust ismertet, amely csak speciális inhomogén egyenletek esetén alkalmazható, azonban rendkívül egyszerű és magától értetődő, ráadásul egyes feladatok esetén igen hatékony is. Ezután megvizsgáljuk, a homogén egyenletrendszerek esetén alkalmazott módszerek miként használhatók inhomogén egyenletrendszerek megoldására.

3.3.1. Naiv, rekurzív megoldás

Ha a (3.1) egyenletben szereplő vektorok egydimenziósak, akkor a klasszikus számpartícionálási feladathoz jutunk [1]. Ennek során egy pozitív egészt kell előállítanunk pozitív egészek összegeként az összes lehetséges módon, a sorrendre való tekintet nélkül. Ez tehát a (3.1) probléma azon speciális esete, amelyben $\mathbf{b} = n$ egy pozitív egész (az egydimenziós vektort a megfelelő skalárral azonosítva) a $\boldsymbol{\nu}_i$ vektorok pedig az $1, 2, \ldots, n$ számok. Ez a speciális eset egy rendkívül egyszerű rekurzív algoritmussal megoldható: az n szám partícióit úgy keressük, mint az olyan partíciókat, amelyekben a legnagyobb szám legfeljebb n; az így átalakított kétparaméteres feladatot pedig rekurzívan oldjuk meg. Az n szám m-nél nem nagyobb számokból álló partícióit úgy állítjuk elő, hogy megkeressük n minden olyan partícióját, amelyben a legnagyobb szám legfeljebb $(m - 1)$, továbbá $(n - m)$ minden partícióját, amelyben a legnagyobb szám legfeljebb m. (Utóbbiakhoz hozzávesszük m-et, az előbbiekhez nem.) A rekurzió lezárása triviális, amikor bármelyik paraméter értéke 1-re, vagy az alá csökken. Ezt az ötletet általánosíthatjuk vektorokra, amivel egy – nem minden esetben alkalmazható! – algoritmust kapunk. Ennek programvázlata a 3.4. algoritmus.

Az algoritmus minden particionáló vektort kétfelé ágazik aszerint, hogy az éppen sorra vett particionáló vektort még egyszer felhasználjuk-e a megoldásban. (Ez teljesen

3.4. algoritmus: Az inhomogén (3.1) egyenlet rekurzív megoldása

Bemenet: A $(\boldsymbol{\nu}_1, \boldsymbol{\nu}_2, \ldots, \boldsymbol{\nu}_n)$ particionáló vektorok és a $\mathbf{b}$ felbontandó vektor
Kimenet: A (3.1) egyenletet kielégítő $(\alpha_1, \alpha_2, \ldots, \alpha_n)$ vektorok, vagy hibaüzenet
Előfeltétel: `Folytatható`$((\boldsymbol{\nu}_1, \boldsymbol{\nu}_2, \ldots, \boldsymbol{\nu}_n), \mathbf{b})$

```
1 if n = 0 then
2     if b = 0 then
3         Egy megoldás van: egy nulla hosszúságú együtthatólista.
4     else
5         Nincs megoldás.
6 else
7     Particionáljuk b − ν1-et a (ν1, ..., νn) vektorokkal, a megoldások (ha vannak)
          első komponensét növeljük eggyel                    // Rekurzív hívás
8     Particionáljuk b-t a (ν2, ..., νn) vektorokkal, a megoldások elé (ha vannak)
          fűzzünk egy 0 komponenst.                           // Rekurzív hívás
9     A két rekurzív hívás eredményének uniója a megoldás.
```

analóg az egészek particionálására adott algoritmus működésével.) Hogy az eljárás biztosan véget érjen, minden vektorról meg kell tudnunk állapítani, hogy hányszor használhatjuk még a felbontás során, ellenkező esetben valamelyik particionáló vektornál az eljárás végtelen ciklusba kerül.[1] A szükséges ellenőrzést az algoritmus elején, a `Folytatható` függvény meghívásával végezzük. A `Folytatható`$((\boldsymbol{\nu}_1, \boldsymbol{\nu}_2, \ldots, \boldsymbol{\nu}_n), \mathbf{b})$ függvény értéke igaz, ha $\boldsymbol{\nu}_1$-nek van olyan pozitív (negatív) komponense, amelyre $\mathbf{b}$ megfelelő (azonos) indexű komponense szintén pozitív (negatív) és legalább akkora abszolút értékű, mint $\boldsymbol{\nu}_1$ megfelelő komponense, továbbá ha $\boldsymbol{\nu}_2, \ldots, \boldsymbol{\nu}_n$ ugyanezen komponensei mind nemnegatívak (nempozitívak). Ez az első pillantásra körülményes és nehezen teljesíthető feltétel csupán annak biztosítására szolgál, hogy a $\boldsymbol{\nu}_1$ vektor együtthatója korlátos maradjon, és az algoritmus ne kerülhessen végtelen ciklusba azért, mert az első rekurzív hívás végtelen sokszor hívódik meg.

Ha a $\boldsymbol{\nu}$ vektornak nincs olyan komponense, amelyik alapján biztosra vehető, hogy a vektort véges sokszor használjuk fel $\mathbf{b}$ particionálásában, akkor az algoritmusnak hibaüzenettel le kell állnia. Ennek a módosításnak az az ára, hogy az algoritmus nem használható minden esetben, így például akkor sem, ha minden $1 \leq i \leq d$-hez (d a komponensek száma) létezik olyan particionáló vektor, amelynek i-edik komponense negatív, és egy másik, amelynek ugyanezen komponense pozitív. Ez bizonyos alkalmazásoknál nem okoz gondot (ilyen például az adott anyagfajták között végbemenő elemi reakciók előállításának feladata), de általában súlyos megszorítást jelent.

[1] Az egészek particionálásánál ilyen probléma természetesen nem lép fel, a gondot a vektorokban megengedett különböző előjelű komponensek okozzák.

Vegyük észre, hogy problémás particionáló vektorok esetében olykor pusztán a vektorok permutálásával elérhetjük, hogy az eljárás véges lépésben véget érjen. Legyen például $n = d = 2, \boldsymbol{v}_1 = (1,0)^T, \boldsymbol{v}_2 = (-1,1)^T$. Ebben a sorrendben semmilyen $\mathbf{b}$ vektor esetén nem tudjuk közvetlenül alkalmazni az algoritmust, míg a vektorokat megcserélve minden jobb oldal esetén megkapjuk az összes megoldást. A végesség szükséges és elégséges feltétele, hogy a vektorok a következő definíció szerinti „megfelelő" sorrendben álljanak.

A $\{\boldsymbol{v}_i\}_{i=1}^n$ sorozatot nevezzük **jólrendezett**nek[2], ha minden vektornak van legalább egy pozitív (vagy negatív) komponense, és ha az utána következő vektorok megfelelő (azonos indexű) komponensei mind nemnegatívak (ill. nempozitívak). Az egyes vektorok jólrendezettségét biztosító komponenseit nevezzük a vektorok **jó komponens**einek. A $\{\boldsymbol{v}_i\}_{i=1}^n$ vektorok **jólrendezés**én a sorozat egy olyan permutációját értjük, amellyel a sorozat jólrendezetté tehető. A következő segédtétel a kulcsa egy jólrendező algoritmusnak.

3.7. lemma

1. *Ha minden* k*-ra a sorozat tagjainak* k*-adik komponensei között pozitív és negatív is található, vagy ha a sorozat valamelyik tagja nullvektor, akkor a sorozat nem jólrendezhető.*

2. *Legyen* k *egy olyan index, amire minden vektor* k*-adik komponense nemnegatív (ill. nempozitív.) A sorozat akkor és csak akkor jólrendezhető, ha elhagyva mindazokat a vektorokat, amelyek* k*-adik komponense nem nulla, egy jólrendezhető sorozatot kapunk. (Az elem nélküli sorozatot jólrendezettnek tekintjük.)*

BIZONYÍTÁS

1. Az első helyre kerülő vektornak, illetve a nullvektornak semmilyen permutáció esetén nincs jó komponense.

2. Az „akkor" irány bizonyítása: ha a redukált sorozat jólrendezhető, rendezzük azt, majd tegyük vissza az elhagyott vektorokat a sorozat elejére. Ez az eredeti sorozat egy jólrendezése, mert az el nem hagyott vektorok rendezésének jóságán nem változtatnak az eléjük tett vektorok, az elhagyott (majd visszatett) vektoroknak pedig van jó komponensük – az a nemnulla komponens, amelyik alapján kiválasztottuk őket. A másik irányú igazoláshoz indirekten tegyük fel, hogy az eredeti sorozat jólrendezhető, míg a redukált sorozat nem. Ez azt jelenti, hogy van olyan vektor, aminek egy jó komponense megszűnik jónak lenni azáltal, hogy más vektorokat törlünk a sorozatból. Ez nyilvánvalóan lehetetlen. ∎

A segédtételből egy polinomidejű jólrendező algoritmus bontakozik ki: ha a vektoraink nem jólrendezettek, ellenőrizzük, hogy teljesül-e az előbbi lemma első pontjának

[2] Ugyan a *jólrendezett halmaz* fogalma a matematikában már foglalt, ebben a szövegkörnyezetben – vektorok hagyományos értelemben vett sorrendezéséről lévén szó – a névütközés remélhetőleg nem okoz zavart.

feltétele, s ha nem, akkor a második pont alapján próbáljuk rekurzívan jólrendezni a sorozat elemeit.

A naiv algoritmus a vektorok jólrendezésével kiegészítve is csak akkor lehet működőképes, ha a megoldások száma véges (olyankor sem mindig), a minimális megoldások előállítására nem alkalmazható.

3.3.2. Visszavezetés homogén egyenletre

Az előző alfejezetben a homogén egyenlet megoldására ismertetett módszerek (esetleg kisebb módosítással) alkalmazhatók inhomogén egyenletek esetén is.

A Contejean–Devie-algoritmus (3.1.1. szakasz) esetén a $\mathbf{v}_i$ vektorokhoz hozzávesszük a jobb oldal ellentettjét, azaz legyen $\mathbf{v}_{n+1} := -\mathbf{b}$, majd az algoritmust úgy alkalmazzuk, hogy az első (inicializáló) lépésben a kiindulási vektor a nullvektor helyett legyen az $(n+1)$ dimenziós $(\underbrace{0,\ldots,0}_{n},1)$ vektor, majd folytassuk az algoritmust a megadott módon, de a vektorok utolsó komponensét soha nem növelve. Végül az M halmaz minden elemének utolsó komponensét hagyjuk el.

A Martínez–Silva-algoritmus (3.1.2. szakasz) esetében is hasonlóan járhatunk el. Mivel itt az egyenletrendszer mátrixának transzponáltjával számolunk, az egyenletrendszer jobb oldalán álló vektor ellentettjét a mátrix transzponáltjának sorvektoraihoz vehetjük hozzá. Ezúttal is ügyelnünk kell arra, hogy az új vektor 1 együtthatóval szerepeljen a megoldásokban. Ismét felhasználjuk, hogy az algoritmus futása során az összevont állapotok folyamatosan bővülnek: amint egy ilyen állapotban az új vektornak megfelelő hely egynél nagyobb együtthatóval szerepel, azt elhagyhatjuk, így egyetlen megoldás sem vész el.

Az LP-alapú leszámlálás (3.1.3. szakasz) esetében semmilyen módosításra nincs szükség, az algoritmus inhomogén egyenletekre is alkalmazható. Ha az inhomogén egyenlet megoldásainak száma véges – amit onnan tudhatunk, hogy a megoldás általános alakjában szereplő paraméterekre lineáris programozással kapunk felső korlátot – akkor a 3.2. alfejezetben megismert korlátok alkalmazása sem szükséges.

3.4. Az algoritmusok összehasonlítása

Ebben az alfejezetben még egyszer röviden összefoglaljuk és összehasonlítjuk az eddigiekben bemutatott megoldó algoritmusok főbb jellemzőit, kifejezetten elvi megfontolásokkal; alkalmazási példákat és számítási eredményeket a 3.7. alfejezet és a 6. fejezet ismertet.

A megoldható feladatok

Az algoritmusok között a legszembetűnőbb különbség az, és ez minden további összehasonlításra rányomja bélyegét, hogy különböző feladatok megoldására használhatók. A legáltalánosabban Contejean és Devie, illetve Martínez és Silva algoritmusai használhatók, ezek tetszőleges egyenlet összes minimális, illetve minimális tartójú megoldását előállítják. (Mindkét esetben azzal a kiegészítéssel, amit a 3.3.2. alfejezetben láttunk.) A Martínez–Silva-algoritmus használatát a kémiai feladatokban alapvetően megnehezíti, hogy a minimális tartójú megoldásoknál kémiailag jobban értelmezhető minimális felbontásokat nehezen kaphatjuk meg a segítségével, a minimális tartójú megoldások generátor tulajdonságát kimondó 3.2. lemma inhomogén egyenletekre nem alkalmazható. Az elemi reakciók előállításánál is csak az teljesül, hogy minden megoldás minimális, ezek azonban nem szükségképpen minimális tartójúak.

Az LP-alapú leszámlálás minden megoldást megad, ha véges sok van, azonban nem használható, ha végtelen sok megoldás van – nem módosítható például úgy, hogy csak a minimális megoldásokat adja meg. Ha felső korlátunk van a keresett megoldások hosszára, akkor ez az algoritmus is használható. A kémiai reakciók felbontásának feladatánál ez a különbség nem nagyon jelentős. Az elemi reakciólépések előállításakor megoldandó egyenletrendszereknek véges sok megoldása van, ilyenkor az algoritmus közvetlenül használható. A felbontások előállításánál pedig a minimális felbontások helyett korlátos számú lépésből álló felbontásokat kereshetünk a segítségével (melyek közül azután a minimálisokat is könnyűszerrel kiválogathatjuk).

A naiv, rekurzív algoritmus a vektorok jólrendezésével kiegészítve is csak speciális inhomogén egyenletrendszerek esetén alkalmazható; néha akkor sem használható, ha a megoldások száma véges. Speciálisan, a felbontások előállítására ez az algoritmus szinte soha nem alkalmazható, az elemi reakciólépések meghatározására azonban mindig.

Invariánskeresésre (informatikai rendszerek modellezésekor) használva az algoritmusokat csak homogén egyenletek megoldása a feladatunk. Ezért ilyenkor az LP-alapú leszámlálás csak hosszkorlátos invariánsok keresésére alkalmazható. Speciális esetként a $\leq$ részbenrendezésre nézve minimális invariánsok méretére a 3.6. tételben adott korlátok alkalmazhatók (3.2. alfejezet).

Hatékonyság

Korábban igazoltuk, hogy a (3.1) egyenlet megoldhatóságának eldöntése **NP**-nehéz, a megoldások felsorolása pedig – már a megoldások nagy száma miatt is – exponenciális tárkomplexitású feladat, így a bonyolultságelméletben hagyományos értelemben hatékony – polinomidejű – algoritmus a megoldásra nem várható. Meghatározott feladatosztályokra azonban a gyakorlatban gyorsan működő algoritmusokat kereshetünk.

A Contejean–Devie-algoritmus ismertetésénél említettük, hogy rendkívül tárhatékonyan is implementálható (eltekintve persze a memóriában tárolt megoldások tárigényétől), azonban a futási ideje exponenciálisan nő a megoldások nagyságának növekedtével. Erre az algoritmusra tehát általában csak addig számíthatunk, amíg a megoldandó egyenletrendszer megoldásai között kis méretűek is vannak. Különösen nagy méretű feladatok esetén szükséges lehet, hogy a megoldásokat futás közben háttértárra mentsük, hogy ne foglalják le az összes memóriát. Ez a Contejean–Devie-algoritmus esetén nem kivitelezhető, hiszen a részmegoldásokat lépésenként összevetjük a megtalált megoldásokkal.

Martínez és Silva algoritmusa esetén nem ismert tárhatékony implementáció, a megoldásokon kívül tárolandó részeredmények maximális mérete mind a megoldások számának, mind a probléma méretének exponenciális függvénye. Jellemzően csak ritka Petri-hálók invariánsainak keresésére használható.

A naiv algoritmus várhatóan mindaddig gyors, amíg a keresés során viszonylag kevésszer ágazik el a futása, azaz folytatódik két rekurzív hívással. Különösen jellemző ez az olyan feladatokra, amelyeknél a felbontandó vektor nem sokkal nagyobb a particionáló vektoroknál. Ellenkező esetben azonban nagyon nagy válaszidőre számíthatunk az alkalmazásakor.

Az LP-alapú leszámlálás hatékonysága előre nehezen megjósolható. A keresési fázis előtti szakasz – az általános megoldás meghatározása – igen gyors. Ugyanakkor a nagy számú lineáris program megoldásának szükségessége első látásra elrettentőnek tűnik. Nem világos, hogy a különböző változatok közül a bemenet ismeretében hogyan válasszuk ki a legmegfelelőbbet.

3.5. Gyorsítás előfeldolgozással

A megoldások előállítása általában rendkívül számításigényes feladat az egyenlethez legjobban illeszkedő algoritmussal is, ezért érdemes olyan kiegészítő eljárásokat keresni, amelyekkel a megoldandó egyenlet egyszerűbbé alakítható.

3.5.1. Minden megoldásban szereplő vektorok keresése

Mind kémiai, mind informatikai alkalmazásból származó Petri-hálókban gyakoriak az olyan helyek/átmenetek, amelyek minden invariánsban szerepelnek. Ez adja az ötletet, hogy általában is keressünk a (3.1) egyenlet ν_i vektorai között olyanokat, amelyek minden megoldásban szerepelnek. Ezek a vektorok (a minimális α_i együtthatójukkal súlyozva) kivonhatók az egyenlet mindkét oldalából, amivel a megoldandó egyenlet-

rendszer megoldásainak mérete csökkenthető – különösen kedvezve ezzel a naiv és a Contejean–Devie-algoritmusnak. Ilyen vektorokat lineáris programozással kereshetünk.

A $\boldsymbol{\nu}_k$ ($k = 1, 2, \ldots, n$) vektor vizsgálatához a következő relaxált ($\alpha_i \in \mathbb{R}$) lineáris programot kell megoldani:

$$\min \alpha_k; \quad \sum_i \alpha_i \boldsymbol{\nu}_i = \mathbf{b}, \ \forall i: \ \alpha_i \geq 0 \tag{3.8}$$

A kapott minimum értéke (illetve annak felső egészrésze is) alsó becslés a vizsgált vektor α_i együtthatójára minden megoldásban. Mivel csupán változónként egyetlen lineáris programot kell megoldani, az eljárás az egyenlet megoldásához képest igen gyors.

3.5.2. A megoldásokban nem szereplő vektorok keresése

Az előző szakaszban alkalmazott ötlet úgy is felhasználható, hogy a (3.8) lineáris programban minimalizálás helyett maximalizálunk. Ha α_k maximuma létezik, és kisebb egynél, akkor a $\boldsymbol{\nu}_k$ vektor együtthatója minden megoldásban nulla. Már ez az egyszerű meggondolás is használható az egyenletrendszer méretének csökkentésére, de ennél hatékonyabb és szellemesebb az eredetileg a [15] cikkből származó, illetve általam kiegészített [21] algoritmus, amely mintegy „mellékesen" meghatározza a megoldásokban mindig zérus együtthatóval szereplő vektorokat is. Ennek leírása a 3.6. alfejezetben olvasható.

3.5.3. Indexezés

Ha a Petri-háló egy rögzített kezdeti jelöléssel adott, akkor csak a kezdeti jelölésből tüzelhető invariánsokat keressük, azaz olyan invariánsokat, amelyekhez létezik olyan tüzelési sorozat, amelyben minden átmenet annyiszor szerepel, amekkora a hozzá tartozó komponens értéke az invariánsban, és amely sorozatban minden átmenet minden tüzelése előtt engedélyezve van.

Ez diszkrét állapottér esetén egy exponenciális bonyolultságú keresési feladat, ezért a keresés során hasznosak az ún. részleges döntési technikák, azaz gyorsan ellenőrizhető szükséges feltételek keresése és ellenőrzése. Folytonos állapotterű és idejű Petri-hálós modellekben (4.1. alfejezet) egyszerűbb a helyzet: megadható egy olyan lineáris lépésszámú algoritmus, amely a megoldásról eldönti, hogy adott kezdeti jelölés esetén realizálható megoldás-e, sőt, a jelölés $t = 0$ közeli viselkedését is jellemzi. Ez az algoritmus a Volpert-indexezés [33], amellyel részletesen a 4.1.1. szakasz foglalkozik. Ezt az eljárást, illetve egy egyszerűbb változatát többször újra felfedezték (köztük e dolgozat szerzője is [21]), egyszerűsített változatát – amely diszkrét állapottérre is adaptálható – mutatja be pl. [13].

Diszkrét állapottér esetén Volpert indexezési technikája közvetlenül nem alkalmazható, azonban – már egy egyszerűsített változata is – szükséges (de nem elégséges) feltételt ad a megoldások tüzelhetőségére. Az eljárás lényege, hogy nem vizsgáljuk a tokenek pontos számát az egyes helyeken, csupán azt figyeljük, hogy kerülhet-e a helyekre token. Ha kerülhet, feltételezzük, hogy a továbbiakban mindig meg is található rajta a szükséges számú token. Kezdetben tehát a pozitív tokenszámú helyek vannak jelölve, majd a Petri-hálót az engedélyezett átmeneteken keresztül „elárasztjuk": ha egy jelöletlen átmenet minden bemenő helye jelölt, akkor az átmenetet és minden kimenetét megjelöljük. Az eljárás végén jelöletlenül maradt átmenetek és helyek törölhetők a Petri-hálóból, hiszen a jelöletlen átmenetek semmilyen tüzelési sorozatban nem tüzelhetnek, a jelöletlen helyekre pedig soha nem kerülhet token.

Az indexezés algoritmusának eredeti, általános változata – programvázlattal együtt – a 4.1.1. szakaszban olvasható. Az előbbiek alapján most úgy alkalmazhatjuk ezt az algoritmust, hogy nem teszünk különbséget a különböző véges indexek között, csupán azt kell nyilvántartanunk, hogy mely anyagok és reakciók kapnak véges indexet.

3.6. A megoldások nem szisztematikus előállítása

Ha az összes megoldás, vagy az összes minimális tartójú megoldás előállítása – például azok nagy száma miatt – reménytelen, csak a megoldások egy részhalmazát állíthatjuk elő. Ilyenkor nem feltétlenül szükséges a megoldásokat szisztematikusan keresni. Publikációjukban [15] szerzői erre a problémára egy lineáris programozáson alapuló megoldást javasoltak, ennek (részben [21]-ben közzétett) kiegészítését ismerteti ez a szakasz.

Az algoritmus lényege, hogy valahányszor találunk egy megoldást, az abban szereplő vektorokat „megjelöljük", és ezek után csak olyan megoldásokat keresünk, amelyekben legalább az egyik felhasznált vektor még nincs jelölve. (Kezdetben egy vektor sincs jelölve.) A formális programvázlatot a 3.5. algoritmus mutatja.

Egy, a 3. lépésben keresett megoldásvektort a következő lineáris programozási feladat megoldásaként kaphatunk:

$$\min \mathbf{c}^T\mathbf{x}; \quad \sum_{i=1}^{n} x_i\boldsymbol{\nu}_i = \mathbf{b},\ \mathbf{c}^T\mathbf{x} \geq 1,\ \mathbf{x} \geq \mathbf{0}^n, \tag{3.9}$$

ahol $\mathbf{c} \in \{0,1\}^n$ az M halmaz karakterisztikus vektorának komplemense, azaz a $\mathbf{c}$ vektor i-edik komponense 1, ha $i \notin M$, és 0 egyébként. A célfüggvény választására hamarosan visszatérünk.

3.5. algoritmus: A (3.1) egyenlet részleges, nem szisztematikus megoldása

Bemenet: $\boldsymbol{\nu}_1, \boldsymbol{\nu}_2, \ldots, \boldsymbol{\nu}_n$ és $\mathbf{b}$

Kimenet: A (3.1) egyenletet kielégítő $(\alpha_1, \alpha_2, \ldots, \alpha_n)$ vektorok S halmaza.

1 $M \leftarrow \emptyset$; `// a jelölt vektorok indexei`

2 $S \leftarrow \emptyset$; `// a megoldáshalmaz`

3 Legyen $\boldsymbol{x} = (x_1, \ldots, x_n)^T$ a (3.1) egyenlet *racionális* megoldása ($x_i \in \mathbb{Q}_0^+$), melyre $\{i \mid x_i > 0\} \not\subset M$. Ha nincs ilyen megoldás, álljunk meg.

4 $M \leftarrow M \cup \{i \mid x_i > 0\}$; `// vektorok megjelölése`

5 $S \leftarrow S \cup \{\boldsymbol{x}\}$; `// új megoldás`

6 Ugorjunk a 3. lépésre.

Az algoritmus véges sok lépésben véget ér, mert minden iterációban, amikor nem áll meg, legalább eggyel növekszik az M halmazban található indexek (azaz a megjelölt vektorok) száma. Azok a vektorok, amelyek jelöletlenek maradnak, egyetlen megoldásban sem szerepelhetnek, így az algoritmus a 3.5.2. szakaszban említett módon előfeldolgozási lépésként is alkalmazható az egyenletrendszer felesleges oszlopvektorainak szűrésére, azaz a változók számának csökkentésére.

Homogén egyenletek esetén – így például P- és T-invariánsok keresésekor – a nemnegatív racionális megoldást a komponensek nevezőinek legkisebb közös többszörösével szorozva nemnegatív egész megoldást kapunk, így az eljárás számos megoldást is szolgáltat az akkor (3.2)-be átmenő feladatra. Megoldásonként csupán egyetlen lineáris programot kell megoldanunk, ami lényegesen hatékonyabb az LP-alapú leszámlálásnál, azonban semmit nem mondhatunk arról, hogy hány és milyen megoldásokat találunk meg, illetve melyeket nem. Inhomogén egyenletrendszereknél a megoldás skálázásával nem az eredeti egyenlet megoldását kapjuk, kémiai reakciók felbontásából származó egyenletek esetén azonban ezek a megoldások is kémiailag értelmesek. (A reakció többszöröse ugyanaz a reakció.) Érdekes módon az algoritmus a mérések során gyakran az inhomogén egyenletekre is sok egész megoldást adott.

A 3. lépésben valójában bármilyen célfüggvényt alkalmazhatunk, amelyre a keresett minimum létezik; akár egy konstanst is „minimalizálhatunk” a (3.9) lineáris program célfüggvényében a $\mathbf{c}^T = \mathbf{0}^T$ választással élve. A feltételeket azonban változatlanul kell hagynunk, hogy biztosan minden megoldás tartalmazzon jelöletlen vektort, ellenkező esetben végtelen sokszor visszakaphatnánk ugyanazt a megoldást. A (3.9)-béli választást az a heurisztikus érvelés indokolja, hogy ezzel minden lépésben kevés jelöletlen vektort használunk fel és jelölünk meg. Így remélhetjük, hogy az eljárásunk sok iteráción keresztül fut, és így sok megoldást ad. (Természetesen erre semmilyen elvi garancia nincsen.) Az eljárás előfeldolgozási lépésként történő felhasználásánál ez nem célszerű megoldás, akkor ugyanis épp ellenkezőleg az a célunk, hogy minél kevesebb iterációval jelöljük meg

az összes megjelölhető vektort. Ilyenkor a célfüggvény lehet például a jelölt vektorok karakterisztikus vektorának és $\mathbf{x}$-nek a skaláris szorzata. A minimum erre a célfüggvényre is létezik, de természetesen most sincsen elvi garancia arra, hogy ez a „mohó" stratégia valóban minimális számú lépéssel célhoz ér.

Különböző célfüggvényekkel különböző megoldáshalmazokat kaphatunk, ami a megoldásokban nem szereplő vektorok kiválasztása szempontjából érdektelen, a megoldások gyors, nem szisztematikus keresésekor azonban így több megoldást kaphatunk.

Könnyen meggondolható, hogy ha a célfüggvény az $\mathbf{x}$ vektor és egy tisztán pozitív vektor skaláris szorzata, akkor minden kapott megoldás minimális. Legyen ugyanis $\mathbf{x}$ egy nem minimális megoldás, amelyhez tehát létezik olyan $\mathbf{x}^*$ megoldás, amelyre $\mathbf{x}^* \leq \mathbf{x}$ teljesül. Akkor azonban $\mathbf{c}^T > \mathbf{0}^T$ miatt $\mathbf{c}^T\mathbf{x}^* < \mathbf{c}^T\mathbf{x}$ is fennáll, tehát abban az iterációban, amikor $\mathbf{x}$-et kaptuk megoldásként, hibásan oldottuk meg a (3.9) lineáris programozási feladatot.

E módszer legfőbb hátránya, hogy nem tehető szisztematikussá: a kapott megoldásoknak semmilyen „struktúrája" nincsen, ezen kívül semmilyen garanciánk nincs arra, hogy egy újabb célfüggvénnyel újabb megoldásokat találhatunk.

3.7. Alkalmazási példák

Ebben a szakaszban olyan reakciókinetikai példák olvashatók, amelyek vizsgálatára az ismertetett módszerek alkalmazhatók, és amelyekkel így az algoritmusok összehasonlíthatók. A futási eredmények az algoritmusok implementációját ismertető fejezet után, a 6. fejezetben találhatók.

A reakciókinetikai vizsgálatok során a Petri-hálós modell a kémiai folyamatban potenciálisan végbemenő elemi reakciókat adja meg (2.1. táblázat, 10. oldal). Az átmenetek felelnek meg a reakcióknak, a helyek az anyagfajtáknak. Az alapfeladat a *bruttó reakciók felbontása*, amelynek során először is magát a Petri-hálót kell előállítanunk, azaz meg kell határoznunk az adott anyagfajták között lehetséges elemi reakciókat.

Ha adva vannak a reakcióban résztvevő anyagfajták, az atom- és töltésmegmaradásnak eleget tevő elemi reakciók előállítása inhomogén lineáris diofantoszi egyenletrendszerek természetes számok felett megoldására vezethető vissza.

Permanganát–oxálsav reakció [15]. Ezt a reakciót több mint 100 éve vizsgálják, a folyamat pontos leírása azonban mind a mai napig nem ismert. A reakcióban 19 anyagfajta vehet részt, amelyek összesen négy elemet: mangánt, hidrogént, szenet és oxigént tartalmaznak. A felbontandó reakció:

$$2MnO_4^- + 6H^+ + 5H_2C_2O_4 \rightarrow 2Mn^{2+} + 8H_2O + 10CO_2 \,.$$

Az anyagok adatai a 3.1. táblázatban láthatók, az utolsó oszlop az anyagok töltését mutatja. Ahogy az a 2.4. alfejezetben olvasható, az összes elemi reakciót megkapjuk, ha minden anyag vektorát, mindegyik kétszeresét, végül minden vektorpár összegét az összes lehetséges módon particionáljuk a többi segítségével. Ezúttal ez összesen 209 egyenletrendszer megoldását jelenti.

	C	H	Mn	O	t
$H_2C_2O_4$	2	2	0	4	0
$HC_2O_4^-$	2	1	0	4	−1
H^+	0	1	0	0	1
$C_2O_4^{2-}$	2	0	0	4	−2
Mn^{2+}	0	0	1	0	2
MnC_2O_4	2	0	1	4	0
MnO_4^-	0	0	1	4	−1
MnO_2	0	0	1	2	0
Mn^{3+}	0	0	1	0	3
CO_2	1	0	0	2	0
H_2O	0	2	0	1	0
$[MnO_2, H_2C_2O_4]$	2	2	1	6	0
CO_2^-	1	0	0	2	−1
$[Mn(C_2O_4)]^+$	2	0	1	4	1
$[Mn(C_2O_4)_2]^-$	4	0	1	8	−1
$[MnC_2O_4, MnO_4^-, H^+]$	2	1	2	8	0
$[MnC_2O_4^{2+}, MnO_3^-]^+$	2	0	2	7	1
$[MnC_2O_4^{2+}, MnO_3^-, H^+]^{2+}$	2	1	2	7	2
$[H^+, MnO_2, H_2C_2O_4]^+$	2	3	1	6	1

3.1. táblázat. A permanganát–oxálsav reakcióban résztvevő anyagok listája (atommátrix)

Levegőkémiai folyamatok. Egy, a University of Leeds Üzemanyag és Energia Tanszékén folyó levegőkémiai kutatás számára készítettük el a 3.2. táblázatban felsorolt anyagok között végbemenő összes olyan elemi reakció listáját, amelyben szerepel ként tartalmazó anyag. (Ezek a táblázat alsó, a vonal alatti részén találhatók.)

Most tehát csak az elemi reakciók egy részét kell előállítanunk; a feltételekből 1668, egyenként 78 illetve 79 változós egyenletrendszer írható fel (attól függően, hogy egy vagy két anyagfajta van-e a reakció bal oldalán), a megoldáshoz itt is az előbb említett algoritmusokat használtuk. Mivel azonban a megoldások négyszázezret meghaladó száma kezelhetetlenül nagynak bizonyult a további vizsgálódások szempontjából, az összes elemi reakció közül ki kellett szűrnünk azokat, amelyeknek háromnál több terméke van. A részleteket lásd a 6.1. alfejezetben.

H_2	CH_4	C_2H_2	C_2H_4	C_2H_6	C_3H_4	C_3H_6	C_4H_2
O_2	H_2O	H_2O_2	CO	CO_2	CH_2O	CH_2CO	C
H	CH	CH_2	$CH_2(S)$	CH_3	C_2H	C_2H_3	C_2H_5
C_3H_2	H_2CCCH	H_2CCCCH	O	OH	HO_2	HCO	CH_3O
CH_2OH	HCCO	CH_2HCO	N_2	CN	HCN	N	NH
NO	HNO	NH_2	H_2NO	NCO	N_2O	NO_2	N_2H_2
HOCN	H_2CN	NNH	NH_3	N_2H_3	C_2N_2	HNCO	NO_3
S	SH	H_2S	SO	SO_2	SO_3	HSO_2	HOSO
$HOSO_2$	SN	S_2	CS	COS	HSNO	HSO	HOS
HSOH	H_2SO	HOSHO	HS_2	H_2S_2	CS_2	HSOO	H_2SO_4

3.2. táblázat. A levegőkémiai feladatban szereplő anyagok listája

Az oxalát–perszulfát–ezüst oszcillátor [5]. Ebben az [5] publikációból származó példában tizenhat anyagfajta található, amelyek ötféle atomból állnak. (Lásd a 3.3. táblázatot.) Ugyanúgy járhatunk el, mint az előző példákban, az elemi reakciólépések előállításához felírandó 152 egyenletrendszernek összesen 89 megoldása van. Ez a példa is mutatja, hogy már egészen kis méretű feladatok esetén is lényegesen több elemi reakciót állíthatunk elé módszeresen, mint amennyit „kézzel” felírhatnánk.

Ag^+	Ag^{2+}	H^+	SO_4^-
SO_4^{2-}	$S_2O_8^{2-}$	$C_2O_4^{2-}$	$Ag(C_2O_4)^-$
OH	H_2O	CO_2^-	O_2
HO_2	H_2O_2	$O_2CO_2^-$	CO_2

3.3. táblázat. Az oxalát–perszulfát–ezüst oszcillátor anyagfajtái

4. fejezet

Kémiai reakciók determinisztikus és sztochasztikus modellje

Amint azt a 2.4. alfejezetben említettük, a Petri-hálók segítségével összetett kémiai reakciók is ábrázolhatók. Az előző fejezetben az ehhez a formalizmushoz szorosan kötődő invariánsok alkalmazását és a megkeresésükre szolgáló algoritmusokat mutattuk be. Ebben a fejezetben azt vizsgáljuk, miként lehet a reakciók időbeli viselkedését leírni. Bemutatok két modellt a kémiai reakciók (illetve általában a Petri-hálók) dinamikájának leírására: egy folytonos állapotterű, determinisztikus, és egy diszkrét állapotterű, sztochasztikus modellt. Végül rámutatok néhány érdekes kapcsolatra a két modell között, Kurtz [17] publikációja és [3, 3. fejezet] alapján, helyenként az idézetteknél egyszerűbb levezetésekkel. Ez utóbbi eredmények eredendően kifejezetten kémiai reakciókra vonatkoznak, azonban teljes általánosságban, módosítás nélkül alkalmazhatók tetszőleges Petri-hálókkal leírt rendszerekre. Érdekes nyitott kérdés, hogy nem kémiai rendszerek esetén a folytonos állapotterű modellnek milyen jelentést tulajdoníthatunk, azaz például hogyan interpretálhatóak és hasznosíthatóak a fejezetben bemutatott eredmények informatikai modellezésben.

Az elsődleges nehézséget az jelenti, hogy a Petri-hálók eredendően diszkrét idejű, diszkrét állapotterű, nemdeterminisztikus automaták. A kémiai reakciók ezzel szemben folytonos idejű folyamatok, az állapottér jellege pedig attól függ, hogy a résztvevő anyagokra mint (diszkrét darabszámú) molekulák halmazára tekintünk-e, vagy mint valós mennyiségű (koncentrációjú) anyagokra. A reakciók esetében a nemdeterminisztikus működés sem értelmezhető, csak determinisztikus és sztochasztikus modellekről beszélhetünk. Ebben a fejezetben a vizsgált alkalmazásra koncentrálva a (P, T, w) Petri-háló helyett gyakran (M, R, α, β) **formális kémiai mechanizmus**ról, vagy másképpen **összetett kémiai reakció**ról fogunk beszélni, amely lényegében ekvivalens egy jelölés nélküli Petri-

hálóval. A (formális) anyagfajták m elemű M, illetve reakciólépések r elemű R halmaza felel meg a helyeknek, illetve az átmeneteknek, az $\alpha \in \mathbb{N}_0^{m\times r}$ mátrix tartalmazza a helyektől az átmenetek felé mutató élek súlyát (ez mutatja, mennyi anyag fogy a reakciólépésekben), míg a $\beta \in \mathbb{N}_0^{m\times r}$ mátrix az átmenetektől a helyek felé menő élek súlyaiból áll (ez a termelődő anyagok mennyisége). Az ezekből kapható $\gamma := \beta - \alpha$ mátrix pedig azonos a korábbiakban $\boldsymbol{W}$-vel jelölt szomszédossági mátrixszal.

4.1. A folytonos idejű, folytonos állapotterű, determinisztikus (CCD) modell

Ebben a modellben az anyagfajták mennyiségét a koncentrációk (nemnegatív valós) vektorával írjuk le, lényegében tehát folytonos jelöléssel dolgozunk. A reakciólépések folyamatosan, párhuzamosan mennek végbe (a jelölés tehát folytonos időben változik, a reakciólépések bekövetkezése pedig determinisztikus), a koncentrációváltozások sebessége a reaktánsok koncentrációjának függvénye. A formális kémiai mechanizmus **folytonos idejű, folytonos állapotterű, determinisztikus** (vagy **CCD**) **modell**jének a következő közönséges, elsőrendű, polinomiális differenciálegyenlet-rendszert, illetve az ahhoz tartozó kezdetiérték-feladatot nevezzük [3]:

$$\dot{c}_i(t) = \sum_{j=1}^{r} \left(k_j \, (\beta_{i,j} - \alpha_{i,j}) \prod_{l=1}^{m} c_l(t)^{\alpha_{l,j}} \right) \qquad (i = 1, \ldots, m), \tag{4.1}$$

ahol a k_j adott pozitív valós számok a reakciólépések **sebességi állandó**i (ezek mutatják, milyen gyorsan zajlanak le a reakciók), a $c_i(t)$ szám pedig az i-edik anyagfajta koncentrációja a t időpillanatban. (A jobb oldalon esetenként megjelenő 0^0 tényezőket 1-nek értelmezzük.)

A fenti (ún. **tömeghatás típusú**) modell mögött az az intuitív gondolat húzódik meg, hogy egy reakciólépés sebessége egyenesen arányos a résztvevő anyagok koncentrációjával (pontosabban minden résztvevő anyag koncentrációjának annyiadik hatványával, ahány molekulája részt veszt a reakciólépésben), tehát azok szorzatával is. Az egyes anyagfajták termelődésének sebessége pedig egyenesen arányos a reakciólépések sebességével. A fenti egyenlet általánosabban is felírható úgy, hogy a jobb oldal nem a fenti alakú, de a dolgozatban csak ezt a – legelterjedtebb – modellt vizsgáljuk.

Érdemes megemlíteni, hogy a bevezetett modell számos olyan tulajdonsága levezethető, amit minden „helyes" modelltől elvárunk. Tehát nem csupán a modell egyszerű, intuitív volta, hanem annak matematikailag bizonyíthatóan teljesülő tulajdonságai is indokolják a használatát. Egy ilyen tulajdonság például, hogy az első ortánsból induló min-

den megoldás benne marad az első ortánsban – azaz miként a valóságban, a modellben sem lehet egy anyagfajta koncentrációja negatív. Még pontosabban az a természetesen elvárható tulajdonság is teljesül, hogy egy $\mathbf{c}(0)$ pontból induló megoldás nem lép ki a ponthoz tartozó **reakciószimplex**ből, azaz a $(\mathbf{c}(0) + S(\gamma)) \cap (\mathbb{R}_0^+)^m$ halmazból. Ezt az állítást pontosítja egy speciális esetben a 4.1. tétel (4.1.1. szakasz).

Példa A biológiai (populációdinamikai) modellek vizsgálatakor is gyakran idézett példa a Lotka–Volterra modell (ragadozó-zsákmány vagy élősködő-gazda modell), amely reakciókinetikai formában a következő alakban írható:

$$R = \{X \to 2X, \quad X + Y \to 2Y, \quad Y \to \emptyset\}.$$

A formális kémiai reakció adatai: $M = \{X, Y\}$, $m = 2$, $r = 3$, (a reakciólépések halmaza a fenti), $\alpha = \left(\begin{smallmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{smallmatrix}\right)$, és $\beta = \left(\begin{smallmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \end{smallmatrix}\right)$. A három reakciólépés sebességi állandóját rendre k_1, k_2, k_3-mal, a két anyagfajta koncentrációját pedig x-szel és y-nal jelölve az anyagfajták koncentrációjára felírható differenciálegyenlet-rendszer a következő:

$$\begin{aligned} \dot{x}(t) &= k_1 x(t) - k_2 x(t) y(t), \\ \dot{y}(t) &= k_2 x(t) y(t) - k_3 y(t). \end{aligned}$$

4.1.1. A modell megoldásainak kvalitatív jellemzői

A (4.1) differenciálegyenlet szimbolikus megoldása általában reménytelen – például már az előbbi egyszerű példa esetében sem írható fel a megoldás véges sok elemi művelet segítségével –, és a numerikus megoldás is nehéz. Ezért érdekes kérdés, hogy mit mondhatunk a megoldás kvalitatív jellemzőiről az egyenletrendszer megoldása nélkül.

Elsőként a 3.5.3. szakaszban már felhasznált Volpert-indexezést mutatjuk be részletesen, majd egy, a stacionárius megoldások létezésére vonatkozó tételt ismertetünk röviden [3, 2. fejezet] alapján; az ezekhez tartozó vizsgálatokat a reakciókinetikai programcsomagban (5. fejezet) is implementáltam.

Volpert-indexezés

Kezdeti anyagfajtának nevezzük az (M, R, α, β) formális kémiai mechanizmus anyagfajtáinak azon M_0 részhalmazát, melyeknek a kezdeti koncentrációja pozitív, azaz amelyek helyei a mechanizmust leíró Petri-hálóban jelölve vannak: $M_0 := \{i \in M \mid c_i(0) > 0\}$.

Volpert egy tétele szerint ha az (M, R, α, β) formális kémiai mechanizmusban minden reakciólépésnek van reaktánsa (azaz ha az α mátrix egyik oszlopa sem nullvektor), akkor egy anyag koncentrációja mindvégig zérus marad ($c_i(t) = 0$) akkor és csak akkor,

ha végtelen indexet kap a következő indexezési eljárás során [33]: a kezdeti anyagok indexe legyen zérus, továbbá minden reakciólépés indexe legyen a reaktánsai indexének maximuma, végül a nem kezdeti anyagok kapjanak eggyel nagyobb indexet, mint az őket előállító legkisebb indexű reakció. (Az üres halmaz minimumát végtelennek definiáljuk.) Az indexeket a 4.1. algoritmussal határozhatjuk meg.

4.1. algoritmus: Volpert-indexezés

Bemenet: (M, R, α, β) formális kémiai mechanizmus, $M_0 \subset M$ kezdeti anyagok.
Kimenet: Az indexezett anyagok és reakciólépések M_i és R_i halmazai.

Jelölés: Az r reakciólépés reaktánsait és termékeit $R(r)$ és $P(r)$ jelöli

1 $R_0 \leftarrow \{r \in R \mid R(r) \subset M_0\}$;
2 $i \leftarrow 1$;
3 **repeat**
4 $\quad M_i \leftarrow \bigcup_{r \in R_{i-1}} P(r) \setminus \bigcup_{j=0}^{i-1} M_j$; `// anyagfajták indexei`
5 $\quad R_i \leftarrow \{r \in R \mid R(r) \subset \bigcup_{j=0}^{i} M_j\} \setminus \bigcup_{j=0}^{i-1} R_j$; `// reakciók indexei`
6 $\quad i \leftarrow i + 1$
7 **until** $R_i = \emptyset$;
8 $M_\infty \leftarrow M \setminus \bigcup_{j=0}^{i-1} M_j$;
9 $R_\infty \leftarrow R \setminus \bigcup_{j=0}^{i-1} R_j$

Egy anyagfajta, illetve reakciólépés **Volpert-index**e j, ha az eljárás végén az M_j, illetve R_j halmazba kerül. Szemléletesen egy páros gráf csúcsait indexezzük, amelynek két pontosztálya az anyagfajták és a reakciólépések. Ez a gráf – melyet a formális kémiai mechanizmus **Volpert-gráf**jának nevezünk – voltaképpen megegyezik a Petri-hálóval, jelölés és súlyfüggvény nélkül. Hatékony implementáció esetén az indexezett gráf minden csúcsát egyszer vesszük sorra, és tesszük valamelyik M_j vagy R_j halmazba, így az algoritmus lineáris idejű.

Ez az eljárás eredendően a megoldások kvalitatív jellemzésére szolgál – az idézett tételen kívül a megoldások $t = 0$ körüli viselkedését is jellemzi –, emellett azonban a bruttó reakciók felbontásainak vizsgálatára is használható. Ezt először a [15] dolgozatban mutatták meg, az ott alkalmazott módszer a 3.7. alfejezetben bemutatott kémiai problémák vizsgálatánál is előkerül, ezért a reakciókinetikai programcsomagban is implementáltam. Egy egyszerűsített változatát pedig előfeldolgozási lépésként is használhatjuk, ezt a változatot ismerteti a 3.5.3. szakasz. Az előbbi formalizmussal az egyszerűsített algoritmus úgy írható le, hogy nem indexezzük az M_j és R_j halmazokat, hanem egy-egy (M_0 és R_0) halmazban gyűjtjük a véges indexű anyagfajták és reakciók csúcsait.

A Feinberg–Horn–Jackson-gráf

Egy másik, összetett kémiai reakcióhoz rendelt gráf, melynek tulajdonságaiból következtethetünk a (4.1) egyenlet megoldásainak tulajdonságaira a Feinberg–Horn–Jackson-gráf. Mivel ez az előző szakaszban bemutatott Volpert-gráffal ellentétben nem kapcsolódik szorosan más, e dolgozatban tárgyalt problémákhoz, csak egy rövid leírást adunk [3, 2.3.6. szakasz] alapján, amely azonban már elégséges ahhoz, hogy az elkészítendő reakciókinetikai programcsomagban (5. fejezet) a megfelelő programrészeket elkészíthessük. A gráfhoz kapcsolódó 4.1. tétel érdekes példája annak, hogy pusztán a reakciólépések kapcsolatának struktúrájából kombinatorikus módszerekkel következtethetünk a kinetikai differenciálegyenletek megoldásainak korántsem nyilvánvaló kvalitatív tulajdonságaira – ebben az esetben arra, hogy bizonyos kombinatorikus feltételek mellett periodikus megoldások nem létezhetnek, aszimptotikusan stabilis megoldásoknak azonban léteznüük kell.

Az (M, R, α, β) formális kémiai mechanizmus **komplex**einek a reakciólépések két oldalán szereplő anyagkombinációkat nevezzük, ezek megfeleltethetők az α és β mátrix különböző oszlopainak. (Például a 4.1. alfejezetben látott Lotka–Volterra-modell komplexei: $\emptyset, X, 2X, Y, 2Y$ és $X + Y$.) A mechanizmus **Feinberg–Horn–Jackson-gráf**ja (röviden **FHJ-gráf**ja) az a gráf, amelynek csúcsai a komplexek (minden komplex csak egyszer szerepelhet), és egy (C_1, C_2) rendezett pár akkor (irányított) éle a gráfnak, ha van $C_1 \to C_2$ reakciólépés a mechanizmusban. Egy mechanizmust **gyengén megfordítható**nak nevezünk, ha minden olyan (C_1, C_2) párra, amelyre az FHJ-gráfban létezik $\overrightarrow{C_1C_2}$ irányított út, hasonlóképpen létezik $\overrightarrow{C_2C_1}$ irányított út is. Igazolható a következő tétel.

4.1. tétel (zéró deficiencia-tétel) *Jelölje* N *az* (M, R, α, β) *formális kémiai mechanizmusban előforduló komplexek számát,* L *az FHJ-gráf erősen összefüggő komponenseinek számát,* S *pedig a* $\gamma := \beta - \alpha$ *mátrix rangját, és tegyük fel, hogy a formális mechanizmus* $\delta := N - L - S$ *egyenlőséggel definiált* ***deficiencia****ja zérus. Ekkor*

1. *A* (4.1) *kinetikai differenciálegyenletnek akkor és csak akkor létezik pozitív stacionárius pontja, ha a mechanizmus gyengén megfordítható.*

2. *Ha a mechanizmus gyengén megfordítható, akkor a sebességi állandók tetszőleges értéke esetén minden reakciószimplexben létezik pontosan egy stacionárius pont. Minden stacionárius pont relatíve, lokálisan aszimptotikusan stabilis. Nemtriviális (azaz nem konstans) periodikus megoldások nem léteznek.*

Példa Tekintsük az $X \rightleftharpoons Y$ két reakciólépésből álló kémiai mechanizmust. Erre $m = r = 2$, $\gamma = \left(\begin{smallmatrix} -1 & 1 \\ 1 & -1 \end{smallmatrix}\right)$, azaz a deficiencia $\delta = N - L - S = 2 - 1 - 1 = 0$, hiszen az FHJ-gráf egyetlen

– kétpontú – erősen összefüggő komponensből áll, és rank $\gamma = 1$. A mechanizmus gyengén megfordítható, tehát a reakciólépések sebességétől függetlenül léteznie kell pozitív stacionárius pontjának is.

Ha az X→Y reakció sebességi állandója $k_1 > 0$, míg a fordított irányúé $k_2 > 0$, a (4.1) kinetikai differenciálegyenletek az alábbi alakot öltik:

$$\begin{aligned} \dot{x}(t) &= -k_1 x(t) + k_2 y(t), \\ \dot{y}(t) &= k_1 x(t) - k_2 y(t), \end{aligned}$$

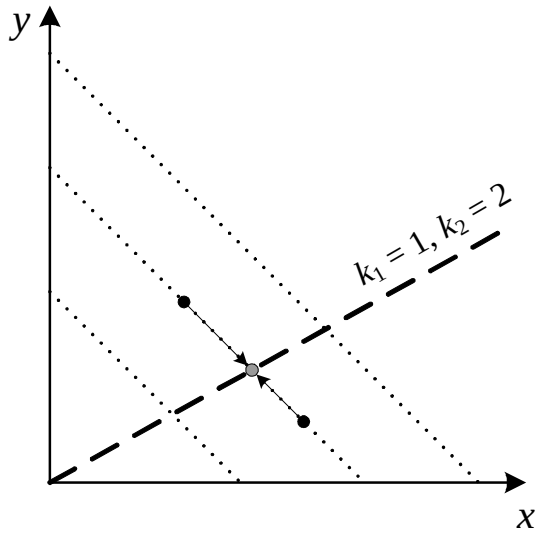
valamilyen $\mathbf{c}_0 = (x(0), y(0)) = (x_0, y_0)$ nemnegatív kezdeti koncentrációvektorral. Az egyenletet megoldva a koncentráció–idő függvények:

$$\begin{aligned} x(t) &= \frac{1}{k_1 + k_2}\left(x_0(k_2 + k_1 e^{-(k_1+k_2)t}) + y_0(k_2 - k_2 e^{-(k_1+k_2)t})\right), \\ y(t) &= \frac{1}{k_1 + k_2}\left(x_0(k_1 - k_1 e^{-(k_1+k_2)t}) + y_0(k_1 + k_2 e^{-(k_1+k_2)t})\right). \end{aligned}$$

A megoldásból azonnal leolvasható, hogy a megoldásvektor bennmarad a $\mathbf{c}_0$-hoz tartozó reakciószimplexben, azaz a $(\mathbf{c}_0 + S(\gamma)) \cap (\mathbb{R}_0^+)^m$ halmazban, hiszen $x_0, y_0, t \geq 0$ és $k_1, k_2 > 0$ miatt $x(t), y(t) \geq 0$, továbbá $x(t) + y(t) \equiv x_0 + y_0$. (Ez utóbbi a differenciálegyenletekből is azonnal látszik, hiszen $\dot{x}(t) + \dot{y}(t) = 0$.)

A megoldás határértékét véve kapjuk, hogy az (x_0, y_0) kezdeti értékhez tartozó megoldás a $\left(\frac{k_2(x_0+y_0)}{k_1+k_2}, \frac{k_1(x_0+y_0)}{k_1+k_2}\right)$ stacionárius ponthoz tart, amely ezek szerint létezik és egyértelmű a kezdeti értékhez tartozó reakciószimplexben. A stacionárius pontokból induló konstans megoldásokon kívül periodikus megoldás tehát nem létezik. Ha a stacionárius pontból úgy mozdítjuk ki a megoldást, hogy az a reakciószimplexben marad, akkor a megoldás visszatér az eredeti stacionárius pontba. Ezzel a 4.1. tétel második pontjának minden állítását illusztráltuk a példán.

Az állításokat szemlélteti a 4.1. ábra. A pontozott vonalak a (példánkban egydimenziós) reakciószimplexek, minden szimplexben pontosan egy stacionárius pont van, ezek halmazát jelzi a vastag szaggatott vonal. A szürke pont egy stacionárius pont. A hozzá tartozó reakciószimplex pontjaiból indított megoldások ehhez a ponthoz tartanak.



4.1. ábra. A zéró deficiencia-tétel szemléltetéséhez

4.2. A folytonos idejű, diszkrét állapotterű, sztochasztikus (CDS) modell

Az ebben a szakaszban ismertetett modellben az anyagfajták mennyiségét a molekulák számának (nemnegatív egész) vektorával írjuk le, a reakciók bekövetkezését pedig sztochasztikusnak tekintjük, azaz hagyományos sztochasztikus Petri-hálókkal dolgozunk, melyben egy átmenet tüzeléséhez – azaz egy reakciólépés bekövetkezéséhez – rendelt exponenciális eloszlású valószínűségi változó paramétere most a reakciólépés sebességének és a reakcióban résztvevő anyagok koncentrációinak függvénye.

Az (M, R, α, β) összetett kémiai reakció **folytonos idejű, diszkrét állapotterű, sztochasztikus** (vagy **CDS**) **modell**jének azt az $\mathbf{X}(t) = (X_1(t), \ldots, X_m(t))^T$ megszámlálható állapotterű, vektorértékű, folytonos paraméterű, homogén sztochasztikus Markov-folyamatot nevezzük, melynek állapottere $\mathbb{N}_0^n$, infinitezimális generátormátrixa (vagy rátamátrixa) az a $\mathbf{Q} = [q_{\mathbf{p},\mathbf{r}}]$ végtelen mátrix, melynek általános eleme (a korábban is használt jelölésekkel $\gamma := \beta - \alpha$; a γ mátrix i-edik oszlopvektora γ_i):

$$q_{\mathbf{p},\mathbf{r}} := \sum_{i:\, \gamma_i = \mathbf{r}-\mathbf{p}} k_i \prod_{j=1}^{m} (p_j)_{\alpha_{j,i}}, \text{ ha } \mathbf{p} \neq \mathbf{r}, \tag{4.2a}$$

$$q_{\mathbf{p},\mathbf{p}} := -\sum_{\mathbf{r}:\, \mathbf{r}\neq\mathbf{p}} q_{\mathbf{p},\mathbf{r}}\,, \tag{4.2b}$$

ahol $(a)_k$ az

$$(a)_0 := 1; \quad (a)_k := a(a-1)\cdots(a-k+1) \quad (k \geq 1)$$

egyenletekkel definiált leszálló faktoriális hatvány függvény, a k_i pozitív konstansok pedig a reakciólépések sebességi együtthatói. Vagyis csak azon állapotpárok közötti átmenetekhez tartozó érték lehet pozitív, amelyek különbsége az R-beli reakciók valamelyikének hatásával egyenlő; ilyenkor a mátrix megfelelő eleme az állapotváltozást előidéző reakciólépések sebességeinek összege, az egyes reakciók sebessége pedig a rendelkezésre álló anyagok mennyiségétől függ – speciálisan, ha $p_j < \alpha_{j,i}$, azaz Petri-hálós szóhasználattal élve az i-edik átmenet nincs engedélyezve, mert a j-edik helyen nincs elegendő számú token, akkor a sebesség zérus. Ha a tokenek száma nagyon nagy ($p_j \gg 0$), akkor a leszálló faktoriális hatvány közönséges hatványozással közelíthető, (4.2a)-ban $(p_j)_{\alpha_{j,i}}$ helyett $p_j^{\alpha_{j,i}}$ írható.

4.2.1. Sztochasztikus szimuláció

A sztochasztikus Petri-hálókkal modellezett rendszerek vizsgálatát praktikus okokból a leggyakrabban szimulációval végzik: a rendszert a kezdőállapotából sokszor – az alkalmazott véletlenszám-generátor különböző kezdőértékei mellett – újraindítják, majd meghatározott ideig futni hagyják. E módszer egyik előnye, hogy a rendszer vizsgált paramétereit a szimuláció eredményéből kiolvasni sokkal egyszerűbb, mint a Petri-hálóhoz tartozó homogén Markov-lánc – esetleg kivitelezhetetlen – szimbolikus analízisét elvégezni. Egy másik érv a szimuláció mellett, hogy az egyszerű sztochasztikus Petri-hálóknál általánosabb hálók estén is lényeges módosítás nélkül alkalmazható. Ilyen – elsősorban az informatikai és a megbízhatósági modellezésben elterjedt – általánosabb hálótípus például az általánosított sztochasztikus Petri-háló (GSPN), amely időzítetlen átmeneteket is tartalmaz. (Az időzítetlen átmeneteknek azonnal kell tüzelniük; többek között logikai feltételek modellezhetők a segítségükkel.) Egy további általánosabb Petri-háló típus a determinisztikus-sztochasztikus Petri-háló (DSPN), amely determinisztikusan időzített (az engedélyezettségük után rögzített Δt idővel tüzelő) átmeneteket is tartalmaz. A GSPN-ek hibatűrő, a DSPN-ek pedig elsősorban valósidejű rendszerek modellezésében használatosak [2]. Ugyancsak lényegében módosítás nélkül alkalmazhatók a szimulációs módszerek félmarkov-folyamatokkal leírható modellek esetén. E szakaszban a sztochasztikus szimuláció módszereit ismertetem.

A szimuláció egy lehetséges megvalósítása a 4.2. algoritmus, amely egy megadott időpontig szimulálja a Petri-háló működését. Az algoritmus egy segédfüggvényt használ: a `VSzám`(λ) függvény egy λ paraméterű exponenciális eloszlású véletlen számot ad vissza.

Az eljárás úgy szemléltethető, hogy minden átmenethez egy-egy „ébresztőórát" rendelünk, amelyet (exponenciális eloszlású) véletlen időpontra állítunk be. Az az átmenet tüzel, amelyiknek az órája a leghamarabb jár le; ilyenkor az eloszlások paramétereit a megváltozott jelölésnek megfelelően újraszámoljuk (4.2a) szerint, és az órákat ez alapján újra beállítjuk. Minthogy az átmenetek tüzelése – azaz a reakciók bekövetkezése – egy-

4.2. algoritmus: Sztochasztikus szimuláció I.

Bemenet: (M, R, α, β) Petri-háló, M_0 kezdeti jelölés, T időpont.
Kimenet: A Petri-háló által generált sztochasztikus folyamat egy realizációja; a folyamat állapota az $[x_i, x_{i+1})$ időintervallumban y_i.

```
1  t ← 0; p ← M_0; lépés ← 0;
2  while t < T do
3      x_lépés ← t ;
4      y_lépés ← p ;
5      for i = 1 to r do
6          Δ_i ← VSzám(1/q_{p,p+γ_i})
7      i ← arg min_j Δ_j ;
8      t ← t + Δ_i ;
9      p ← p + γ_i ;
10     lépés ← lépés + 1
11 x_lépés ← T ;
12 return (x, y)
```

mástól teljesen független, ez a szimulációs stratégia igen szemléletes. Létezik azonban hatékonyabb stratégia is, jobb megvalósítás a 4.3. algoritmus.

4.3. algoritmus: Sztochasztikus szimuláció II.

Bemenet: (M, R, α, β) Petri-háló, M_0 kezdeti jelölés, T időpont.
Kimenet: A Petri-háló által generált sztochasztikus folyamat egy realizációja; a folyamat állapota az $[x_i, x_{i+1})$ időintervallumban y_i.

```
1  t ← 0; p ← M_0; lépés ← 0;
2  while t < T do
3      x_lépés ← t ;
4      y_lépés ← p ;
5      t ← t+ VSzám(∑ q_{p,r}) ;
6      p ← VÁllapot(α, β, p) ;
7      lépés ← lépés + 1 ;
8  x_lépés ← T ;
9  return (x, y)
```

Ez az algoritmus egy további segédfüggvényt használ. A `VÁllapot`$(\alpha, \beta, \mathbf{p})$ függvény egy $\mathbf{p}$-től különböző véletlen $\mathbf{r}$ állapotot sorsol ki az egy lépésben pozitív valószínűséggel elérhető állapotok, azaz a $\mathbf{p} + \gamma_i, (i = 1, \ldots, r)$ állapotok közül úgy, hogy a $\mathbf{p} + \gamma_i$ állapot valószínűsége (4.2a) jelöléseivel $\frac{q_{\mathbf{p},\mathbf{p}+\gamma_i}}{\sum_i q_{\mathbf{p},\mathbf{p}+\gamma_i}}$.

Az algoritmus működése úgy szemléltethető, hogy előbb kisorsoljuk, mikor tüzel *valamelyik* átmenet, majd azt, hogy *melyik* átmenet tüzel. Megmutatható, hogy ha a második megvalósításban a véletlen számokat és állapotokat a megadott származtatott paraméte-

rekkel sorsoljuk, akkor a két modell ekvivalens. (Sajnos több nagy lélegzetű, összefoglaló munka is, mint például a [2,3] könyvek is vagy csak az első, szemléletes, vagy csak a második, hatékony stratégiát említi meg.) Részletes bizonyítás helyett csak annyit jegyzünk meg, hogy az állítás a következő észrevételek segítségével igazolható:

4.2. lemma *Legyenek* $X_1, \dots X_r$ *rendre* $\lambda_1, \dots, \lambda_r$ *paraméterű exponenciális eloszlású, teljesen független valószínűségi változók. Ekkor*

1. *Az* $A := \min_i X_i$ *valószínűségi változó egy* $\sum_i \lambda_i$ *paraméterű exponenciális eloszlású valószínűségi változó,*

2. *A* $B := \arg\min_i X_i$ *valószínűségi változó egy* $\mathbf{P}(B = k) = \frac{\lambda_k}{\sum_l \lambda_l}$ $(k = 1, \dots, r)$ *eloszlású valószínűségi változó, továbbá*

3. A *és* B *függetlenek.*

A második, kevésbé intuitív szimulációs stratégia mellett az szól, hogy egy szimulációs lépéshez az átmenetek (reakciók) számától függetlenül csupán két véletlen számot kell sorsolnunk, míg az első megvalósításban r számú átmenet (reakció) esetén r független véletlen számra van szükségünk lépésenként, míg az egyéb számítási költségek lényegében azonosak.

Az előbbiek alapján a második stratégia látszólagos aszimmetriája is egyszerűen magyarázható: a tüzelő átmenetet és a tüzelési időpontot kijelölő valószínűségi változók a 4.2. lemma állítása szerint függetlenek, így a sorsolásuk sorrendje felcserélhető.

4.3. A CCD és CDS modellek kapcsolata

Ebben a szakaszban a két bemutatott modell viszonyával kapcsolatos néhány eredményt mutatok be Kurtz [17] publikációja és a [3, 3. fejezet] könyvfejezet alapján, egyszerűsített levezetésekkel. A leglényegesebb idézett eredmény szemléletesen úgy fogalmazható, hogy a sztochasztikus modell realizációjának várható értéke „majdnem kielégíti" a determinisztikus modell differenciálegyenletét.

Mint a folytonos idejű Markov-láncok vizsgálatánál általában, a CDS modellű folyamat működését a (4.2) egyenletekkel ekvivalens módon a következőképpen is leírhatjuk. A rendszer állapotának megváltozása egy rövid időintervallumban ($t \geq 0$ tetszőleges, $s \geq 0$ kicsi) a (4.2) egyenletben bevezetett jelölésekkel:

$$\mathbf{P}(\mathbf{X}(t+s) = \mathbf{r} \,|\, \mathbf{X}(t) = \mathbf{p}) = q_{\mathbf{p},\mathbf{r}} s + \varepsilon(s)s, \tag{4.3}$$

ahol $\lim_{s \to 0} \varepsilon(s) = 0$.

Írjuk fel a teljes valószínűség tételét $\mathbf{X}$ megváltozására egy rövid intervallumon (4.3) felhasználásával:

$$\begin{aligned}\mathbf{P}(\mathbf{X}(t+s)-\mathbf{X}(t)=\mathbf{d}) &= \sum_{\mathbf{p}} \left(\mathbf{P}(\mathbf{X}(t+s)=\mathbf{p}+\mathbf{d}\,|\,\mathbf{X}(t)=\mathbf{p})\right)\mathbf{P}(\mathbf{X}(t)=\mathbf{p}) \\ &= \sum_{\mathbf{p}} \left(q_{\mathbf{p},\mathbf{p}+\mathbf{d}}s+\varepsilon(s)s\right)\mathbf{P}(\mathbf{X}(t)=\mathbf{p}).\end{aligned}$$

Eszerint a megváltozás várható értéke:

$$\begin{aligned}\mathbf{E}\big(\mathbf{X}(t+s)-\mathbf{X}(t)\big) &= \sum_{\mathbf{d}} \mathbf{P}(\mathbf{X}(t+s)-\mathbf{X}(t)=\mathbf{d})\mathbf{d} \\ &= \sum_{\mathbf{d}} \mathbf{d}\sum_{\mathbf{p}} \left(q_{\mathbf{p},\mathbf{p}+\mathbf{d}}s+\varepsilon(s)s\right)\mathbf{P}(\mathbf{X}(t)=\mathbf{p}).\end{aligned}$$

Osszuk mindkét oldalt s-sel, helyettesítsük be (4.2)-t, végül vegyük mindkét oldal határértékét $s\to 0$ esetén! (A jobb oldalon a határátmenet nyilvánvalóan elvégezhető, tehát a bal oldal határértéke is létezik):

$$\begin{aligned}\lim_{s\to 0}\frac{\mathbf{E}\,\mathbf{X}(t+s)-\mathbf{E}\,\mathbf{X}(t)}{s} &= \\ \frac{d}{dt}\mathbf{E}\,\mathbf{X}(t) &= \sum_{\mathbf{d}}\mathbf{d}\sum_{\mathbf{p}}\sum_{i:\,\gamma_i=\mathbf{d}} k_i\prod_{j=1}^{m}(p_j)_{\alpha_{j,i}}\mathbf{P}(\mathbf{X}(t)=\mathbf{p}) \\ &= \sum_{i=1}^{r}\sum_{\mathbf{p}} k_i\gamma_i\prod_{j=1}^{m}(p_j)_{\alpha_{j,i}}\mathbf{P}(\mathbf{X}(t)=\mathbf{p}) \\ &= \sum_{\mathbf{p}}\mathbf{P}(\mathbf{X}(t)=\mathbf{p})\sum_{i=1}^{r} k_i\gamma_i\prod_{j=1}^{m}(p_j)_{\alpha_{j,i}} \\ &= \mathbf{E}\,f_2(\mathbf{X}(t)),\end{aligned}$$

bevezetve az $f_2(\mathbf{p}) := \sum_{i=1}^{r} k_i\gamma_i\prod_{j=1}^{m}(p_j)_{\alpha_{j,i}}$ jelölést. Így egy differenciálegyenletet kaptunk a sztochasztikus modell várható értékére. Ha $\mathbf{X}(t)\gg 0$, akkor a leszálló faktoriális hatvány jól közelíthető egyszerű hatványozással, így kapjuk a következő, a CCD-modellre még jobban emlékeztető egyenletet:

$$\frac{d}{dt}\mathbf{E}\,\mathbf{X}(t)=\mathbf{E}\,f(\mathbf{X}(t)), \tag{4.4}$$

ahol $f(\mathbf{p}) := \sum_{i=1}^{r} k_i\gamma_i\prod_{j=1}^{m} p_j^{\alpha_{j,i}}$. Anyagfajtánként felírva:

$$\frac{d}{dt}\mathbf{E}\,X_i(t)=\mathbf{E}\left(\sum_{j=1}^{r} k_j\,(\beta_{i,j}-\alpha_{i,j})\prod_{l=1}^{m}\big(X_l(t)\big)^{\alpha_{l,j}}\right)\qquad (i=1,\ldots,m). \tag{4.5}$$

A (4.5) egyenlet már majdnem analóg a CCD modell (4.1) egyenletével Az f függvény azonban általában nem lineáris – pontosabban csak ún. *elsőrendű*, azaz egyetlen reaktánsú reakciók esetén az[1] –, ezért általában nem cserélhető fel a várhatóérték-képzés operátorával. A (4.1) egyenlet legpontosabb analogonja a $\frac{d}{dt}\mathbf{E}\,\mathbf{X}(t) = f(\mathbf{E}\,\mathbf{X}(t))$ egyenlet volna („a sztochasztikus modell várhatóérték–idő függvénye kielégíti a determinisztikus modellre vonatkozó differenciálegyenletet"), ez azonban általában nem teljesül, csupán a hasonló, közelítő (4.4)–(4.5) alakban.

Ugyancsak az f illetve f_2 függvénnyel írható fel a *feltételes várható sebesség*re vonatkozó egyenlet. Írjuk fel az $\mathbf{X}$ megváltozásának feltételes várható értékét egy kis intervallumon (a feltétel a pillanatnyi állapot):

$$\begin{aligned}\mathbf{E}\big(\mathbf{X}(t+s)-\mathbf{X}(t)\,|\,\mathbf{X}(t)=\mathbf{p}\big) &= \sum_{\mathbf{d}} \mathbf{d}\;\mathbf{P}\big(\mathbf{X}(t+s)-\mathbf{X}(t)=\mathbf{d}\,|\,\mathbf{X}(t)=\mathbf{p}\big)\\ &= \sum_{\mathbf{d}} \mathbf{d}\;\mathbf{P}\big(\mathbf{X}(t+s)=\mathbf{p}+\mathbf{d}\,|\,\mathbf{X}(t)=\mathbf{p}\big)\\ &= \sum_{\mathbf{d}} \mathbf{d}\,\big(q_{\mathbf{p},\mathbf{p}+\mathbf{d}}s+\varepsilon(s)s\big)\,.\end{aligned}$$

Ismét s-sel osztva, behelyettesítve (4.2)-t, és véve mindkét oldal (létező) $s\to 0$ határértékét, rövid számolással a

$$\frac{d}{dt}\mathbf{E}\big(\mathbf{X}(t)\,|\,\mathbf{X}(t)=\mathbf{p}\big)=f_2(\mathbf{p}) \tag{4.6}$$

egyenlőség adódik. (Vagy $\frac{d}{dt}\mathbf{E}\big(\mathbf{X}(t)\,|\,\mathbf{X}(t)=\mathbf{p}\big)\approx f(\mathbf{p})$, a korábbi $f_2(\mathbf{p})\approx f(\mathbf{p})$ közelítést alkalmazva.) Ezzel a (4.1) kinetikai differenciálegyenlet jobb oldalán található függvénynek a sztochasztikus modellben is szemléletes jelentést tulajdoníthatunk.

[1] Ugyanez a Petri-hálók nyelvén megfogalmazva: minden átmenetnek egyetlen bemenő helye van, és minden olyan él súlya egy, amely helytől átmenet felé mutat.

5. fejezet

A reakciókinetikai programcsomag

Ebben a fejezetben bemutatom a korábbiakban ismertetett eredmények felhasználásával általam készített, általános célú reakciókinetikai programcsomagot. Először röviden ismertetek néhány korábbi, reakciókinetikai problémák, illetve Petri-háló alapú modellezés számítógépi támogatására készült programot, azok leglényegesebb előnyeire és hiányosságaira koncentrálva. Az ezekből nyert tapasztalatok nagy mértékben meghatározták az általam elkészítendő programmal szemben támasztott követelményeket is. Ezután áttérek az implementáció részleteire, valamint a program bemutatására.

5.1. A program előzményei

Petri-hálók analízise és szimulációja. Petri-háló analizáló programból számtalan készült már, csak néhány a legelterjedtebb, PC-s platformokon futó, ingyenes szoftverek közül, amelyek képesek egyszerűbb analízis feladatokat megoldani, például T- és P-invariánsokat keresni [37]: Integrated Net Analyzer (INA), Platform Independent Petri Net Editor (PIPE), Programming Environment based on Petri Nets (PEP), Petri Net Kernel, Analisador de Redes de Petri (ARP), Predator. E programok többsége azonban meglehetősen régi keletű, és a nyílt forráskódúak is rosszul vannak dokumentálva; terméktámogatás nélküli fejlesztésük, hibajavításuk rendkívül nehéz feladat, ugyanakkor számos programhiba nehezíti a használatukat.

Sztochasztikus szimulációra készült ingyenes, elterjedt szoftverek például a PetriSim, a PNTalk vagy a már említett PIPE. A Petri-háló szimulátorok kémiai mechanizmusok szimulációjára is alkalmazhatók. A legegyszerűbb ingyenes termékek is megfelelnek a célnak, hiszen csak egyszerű sztochasztikus Petri-hálókat (SPN) kell szimulálnunk a 4.2.1. szakaszban ismertetett módon. (A legtöbb szoftver SPN-t, illetve a 4.2.1. szakaszban röviden szintén megemlített általánosított sztochasztikus Petri-hálót, GSPN-t, is szimulál.) E programok használatának egyetlen hátránya, hogy semmilyen felületet

nem adnak a reakciókinetikai alkalmazáshoz. (A programokhoz nem kapcsolható például olyan *plug-in*, amellyel a reakciókinetika és a Petri-hálók nyelve közötti fordítás elvégezhető.) Ha mégis ezeket választjuk, a legtöbb, amit megtehetünk, hogy olyan programot keresünk, amelynek kvázi-szabványos, vagy legalábbis nyílt Petri-háló formátuma van, majd a reakciókinetikai célú program kimenetét ugyanilyen formátumra hozzuk. Ilyen, a közelmúltban egyre elterjedtebbé váló, de még napjainkban is fejlesztés alatt álló[1] formátum a PNML (*Petri Net Markup Language*), amelyet egyre több eszköz támogat, például a már említett PIPE és a PEP is. E nyelv előnye, hogy XML-alapú, így írásához, olvasásához és más XML formátumokra konvertálásához (mint amilyen a biokémiai reakcióhálózatok leírására alkalmas, szintén fejlesztés alatt álló *Systems Biology Markup Language* [38]) szinte minden korszerű programozási nyelven találunk kész függvény- (vagy objektum-) könyvtárat.

Elemi reakciók előállítása. A bruttó reakciók felbontásával foglalkozó szerzők rendszerint feltételezik, hogy az elemi reakciólépések listája könnyen meghatározható, és így a bruttó reakcióval együtt adva van. Ennek gyakran az az eredménye, hogy a folyamat modellje a valóságosnál sokkal egyszerűbb, a figyelembe vett elemi lépések száma gyakran nem, vagy alig nagyobb a reakcióban résztvevő anyagok számánál [5,28]. Kifejezetten erre a célra készített programok híján a lineáris diofantoszi egyenletrendszerek megoldására képes matematikai programcsomagok használhatók. (Például a Contejean–Devie-algoritmusra építő *Mathematica* 5.)

Felbontások előállítása. A felbontások előállítására szolgáló általános célú program mindeddig nem készült. A reakciók felbontásával kapcsolatos vizsgálatok során természetesen korábban is alkalmaztak számítógépes módszereket, azonban a meglévő eredmények többnyire *ad hoc* megoldásokra épülnek, és felhasználnak – jellemzően csak szerves reakciók esetén – rendelkezésre álló termodinamikai adatokat. Jól ismerik például a szénhidrogének égési folyamatait [30], a szervetlen reakciók felbontásáról azonban sokkal kevesebbet tudni. A cél ezért olyan program készítése volt, amely általános célú, azaz kiegészítő kémiai adatok nélkül is használható, ugyanakkor lehetőséget ad a felhasználónak kiegészítő feltételek (vagy akár teljes algoritmusok) beépítésére is a felbontások meghatározása során.

[1] Bár e formátum már nem mondható gyerekcipőben járónak, a szabványosítása körüli problémák megoldása napjainkban is olyan önálló konferenciák témája, mint például az e dolgozat elkészültének hónapjában esedékes *Second Workshop on the Petri Net Markup Language 2005 (PNML 05) - "Towards an ISO/IEC Standard Transfer Syntax for Petri Nets"*. Az érdeklődő olvasó/fejlesztő az ehhez hasonló workshopok konferenciakiadványaiból bőségesen meríthet információt.

Kinetikai differenciálegyenletek megoldása. Erre a feladatra sokféle program készült korábban, de ezek is jobbára speciális egyenletek megoldásával foglalkoznak. Vélhetően a legelső ilyen, kereskedelmi forgalomba hozott program a John Chesick által 1979-ben FORTRAN nyelven írt megoldóprogram [4]. Az egyébként számtalan (fizetős) mérnöki és tudományos programcsomaggal kiegészített matematikai szoftverekhez (*Mathematica*, *Maple*, *Matlab*) nem kapható általános célú reakciókinetikai csomag, a *Mathematica* fejlesztőinek javaslata, hogy írjuk fel az egyenleteket kézzel, majd oldassuk meg a beépített `DSolve` vagy `NDSolve` differenciálegyenlet-megoldóval [36]. Hasonló megoldás minden más matematikai szoftverrel is adható, az ilyen szoftverek alkalmazásának előnye, hogy magunk készíthetünk hozzájuk kiegészítő csomagokat; végül én is egy ilyen implementáció mellett döntöttem. Az 5.1. táblázat néhány komolyabb, napjainkban is hozzáférhető, karbantartott, ingyenes (illetve demo változatban elérhető fizetős) program jellemzőit mutatja be.

5.2. Formai követelmények, általános megfontolások

Az előző alfejezetben leírt tapasztalatok alapján az elkészítendő programmal szemben támasztott legfontosabb követelmények az alábbiak voltak:

- a program legyen általános célú, ne speciális részterületre koncentráljon,
- a felhasználónak használhasson kémiai (nem mátrixos) formalizmust is,
- a felhasználónak alkalmazhasson kiegészítő feltételeket, esetleg saját algoritmusokat is,
- a program legyen platformfüggetlen,
- legyen jól dokumentált, (mások által is) továbbfejleszthető.

A programcsomag tervezett funkciói négy fő modulra oszthatók:

- bruttó reakciók felbontása elemi reakciólépésekre,
- reakciók sztochasztikus szimulációja,
- a kinetikai differenciálegyenletek felírása és megoldása,
- a kinetikai differenciálegyenletek megoldásainak kvalitatív jellemzésének támogatása.

5.3. Implementációs kérdések

Az előző alfejezetben ismertetett követelményeket és az implementálandó algoritmusok matematikai részleteit számba véve egy *Mathematica* csomag (*add-on package*) készítése

a program neve	platform	nyelv	általános célú	kémiai felület	megjelenítés	speciális funkciók	ár
BioKMod [26]	platform-független	*Mathematica*	nem	nincs	a futtatókörnyezetben	—	ingyenes
ChemSimul [14]	Windows, egyes Unixok	FORTRAN	nem	van	van	—	1000 USD
CRNT [9]	DOS	N/A	igen	igen	van	kvalitatív vizsgálat	ingyenes
KinAl [31]	DOS	FORTRAN	igen	van	nincs	érzékenységi analízis	ingyenes

5.1. táblázat. Reakciókinetikai programcsomagok

mellett döntöttem. Ennek a megvalósításnak számos előnye közül is kiemelendőek az alábbiak:

- jól olvasható, tömör, könnyen áttekinthető kód,
- rugalmas adatszerkezetek,
- matematikai szemléletmód, hatékony funkcionális programozási elemek,
- nagyteljesítményű beépített nagyszám-aritmetika, pontos egész és racionális aritmetika,
- nagyteljesítményű beépített szimbolikus és numerikus függvények,
- integrált lineáris programozási rutin,
- kiváló grafikai képességek, két- és háromdimenziós ábrák készítésében egyaránt,
- platformfüggetlenség.

A pontos egész és racionális aritmetika azért is kiemelendő, mert nagy méretű egyenletek esetén a numerikus hibák (igaz, nagyon ritkán) hibás végeredményre vezettek. A választott környezet főbb hátrányai:

- költséges (bár kutatóintézetekben és akadémiai környezetben igen elterjedt),
- nem nyílt forráskódú, néha körülményes a terméktámogatás,
- az elkészített programok nem fordíthatók, csak interpretáltan futtathatók.

5.3.1. Adatstruktúrák, adattípusok

A választott környezet nem támogatja a magas szintű objektum-orientált programozást, erre – sem a feladat jellegét, sem pedig a megcélzott felhasználói réteget figyelembe véve – nincs is szükség. A *Mathematica* funkcionális programozási felülete ezzel szemben meglehetősen kényelmes lehet mindazok számára, akik a csomag mögött álló matematikai és kémiai formalizmusban jártasabbak, mint a „konvencionális” programozásban. Ennek megfelelően speciális adatstruktúrákat sem alkalmaztam, a *Mathematica* belső adatábrázolásához illeszkedve listát, illetve listák listáját használtam vektorok illetve mátrixok megadására (mátrixok esetén sorfolytonosan). A formális kémiai reakciók adatait (α, β és γ mátrixok, résztvevő anyagok stb.) is egy listában fogjuk össze, erről bővebben lásd a listát előállító `ToStoichiometry` függvény leírását az 5.4.4. szakaszban.

Az anyagneveket karakterláncok formájában adjuk meg, a kémiai reakciókban szereplő együtthatók szimbolikus egészek.

5.4. A megvalósított függvények

Az elkészült függvények interfészét az egyszerűség kedvéért az alkalmazott *Mathematica* környezet formalizmusával adtam meg; a meglehetősen beszédes, bár elsőre talán szokatlan jelöléseket a programcsomag nyelvét nem ismerő olvasó is megértheti.[2] Ezenkívül pontosan definiáltam az egyes függvények szemantikáját is. A *Mathematica* konvencióinak megfelelően minden függvény „beszédes nevet" kapott, és minden függvényhez elkészítettem a használatát segítő leírását (*usage messages*). Ezért, illetve terjedelmi korlátok miatt ebben a fejezetben csak az elkészített függvények felsorolása és nagyon tömör leírása olvasható. A megvalósítás részletezése helyett többnyire a 3. és a 4. fejezet megfelelő pontjaira hivatkozom, az ott ismertetett algoritmusok és programvázlatok alapján a *Mathematica* kód a legtöbb esetben nagyobb nehézség nélkül elkészíthető.

5.4.1. Segédfüggvények

A könnyebb kezelhetőség, illetve a programok robosztusabbá tétele érdekében (a függvények bemenő paramétereinek ellenőrzéséhez), számos segédfüggvényt definiáltam. A `Q` (*question*) végű függvények a *Mathematica* konvencióinak megfelelően a bemenő paraméter (minden esetben egy lista) valamely tulajdonságát („típusát", halmazba tartozását) ellenőrzik, logikai igaz/hamis értéket adva vissza.

a függvény neve	az ellenőrzött tulajdonság
`DecompositionQ`	az inputlista egy felbontást jelöl (nemnegatív egész vektor)
`GeneralizedDecompositionQ`	az inputvektor egy pozitív egész számszorosa felbontás (nemnegatív racionális vektor)
`ElementaryReactionQ`	az inputlista egy elemi reakció vektorát jelöli
`ReactionQ`	az inputlista egy reakció vektorát jelöli
`WeaklyReversibleQ`	az inputlista egy gyengén megfordítható összetett reakció adatainak listája (lásd az 5.4.4. szakaszban a `ToStoichiometry` függvény leírását)

A reakciókat jelölő vektorok egész számokból álló vektorok, az elemi reakciók vektoraiban a negatív komponensek összege -1 vagy -2 lehet csak.

A `WeaklyReversibleQ` függvény a kémiai mechanizmus gyenge megfordíthatóságát úgy ellenőrzi, hogy meghatározza a Feinberg–Horn–Jackson-gráf (39. oldal) erősen összefüggő komponenseit, majd ellenőrzi, hogy nincs-e a gráfnak olyan éle, amelynek vég-

[2] Rendkívül kompakt (nagyrészt jelenleg is aktuális) bevezető a szoftver használatába a [19] cikk, bővebb ismertetést tartalmaz a [29] könyv. A program hivatalos (a világhálón is elérhető) dokumentációja [35] minden részletre kiterjed, a jelölésekkel ismerkedő olvasónak ennek 2. fejezetével érdemes kezdenie.

pontjai különböző komponensekben vannak. A gráf éllistás ábrázolása esetén ez az algoritmus lineáris futásidejű, ha az erősen összefüggő komponenseket mélységi bejárásokkal határozzuk meg [25, 6.4.3. szakasz].

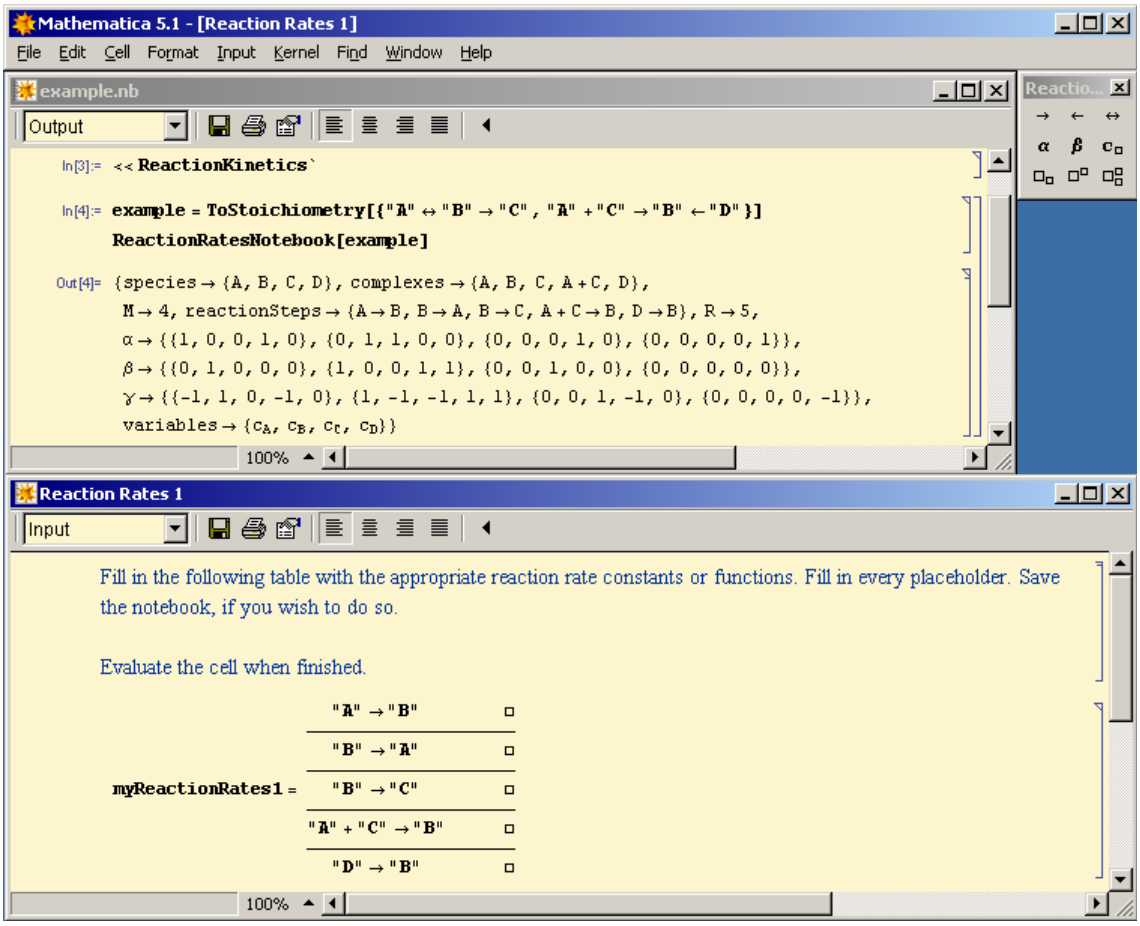
5.4.2. Ki- és beviteli függvények

További segédfüggvények segítik a felhasználót az adatok bevitelében és az eredmények formázott kivitelében. A felhasználó számára természetesebb, ezért könnyebben visszaolvasható és ellenőrizhető az adatbevitel, ha a reakciólépéseket a felhasználó a hagyományos kémiai jelöléseket használva adhatja meg, és nem kell a dolgozatban használt vektoros/mátrixos matematikai formalizmust alkalmaznia. Hasonlóképpen, a mátrixos formában kapott eredményt is érdemes újra olvasható formába öntenünk. Ennek megkönnyítésére szolgálnak az alábbi függvények:

a függvény neve	a függvény feladata
`FromStep`	kémiai jelölésekkel adott reakció vektorrá alakítása
`ToStep`	vektor kémiai jelölésekkel adott reakcióvá alakítása
`PrintDecompositions`	bruttó reakció felbontásának szép megjelenítése, lásd az 5.4.3. szakaszban
`ToStoichiometry`	a formális kémiai mechanizmus legfontosabb jellemzőinek előállítása, lásd az 5.4.4. szakaszban
`ReactionRatesNotebook`	a sebességi állandók bevitelének támogatása, részletesen lásd alább

A felhasználó teljes szabadságot élvez egy kémiai mechanizmus reakcióinak megadásakor, a csomag értelmezi a jobbra, balra és oda-vissza mutató nyilakból felépített reakcióláncok listáját is, egy mechanizmus megadható például így: {A⇌B→C, A+C→B←D}. Az így olvashatóbbá tett adatok bevitelét segítendő a csomag betöltésekor egy paletta nyílik meg a formázáshoz és adatbevitelhez legszükségesebb gombokkal (nyilak, alsó-felső indexek stb.); ez az 5.1. ábra képernyőképén is látható.

Amikor a mechanizmushoz tartozó (4.1) differenciálegyenlet vagy a (4.2) egyenletekkel definiált infinitezimális generátor felírásához szükséges sebességi állandókat kell megadnunk – például a kinetikai egyenletek megoldására szolgáló függvény (5.4.5. szakasz) vagy a szimulációs függvény (5.4.6. szakasz) alkalmazásakor –, csak egy közönséges szám- vagy szimbólumlistát adhatunk meg, amely a sebességi együtthatókat a reakciólépések sorrendjében kell, hogy tartalmazza. A helyes sorrendet a vizsgált kémiai mechanizmus főbb jellemzőinek adatait előállító `ToStoichiometry` függvény szabja meg (ennek részletes leírását lásd az 5.4.4. szakaszban): az ennek kimenetében szereplő sorrend (`reactionSteps` paraméter) a megfelelő. Sok reakciólépés esetén a felhasználó



5.1. ábra. `ReactionRatesNotebook` példa

számára igen körülményes lehet közvetlenül ezt a reakciólistát böngészve megadni az együtthatókat, ez könnyen hibához is vezethet. A sebességi együtthatók kényelmesebb megadásához ezért a csomag egy külön felületet is ad: a `ToStoichiometry` függvény kimenetét a `ReactionRatesNotebook` függvénynek átadva egy új *Mathematica* notebook nyílik meg, amely táblázatosan tartalmazza a reakciókat. A felhasználónak csak a reakciók mellé kell írnia a sebességi állandókat, és a sebességi állandókat megfelelő sorrendben tartalmazó listát kap eredményül. Ez elsősorban nagy méretű – több tíz, száz vagy akár ezer reakcióból álló – mechanizmus esetén jelent könnyebbséget. A függvény használata közben mentett képernyőképet mutat az 5.1. ábra.

5.4.3. Felbontások előállítása

Amint azt a 2.4. szakaszban láttuk, egy megadott bruttó reakció felbontásainak előállításához előbb az elemi reakciókat kell meghatároznunk, majd a kapott elemi reakciók (vagy egy részük) segítségével kell előállítanunk a felbontásokat.

Az elemi reakciók előállításának feladatát egyetlen függvénnyel (`ElementaryReactions`) megvalósíthatjuk, ennek első argumentuma az atommátrix (amelynek a *sorai* felelnek meg az anyagfajtáknak, ahogyan azt például a 3.1. táblázatban láthatjuk). Második, opcionális argumentuma a reakciók termékeinek maximális száma (alapértelmezés szerint végtelen).

```
ElementaryReactions[atommatrix_?MatrixQ,
                    maxlen : (_Integer?Positive | Infinity) : Infinity,
                    opts___?OptionQ]
```

A függvény működését a következő táblázatban összefoglalt opciókkal befolyásolhatjuk:

opció neve	opció hatása	lehetséges értékek	alapértelmezés
`Method`	a megoldásokat előállító algoritmus kiválasztása	`ContejeanDevie`, `LPBased`, `MartinezSilva`	`ContejeanDevie`
`Verbose`	engedélyezi, ill. tiltja a futás közbeni részeredmények kijelzését	`True`, `False`	`True`

A termékek számára vonatkozó korlát (`maxlen` argumentum) mindhárom algoritmus esetén figyelembe vehető úgy, hogy a felbontandó vektort egy új, `maxlen` értékű komponenssel, a particionáló vektorokat pedig egy 1 értékű komponenssel bővítjük, végül egy további, szintén egy 1 értékű komponenssel kiegészített nullvektort veszünk a particionáló vektorokhoz. (Utóbbi ahhoz kell, hogy ne csak a pontosan `maxlen` termékű reakciókat kapjuk meg.) Az így kapott feladatot megoldva az utolsó particionáló vektorhoz tartozó együtthatókat eldobjuk, így kapjuk az eredeti feladat hosszkorlátos megoldásait. Ez a megoldás a fabejáráson alapuló algoritmusok (a naiv rekurzív algoritmus és a Contejean–Devie-algoritmus) esetén ekvivalens azzal, hogy a keresési tér bejárásakor nem lépünk a keresési fa `maxlen`-nél mélyebb szintjeire. A Contejean–Devie-algoritmus esetén a 3.3. tétel alkalmazásával további javulást érhetünk el, ezt a megoldást is beépítettem a függvénybe.

A felbontásokat előállító függvény paraméterei rendre: a bruttó reakció vektora, az elemi reakciók vektorainak listája, valamint (opcionálisan) a felbontások maximális mérete. A függvény működését befolyásoló opciók megegyeznek az `ElementaryReactions` függvény opcióival, ezeken kívül egy további, alapértelmezés szerint `True` értékű `Preprocess` opcióval választhatunk, hogy alkalmazzuk-e a 3.5. alfejezetben ismertetett előfeldolgozási algoritmusokat.

```
Decompositions[overall_?ReactionQ,
               steps : __?ReactionQ,
               maxlen : (_Integer?Positive | Infinity) : Infinity,
               opts___?OptionQ]
```

Az előfeldolgozási lépéseket külön, a felbontások előállítása nélkül is elvégezhetjük. Az `Obligatory` függvény a minden felbontásban szereplő reakciólépéseket, míg az `Omittable` függvény az egyetlen felbontásban sem szereplő reakciólépéseket adja vissza. Az `Obligatory` függvény egy párokból álló listával tér vissza, a kiválasztott lépésekkel együtt azok minimális együtthatóját is megadja. Mindkét függvénynek két argumentuma van: az elemi lépések listája (vektoros formában) és a bruttó reakció vektora.

A felbontások nem szisztematikus előállítása

A felbontások 3.6. szakaszban ismertetett nem szisztematikus előállítását (3.5. algoritmus) a `CoveringDecompositionSet` függvénnyel végezhetjük, ennek használata megegyezik a `Decompositions` függvényével, kivéve hogy annak opciói közül értelemszerűen csak a `Verbose` használható, egy további opcióval (`ObjectiveFunction`) pedig a (3.9) lineáris programozási feladat célfüggvényét módosíthatjuk három előre beállított $\mathbf{c}$ vektorral. Az opció lehetséges értékei és hatásuk a következő táblázatban olvasható:

az opció értéke	$\mathbf{c}^{\mathrm{T}}$ értéke (3.9)-ban
`OriginalSelection`	az M halmaz karakterisztikus vektorának komplemense
`GreedySelection`	az M halmaz karakterisztikus vektora
`FastSelection`	nullvektor

Az első variáció a 3.5. algoritmus eredeti változata. A célfüggvény ilyen megválasztását az motiválja, hogy mohó módon a lehető legkevesebb jelöletlen – vagyis a lehető legtöbb jelölt – vektort igyekszünk kiválasztani minden iterációban, ezzel az iterációk száma – és így az előállított felbontások száma is – megnő. Ha az algoritmust minél több felbontás gyors generálására kívánjuk használni, ez a választás lehet a legcélszerűbb. A második változatot az előző érveléshez hasonló gondolatmenet alapján akkor célszerű választani, ha az algoritmust előfeldolgozási lépésként alkalmazzuk. (Természetesen az előbbiek csak heurisztikus állítások és érvelések, nem tételek és bizonyítások.) A harmadik választást a gyorsasága indokolja. Ilyenkor nem optimalizálunk, csupán a lineáris program egy megengedett megoldását kell megtalálnunk. Ha a lehető legtöbb felbontásra van szükségünk, a legbiztosabb mindhárom célfüggvénnyel lefuttatni az algoritmust.

Indexezés

A 4.1.1. szakaszban ismertett Volpert-indexezést a `VolpertIndexing` függvény végzi. E függvény paraméterezése egyrészt attól függ, hogy felbontást indexezünk-e (utólagos indexezés), vagy pedig reakciólistát (előfeldolgozási lépésként, ahogyan azt a 3.5.3. szakaszban tettük), másrészt pedig attól, hogy milyen formában kívánjuk megkapni az eredményt. A következő négy lehetőségünk van:

```
VolpertIndexing[decomp_?DecompositionQ, atomm_?MatrixQ, initial_List]
VolpertIndexing[decomp_?DecompositionQ, atomm_?MatrixQ, initial_List,
                                        species_List, reactionsigns_List]
VolpertIndexing[reactions : __?ReactionQ, initial_List]
VolpertIndexing[reactions : __?ReactionQ, initial_List,
                                        species_List, reactionsigns_List]
```

Az első két változatot, amint azt az argumentumok típusai is jelzik, felbontások indexezésére használhatjuk. Meg kell adnunk a felbontást, az atommátrixot (ugyanúgy, mint a felbontásokat előállító függvények esetében), valamint a kezdeti anyagok sorszámainak listáját. (A sorrendet az atommátrix sorai adják meg.) Az első esetben a kimenet a reakciók és az anyagfajták Volpert-indexeiből álló listapár. A második változatban megadjuk az anyagfajták neveit, valamint a reakciók neveit. Ilyenkor a függvény jobban olvasható, táblázatos formában jeleníti meg az eredményt.

Ugyanez a különbség a harmadik és negyedik változat között, ezeket azonban előfeldolgozási lépésként használhatjuk. Atommátrixra nincs szükség, csupán a reakciók vektorait, valamint a kezdeti anyagfajták listáját kell megadnunk.

Ha csupán arra vagyunk kíváncsiak, hogy indexezhető-e a felbontás, illetve a reakciólista, az `Indexable` függvényt használhatjuk. Értelemszerűen úgy paraméterezzük, mint az előbbi függvényt az első, illetve a harmadik esetben. A visszatérési érték egy igaz/hamis logikai érték.

Két további, a felbontások előállításával kapcsolatos függvény a `SelectMinimalDecompositions`, amely a megtalált felbontások listájából (ez a bemenő paraméter) kiválogatja a minimálisakat, valamint a már említett `PrintDecomposition`, amelyet kétféleképpen paraméterezhetünk:

```
PrintDecomposition[decomp_?DecompositionQ, reactions_List]
PrintDecomposition[decomp_?DecompositionQ, reactionlist_?MatrixQ, species_List]
```

Az első esetben a felbontás mellett a reakciók megnevezését (a megjelenítendő karakterláncot) kell megadnunk, a második esetben pedig a reakciók vektorainak listáját, valamint az anyagfajták neveit. Ekkor a reakciókat jelölő karakterláncokat a függvény összeállítja az anyagfajták neveiből és a reakciók vektoraiból, az 5.4.2. szakaszban már látott `ToStep` függvény segítségével.

5.4.4. Kvalitatív vizsgálatok

A kvalitatív vizsgálathoz elsődlegesen szükséges függvény a `ToStoichiometry` függvény, ez a reakciók listájából állítja elő a kémiai mechanizmus mindazon jellemző paramétereit (*Mathematica* helyettesítési szabályok formájában, pl. `species` $\rightarrow$ {H_2,O_2}), amelyek a reakciólépésekből közvetlenül – komolyabb számítások nélkül – leolvashatók. Ezeket a következő táblázat foglalja össze, a függvény bemenő paramétereit pedig az alábbi függvénydefiníció mutatja.

```
ToStoichiometry[reactionlist_List, externalspecies_List : {}]
```

paraméter neve	jelentése
`species`	az anyagfajták listája
`complexes`	a komplexek listája
`M`	a résztvevő anyagok száma
`reactionSteps`	a reakciólépések listája
`R`	a reakciólépések száma
α	a kémiai mechanizmus α mátrixa
β	a kémiai mechanizmus β mátrixa
γ	a kémiai mechanizmus γ mátrixa
`variables`	az anyagok koncentrációját jelölő változók

A `ToStoichiometry` függvény első bemenő paramétere a reakciók listája, ennek megadásakor a felhasználó a kémiai környezetben szokásos jelöléseket használhatja, azaz balra és jobbra mutató, valamint kettős nyilak kombinációjával tetszőleges láncokat is képezhet. A kimenő listában már kanonikus alakban olvashatók a reakciók, azaz minden reakció $C_1 \rightarrow C_2$ alakú; láncok, fordított irányú reakciók vagy oda-vissza nyíllal ábrázolt lépéspárok nincsenek. Anyagnévként tetszőleges *Mathematica* karakterlánc elfogadható, az X anyag koncentrációjának jelzésére a függvény egy c_X alakú változót vesz fel, ezek a változók kerülnek a `variables` listába. A második, opcionális paraméterben a külső, állandó koncentrációjúnak tekintett anyagokat sorolhatjuk fel. Ezek nem kerülnek az anyagok listájába (így az α, β és γ mátrixokba sem). A `"0"`-val jelölt üres komplex automatikusan állandó koncentrációjúnak tekintett anyagfajta.

A `ToStoichiometry` függvény kimeneteként kapott lista szolgál a kémiai mechanizmusok kvalitatív és kvantitatív tulajdonságait vizsgáló minden további függvény bemenő paraméteréül. Ebben a szakaszban csak a dolgozat 4.1.1. szakaszában ismertetett kvalitatív tulajdonságokkal foglalkozunk, a kvantitatív vizsgálatokhoz szükséges függvényeket a következő szakasz mutatja be.

A felbontások indexezésére szolgáló függvényeket az előző szakaszban már bemutattuk. A Feinberg–Horn–Jackson-gráfhoz kapcsolódó függvényeket is implementáltam. A mechanizmus deficienciáját a `Deficiency` függvény adja meg, a Feinberg–Horn–Jackson-gráfot pedig az `FHJ` függvény állítja elő. Ez utóbbi kimenete a *Mathematica* `Combinatorica` csomagjában használt `Graph` adatstruktúrával ábrázolt gráfot ad vissza. Az erősen összefüggő komponenseket (az egy komponensben található komplexek neveinek listáit tartalmazó listában) az `FHJComponents` függvénnyel kaphatjuk meg. A Feinberg–Horn–Jackson-gráfot ábrázolhatjuk is a `ShowFHJ` függvény segítségével.

5.4.5. A kinetikai egyenletek megoldása

A (4.1) kinetikai egyenletek megoldására a `Concentrations` függvény szolgál:

```
Concentrations[tst_List, reactionrates_?VectorQ, initialvalues_?VectorQ,
                                                              opts___?OptionQ]
Concentrations[tst_List, reactionrates_?VectorQ, initialvalues_?VectorQ,
                                          endtime_?NumericQ, opts___?OptionQ]
```

Az első paraméter ezúttal is a `ToStoichiometry` függvénnyel kapott paraméterlista, a második a sebességi együtthatók listája (abban a sorrendben, ahogyan a reakciólépések a `ToStoichiometry` kimenetében állnak, lásd a `ReactionRatesNotebook` függvény leírását). Ha az `endtime` paramétert nem adjuk meg (első sor), akkor a függvény szimbolikusan kísérli meg megoldani a kinetikai differenciálegyenleteket. Ez általában reménytelen, az eljárás sikere a *Mathematica* `DSolve` függvényébe beépített differenciálegyenlet-megoldó rutinoktól függ. Ha megadjuk az `endtime` paramétert (második sor), akkor numerikus megoldást kapunk a $[0, \text{endtime}]$ intervallumban, a *Mathematica* `NDSolve` függvényének felhasználásával.

Mindkét esetben a *Mathematica* megfelelő függvényének saját opcióit használva irányíthatjuk a megoldást, ezeket itt nem ismertetjük, részletes leírás olvasható a szoftver dokumentációjában [35]. A szimbolikus megoldást választva a reakciólépések sebességi állandóit és a kezdeti koncentrációkat is megadhatjuk szimbolikusan, a numerikus megoldást választva azonban nem.

A csomag nem tartalmaz a megoldásokat vizualizáló függvényt, valóban általánosan használható megjelenítő készítése szinte reménytelen feladat. A koncentrációk a vizsgált intervallum egyes szakaszain több nagyságrenddel eltérhetnek, és nem is lehet mindig szükség az összes anyagfajta koncentráció–idő függvényének egyidejű megjelenítésére. A felhasználó azonban a *Mathematica* beépített függvényrajzoló utasításaival (`Plot` stb.) természetesen felrajzolhatja a kapott megoldásokat.

A kinetikai differenciálegyenlet jobb oldalát az egyenlet megoldása nélkül is felírhatjuk, erre a `RightHandSide` függvény szolgál. Ennek csak a `Concentrations` függvény első két argumentumát kell átadnunk, szimbolikus paramétereket is elfogad.

A megoldások stacionárius pontjainak halmazát a `StationaryPoints` függvénnyel határozhatjuk meg, szimbolikusan vagy numerikusan, ennek paraméterezése megegyezik a `RightHandSide` függvényével.

```
RightHandSide[tst_List, reactionrates_?VectorQ]
StationaryPoints[tst_List, reactionrates_?VectorQ]
```

A függvény a (4.1) differenciálegyenlet-rendszer jobb oldalának zérushelyeit határozza meg szimbolikusan (ha minden paraméter szimbolikus egész, racionális szám vagy változó) vagy pedig numerikusan (ha lebegőpontos szám is szerepel a paraméterek között).

5.4.6. Sztochasztikus szimuláció

A szimulációt a 4.2.1. szakaszban ismertetett 4.3. algoritmussal valósítottam meg. A csomagban implementált függvény bemenő paramétereit az alábbi függvénydefiníció mutatja.

```
StochasticSimulation[tst_List, rrates_List, initialstate_List, endtime_?NumericQ]
```

Az első paraméter a formális kémiai mechanizmus adatait tartalmazó, a `ToStoichiometry` függvény által előállított lista. Kimenetként a 4.3. algoritmus programvázlatánál látható módon egy időpont- és egy állapotlistát ad vissza. Mivel a szimuláció eredménye az is lehet, hogy a rendszer „felrobban”, azaz egy anyag mennyisége végtelenhez tart, amikor is a reakciólépések egyre gyorsuló ütemben követik egymást, a szimuláció könnyebb nyomonkövethetősége érdekében a szimuláció rendszeresen kiírja a szimulációs időt és az aktuális állapotot.

A szimuláció eredményének megjelenítésére a csomag nem tartalmaz beépített függvényt, mivel – a kinetikai egyenletek megoldásainak megjelenítéséhez hasonlóan – a szimulációs eredmények robosztus és praktikus vizualizációja is rendkívül nehéz. A felhasználó azonban a *Mathematica* beépített rajzoló utasításaival (`ListPlot`, `MultipleListPlot` stb.) természetesen felrajzolhatja a kapott megoldásokat.

6. fejezet

Alkalmazási példák

6.1. Elemi reakciók meghatározása

A 3.7. alfejezetben bemutatott három reakciókinetikai feladatot az elkészített programcsomaggal oldottam meg. Elsőként mindhárom feladat esetén az elemi reakciókat határoztam meg. Amint láttuk, ehhez az összes lehetséges bal oldalhoz (reaktánskombinációhoz) meg kell keresnünk az összes lehetséges jobb oldalt (termékkombinációt); az előző fejezetben ismertetett programcsomagban erre az `ElementaryReactions` függvény szolgál.

A permanganát–oxálsav reakcióban 19 anyagfajta van, ezért $2 \cdot 19 + \binom{19}{2} = 209$ egyenletrendszert kell megoldanunk, amelyek rendre 17, illetve 18 változósak attól függően, hogy a vizsgált reakciónak két vagy csak egy reaktánsa van-e. A 6.1. táblázat mutatja, hogy az ismertetett algoritmusokkal mennyi ideig tart az elemi reakciók előállítása ebben a feladatban. A Martínez–Silva-algoritmus nem található meg a táblázatban. Ennek az az oka, hogy az csupán a minimális tartójú megoldásokat állítja elő – ez ebben a feladatban kevés. A 3.2. lemma – inhomogén egyenletrendszerekről lévén szó – nem alkalmazható az összes megoldás, vagy az összes minimális megoldás előállítására. Ezzel szemben az egyenletek és a megoldások viszonylag kis mérete miatt – lényegében bármelyik másik ismertetett módszer hatékony. Az egy reaktánssal rendelkező elemi reakciólépésekhez vezető egyenletek megoldásakor a naiv rekurzív algoritmus bizonyult a leghatékonyabbnak, de a legnagyobb megoldásokkal rendelkező kétreaktánsú lépéseknél mind ezt, mind pedig a Contejean–Devie-algoritmust túlszárnyalja az LP-alapú leszámlálás. A táblázatban foglalt eredmény ezzel együtt is csalóka: a legkisebb egyenletek esetén az LP-alapú leszámlálás igen kevéssé hatékony. Minden módszerrel összesen 1022 megoldást találtam.

A második, levegőkémiai példában azokat az elemi reakciókat kerestük, amelyekben szerepel ként tartalmazó anyag. A 80 anyagfajtából 24 ilyen, így $2 \cdot 24 + \binom{24}{2} + 24 \cdot 56 = 1668$

egyenletrendszert kell megoldanunk, melyek 78 illetve 79 változósak. A futásidő minden algoritmussal meghaladta a nyolc órát. A részeredmények ismeretében is világossá vált, hogy a hatalmas (négyszázezret meghaladó) számú megoldás a további – e dolgozat keretén túlmutató – vizsgálódások során kezelhetetlen, ezért egy további korlátozó feltételt szabtunk a megoldásokra: csak azokat a megoldásokat kerestük, amelyeknek maximum három terméke van.

Ez a módosítás egyértelműen a fabejáró algoritmusoknak kedvez, nem meglepő, hogy az LP-alapú leszámlálás alulmarad a többi algoritmussal szemben. A 6.1. táblázatban foglalt eredmények[1] ugyanakkor azt is mutatják, hogy a Contejean–Devie-algoritmus a 3.3. tételben javasolt módosítással sokkal hatékonyabb, mint anélkül.

A harmadik példában (oxalát–perszulfát–ezüst oszcillátor) $2 \cdot 16 + \binom{16}{2} = 152$ egyenletrendszert kell megoldanunk. Meglehetősen kis méretű feladatról lévén szó minden algoritmussal gyorsan célhoz érünk, a naiv rekurzív algoritmus is kimondottan jól szerepel. Az eredmények a 6.1. táblázatban olvashatók.

Mindhárom példa jól mutatja, hogy a Contejean–Devie módszer lényegesen kevésbé hatékony módja az elemi reakciók meghatározásának, mint a két másik ismertetett eljárás. Az egyszerű rekurzív eljárás a legjobb, amíg a felbontandó vektor és a megoldások száma kicsiny, ekkor azonban még minden eljárás elég gyors. A nagyobb jobboldalú és sok megoldással rendelkező egyenletek esetén az LP-alapú leszámlálás alkalmazása látszik a legcélravezetőbbnek.

Probléma	Feladatok száma	Egyenletek/ változók	Algoritmus	CPU idő (s)
Permanganát–oxálsav	209	5 / 17	naiv BFS	163.04
			Contejean–Devie	1589.48
			LP-alapú leszámlálás	51.66
Levegőkémia	1668	5 / 78	naiv BFS	$>$ 8 óra
			módosított C–D	13429.5
			eredeti C–D	$>$ 8 óra
			LP-alapú leszámlálás	$>$ 8 óra
Oxalát–perszulfát–ezüst	152	6 / 14	naiv BFS	3.10
			Contejean–Devie	116.67
			LP-alapú leszámlálás	2.00

6.1. táblázat. Elemi reakciók előállításának feladatán végzett mérési eredmények

[1] Az időméréseket egy AMD Athlon XP 1.6 GHz CPU, 768 MB RAM asztali PC-n végeztem, Windows XP alatt futó *Mathematica* 5-tel.

6.2. Bruttó reakciók felbontása

A bruttó reakciók felbontásainak megkeresésére szolgáló módszereket a permanganát–oxálsav reakció példáján végzett méréseken keresztül mutatjuk be. Az előző alfejezetben előállt az az 1022 elemi reakció, amely esetleg végbemehet a bruttó reakció során. Ezek egy része kémiailag értelmetlen. Megfelelő adatok híján termodinamikai szempontok alapján nem tudunk számítógép segítségével szűrni, ehelyett egy vegyész kolléga közreműködésével húztunk ki a reakciók közül 349-et. A fennmaradó 673 termodinamikailag lehetséges reakció már a dolgozatban ismertetett módszerekkel vizsgálható tovább.

Elsőként a 3.5.1. szakaszban bemutatott módszerrel megállapítottam, hogy két reakciólépés minden felbontás során végbemegy. A $H_2C_2O_4 + MnO_2 \rightarrow [MnO_2, H_2C_2O_4]$ reakciólépés legalább négyszer, a $2\,CO_2^- \rightarrow C_2O_4^{2-}$ reakciólépés pedig legalább egyszer végrehajtódik. Ezzel a megoldások mérete öttel csökkenthető, ami elsősorban a Contejean–Devie-algoritmussal, illetve a naiv rekurzív algoritmussal végzett keresés esetén jelentős előny. A minden felbontásban szereplő reakciólépések megkeresése alig 2 percet vett igénybe.

Mindkét, a 3.5. alfejezetben ismertetett módszerrel kerestem nem végbemenő reakciókat és elő nem állítható anyagokat. Az első, lineáris programozáson alapuló algoritmus (3.5.2. szakasz) további kémiai feltételek nélkül 314 lépést talált, amelyek semmiképpen nem mehetnek végbe. Megvizsgáltam azt is, milyen hatása van annak, ha a (3.9) lineáris programban megváltoztatjuk a célfüggvényt. A várakozásoknak megfelelően kevesebb iterációra volt szükség, amikor a $\mathbf{c}^{\mathrm{T}}\mathbf{x}$ kifejezés helyett az $(\mathbf{1} - \mathbf{c}^{\mathrm{T}})\mathbf{x}$ kifejezést minimalizáltuk minden lépésben. (313 helyett 280 lineáris programozási feladatot kell megoldanunk.) Végül 359 reakció marad. Az eljárások futásideje 5–8 perc volt.

A megmaradt reakciókat Volpert algoritmusával (3.5.3., illetve 4.1.1. szakasz) indexeztem, különböző kezdetianyag-kombinációk mellett. A [15] publikációban szereplő öt kezdeti anyag feltételezése esetén pontosan ugyanaz a 297 reakciólépés (és csak 16 anyagfajta) indexezhető, mintha az összes nem komplex anyagot a kezdeti anyagok közé választjuk. Sőt, pontosan ugyanezeket a reakciókat kapjuk akkor is, ha a legelső, kémiai alapon történő szűrés után megmaradt reakciólépéseket indexezzük. Az indexezés minden esetben a másodperc tört része alatt végbement.

Az előfeldolgozási lépések után megmaradt 297 reakcióval megkezdhető a bruttó reakció összes felbontásának meghatározása. A legrövidebb lépéssorozatok meghatározásához először lineáris programozással becsültem a megoldások méretét. (Erre szigorúan véve semmi szükség, de az eredmény gyorsan hozzávetőleges képet ad a megoldások bonyolultságáról.) Az egyes reakciólépések együtthatóit jelentő változók összegének minimumára (a (3.1) egyenlet, mint kényszerfeltétel mellett) 15 adódott, a talált optimum ráadásul minden változóhoz egész értéket rendelt, így azonnal egy minimális felbontást

$4\times$	$H_2C_2O_4 + MnO_2$	$\rightarrow$	$[MnO_2, H_2C_2O_4]$
$2\times$	$H_2C_2O_4 + [MnO_2, H_2C_2O_4]$	$\rightarrow$	$2CO_2 + 2H_2O + MnC_2O_4$
$1\times$	$H^+ + HC_2O_4^-$	$\rightarrow$	$H_2C_2O_4$
$1\times$	$C_2O_4^{2-} + H^+$	$\rightarrow$	$HC_2O_4^-$
$2\times$	$MnC_2O_4 + MnO_4^-$	$\rightarrow$	$CO_2^- + CO_2 + 2MnO_2$
$1\times$	$2CO_2^-$	$\rightarrow$	$C_2O_4^{2-}$
$2\times$	$H^+ + [MnO_2, H_2C_2O_4]$	$\rightarrow$	$[H^+, MnO_2, H_2C_2O_4]^+$
$2\times$	$H^+ + [H^+, MnO_2, H_2C_2O_4]^+$	$\rightarrow$	$2CO_2 + 2H_2O + Mn^{2+}$

6.2. táblázat. A permanganát–oxálsav reakció egy felbontása

is megkaptam (6.2. táblázat). Vegyük észre, hogy a felbontás valóban tartalmazza a két említett reakciólépést, a megfelelő együtthatókkal. A táblázat formátumát imitálja a csomag `PrintDecomposition` függvénye.

Érdemes megjegyeznünk, hogy az LP-alapú leszámlálás algoritmusának azon kiegészítése, amellyel a legfeljebb `maxlen` lépéses megoldásokat megkereshetjük, fél másodperc alatt felismeri, hogy nincsen legfeljebb 14 lépéses megoldás. A Contejean–Devie-algoritmus tárhatékony implementációjának ehhez 25 percre volt szüksége (a ciklus többszázezerszeri lefutásával) még azután is, hogy a biztosan végbemenő öt lépést kivontam az egyenletből.

A többi minimális felbontást is megkaphatjuk, ha a Contejean–Devie-algoritmus kiegészített változatát használjuk a legfeljebb 15 lépéses megoldások előállítására, vagy ha a másik két módszer valamelyikének módosított változatát alkalmazzuk a pontosan 15 lépéses megoldások megkeresésére. A biztosan végbemenő öt lépést itt is kivonhatjuk az egyenletből, ekkor természetesen 10 lépéses megoldásokat keresünk. 11 különböző 15 lépéses megoldás van, ezek természetesen mind minimálisak. Előállításuk az LP-alapú leszámlálással (ami ilyen méretű feladatoknál már lényegesen gyorsabb a másik két eljárásnál) másfél percig tart. További felbontásokat állíthatunk elő, ha egyesével növelve a lépésszámot előállítjuk a többlépéses megoldásokat. Itt már elérjük az LP-alapú leszámlálás korlátait is – a 18 lépéses felbontások előállítására nem volt elég nyolc óra sem.

A megoldás utolsó lépéseként a megtalált felbontások közül Volpert algoritmusával ki kell választanunk az indexezhetőeket. A 6.3. táblázatban látható az összes időeredmény. A harmadik oszlopban az egyes feladatok során megoldott lineáris programozási feladatok száma olvasható.

Feladat	CPU idő (s)	LP-k száma	Eredmények
Minden felbontásban szereplő reakciólépések	124.80	673	2 reakciólépés
Nem szisztematikus keresés $\mathbf{c}^{\mathrm{T}}\mathbf{x}$ célfüggvénnyel	329.19	313	185 felbontás
Nem szisztematikus keresés $(\mathbf{1}-\mathbf{c})^{\mathrm{T}}\mathbf{x}$ célfüggvénnyel	443.11	280	230 felbontás
Nem szisztematikus keresés 0 célfüggvénnyel	330.00	313	189 felbontás
Nem szisztematikus keresés összesen	1102.30	906	420 felbontás
Az előző 420 felbontás Volpert-indexezése	0.5	—	110 indexezhető felbontás
A $\leq$ 14 lépésből álló felbontások előállítása	0.40	1	0 felbontás
A $\leq$ 15 lépésből álló felbontások előállítása	656.56	648	11 felbontás
A $\leq$ 16 lépésből álló felbontások előállítása	1096.44	8160	235 felbontás
A $\leq$ 17 lépésből álló felbontások előállítása	13619.50	186515	3170 felbontás
Az előző 3170 felbontás Volpert-indexezése	8.28	—	1918 indexezhető felbontás

6.3. táblázat. A permanganát–oxálsav reakció felbontásán végzett mérési eredmények

7. fejezet

Összefoglalás

Diplomatervem elsődleges célja egy olyan programcsomag készítése volt, amely támogatja a reakciókinetikai modellek, illetve az ehhez kapcsolódó matematikai feladatok számítógépes megoldását. A dolgozat kiindulópontja az volt, hogy a reakciókinetikai vizsgálatok tárgyát képző kémiai mechanizmusok jól leírhatók az informatikai modellezésben elterjedt Petri-hálók segítségével, így a Petri-hálós modellek analízisére szolgáló módszerek adaptálása a reakciókinetika területén is sikerrel kecsegtet. (A két tudományterület „fordított irányú" összekapcsolása – például P-gráfok alkalmazása az informatikában – szintén aktív kutatási terület.)

Eredmények

A diplomaterv az irodalomkutatáson és a programfejlesztés eredményén kívül önálló algoritmikus eredményeket is tartalmaz. Ezek: a lineáris diofantoszi egyenletrendszerek megoldására szolgáló Contejean–Devie-algoritmus kiegészítése; a csak speciális esetekben alkalmazható egyszerű rekurzív algoritmus alkalmazhatóságának vizsgálata és kiterjesztése a vektorok átrendezésével; valamint egy lineáris programozáson alapuló algoritmus leírása, amely a fabejáráson alapuló algoritmusokkal ellentétben egyes nagy méretű megoldásokkal rendelkező egyenletek esetén is alkalmazható. További, kisebb önálló eredményeket tartalmaznak az indexezésről és a megoldások nem szisztematikus kereséséről szóló fejezetek is.

Az elkészült programcsomag támogatja a bruttó reakciók felbontását – ilyen általános célú, azaz nem csak speciális reakciócsaládokra koncentráló program mindeddig nem készült. Az adott anyagfajták között végbemenő elemi reakciólépések automatikus előállítására is most készült először program. A további számítógépesíthető reakciókinetikai vizsgálatok közül a csomag támogatja a kémiai mechanizmusok szimulációját, a kinetikai differenciálegyenletek megoldását és a megoldások kvalitatív jellemzését is.

Tapasztalatok, továbbfejlesztési lehetőségek

A programcsomag jól használható eszköznek bizonyult az elemi reakciólépések előállítására, a bruttó reakciók felbontásának kiegészítése az elemi lépések számítógépi előállításával hasznos és megvalósítható ötletnek bizonyult. Az irodalomban vizsgált reakciók esetében a felbontások keresésekor figyelembe vett reakciólépések száma legalább egy nagyságrenddel kisebb volt, mint ahány reakciólépést az elkészült programmal előállíthatunk.

A programcsomag nem tartalmaz grafikus megjelenítéssel kapcsolatos függvényeket, a felhasználó a *Mathematica* környezet rajzoló utasításaival azonban megjelenítheti a numerikus számítási és szimulációs eredményeket. Érdekes, ám nehéz feladat a szimuláció eredményeinek, illetve a kinetikai differenciálegyenlet megoldásának a sokatmondó, szemléletes vizualizációja.

A kinetikai egyenletek megoldásainak kvalitatív jellemzése is bővíthető további jellemzők tesztelésével. Ilyenek például a konzervativitás ellenőrzése, illetve további, a periodikus vagy stabilis megoldások létezését vagy nem létezését garantáló feltételek tesztelése.

Ehhez kapcsolódó érdekes elvi kérdés az informatikai és a reakciókinetikai fogalomrendszer közötti párhuzamok mélyebb felderítése. A dolgozatban mindvégig azonos módon kezelt T- és P-invariánsok dualitásának reakciókinetikai értelmezése egyáltalán nem kézenfekvő. Az absztraktabb Petri-hálós, illetve matematikai megfogalmazásban a helyek és átmenetek halmaza (azaz a páros gráf két pontosztálya) teljesen szimmetrikus, a reakciókinetikában nem. Nem világos továbbá az sem, hogy a T-invariánsok a kinetikai differenciálegyenlet megoldásainak milyen tulajdonságát fejezik ki. A két terület között a másik irányban is kereshetünk szorosabb kapcsolatot: a reakciók CCD és CDS modelljének „majdnem-ekvivalenciájára” vonatkozó tétel független a kémiai interpretációtól, ám az informatikai alkalmazásokban csak a sztochasztikus Petri-hálók elterjedtek, a folytonos jelölésnek nincsen szemléletes, kombinatorikai jelentése. Minden bizonnyal tanulságos volna a 4. fejezet eredményeinek informatikai interpretációját megadni.

Irodalomjegyzék

[1] SCOTT AHLGREN, KEN ONO. Addition and Counting: the Arithmetic of Partitions, *Notices of the AMS* **48**(9):978–984, 2001.

[2] M. AJMONE MARSAN, GIANFRANCO BALBO, GIANNI CONTE, SUSANNA DONATELLI, AND GIULIANA FRANCESCHINIS. *Modelling with Generalized Stochastic Petri Nets*, Wiley Series in Parallel Computing, John Wiley and Sons, 1995. ISBN: 047 193 059 8

[3] BAZSA GYÖRGY (szerk.) *Nemlineáris dinamika és egzotikus kinetikai jelenségek kémiai rendszerekben*, egyetemi jegyzet, Debrecen–Budapest–Gödöllő, 1992.

[4] JOHN P. CHESICK. Interactive computer program system for integration of multistep reaction rate equations, *Journal of Chemical Education* **56**(9):585, 1979.

[5] BRUCE L. CLARKE. Stoichiometric network analysis of the oxalate–persulfate–silver oscillator, *Journal of Chemical Physics* **97**(4):2459–2472, 1992.

[6] EVELYNE CONTEJEAN, HERVÉ DEVIE. An efficient incremental algorithm for solving systems of linear Diophantine equations, *Information and Computation* **113**:143–172, 1994.

[7] ERIC DOMENJOUD. Solving Systems of Linear Diophantine Equations: an Algebraic Approach, in A. Tarlecki (ed.) *Mathematical Foundations of Computer Science 1991: Proc. of the 16th International Symposium*, pp. 141–150. Springer, 1991.

[8] PÉTER ÉRDI, JÁNOS TÓTH. *Mathematical Models of Chemical Reactions, Theory and Applications of Deterministic and Stochastic Models*, Manchester University Press, Manchester, Princeton University Press, Princeton, 1989. ISBN: 069 108 532 3

[9] MARTIN FEINBERG. *The Chemical Reaction Network Toolbox: Downloads and Instructions*, `http://www.che.eng.ohio-state.edu/~feinberg/crnt/`, 2005. március 13.

[10] FERENC FRIEDLER, L. T. FAN AND BALÁZS IMREH. Process Network Synthesis: Problem Definition, *Networks* **31**(2):119–124, 1998. URL: `http://dcs.vein.hu/cikkek/Proc_Netw_Synth_Probl_Def.pdf`

[11] FERENC FRIEDLER, K. TARJÁN, Y. W. HUANG, AND L. T. FAN. Combinatorial Algorithms for Process Synthesis, *Computers & Chemical Engineering* **16**(4):313–320, 1992. URL: http://dcs.vein.hu/cikkek/Comb_Alg_for_Proc_Synth.pdf

[12] SZILVIA GYAPAY, ANDRÁS PATARICZA. A Combination of Petri Nets and Process Network Synthesis, in *Proc. 2003 International Conference on Systems, Man & Cybernetics*, Washington D.C., pp. 1167–1174. 2003.

[13] BALÁZS IMREH. Automaton Theory Approach for Solving Modified PNS problems, *Acta Cybernetica* **15**:327–338, 2002.

[14] PETER KIRKEGAARD, ERLING BJERGBAKKE. CHEMSIMUL: a simulator for chemical kinetics, Technical Report Risø-R-1085(EN) (rev. ed.), Risø National Laboratory, Roskilde, Denmark, 2000. URL: http://www.risoe.dk/rispubl/PBK/pbkpdf/ris-r-1085rev.pdf

[15] KRISZTIÁN KOVÁCS, MIKLÓS RIEDEL, JÁNOS TÓTH, AND BÉLA VIZVÁRI. Decomposition of the permanganate/oxalic acid overall reaction to elementary steps based on integer programming theory, *Physical Chemistry Chemical Physics* **6**(6):1236–1242, 2004.

[16] LÁSZLÓ BÉLA KOVÁCS. *Combinatorial Methods of Discrete Programming*, Mathematical Methods of Operations Research (vol. 2), Akadémiai Kiadó, Budapest, 1980. ISBN: 963 052 004 4

[17] THOMAS G. KURTZ. The Relationship between Stochastic and Deterministic Models for Chemical Reactions, *Journal of Chemical Physics* **57**(7):2976–2978, 1972.

[18] DANIEL LICHTBLAU. Revisiting strong Gröbner bases over Euclidean domains, Research Report, Wolfram Research Inc., Champaign, 2001.

[19] LÓCZI LAJOS. Mathematica, *Középiskolai Matematikai és Fizikai Lapok* **52**(6), 2002. URL: http://www.math.bme.hu/~jtoth/FelsMma/KoMaLMathematicaLL.pdf

[20] JAVIER MARTÍNEZ, MANUEL SILVA. A simple and fast algorithm to obtain all invariants of a generalized Petri net, in C. Girault, W. Reisig (eds.) *Application and Theory of Petri Nets*, pp. 301-310. Informatik Fachberichte **52**, Springer, Berlin, 1982.

[21] DÁVID PAPP, BÉLA VIZVÁRI. Effective solution of linear Diophantine equation systems with an application in chemistry, Research Report RRR 28-2004, RUTCOR, Rutgers State University of New Jersey, 2004. URL: http://rutcor.rutgers.edu/pub/rrr/reports2004/28_2004.ps

[22] PATARICZA ANDRÁS (szerk.) *Formális módszerek az informatikában*, Typotex, Budapest, 2004. ISBN: 963 932 608 1

[23] BALÁZS POLGÁR, ENDRE SELÉNYI. Probabilistic Diagnostics with P-graphs, *Acta Cybernetica* **16**:279–291, 2003.

[24] LOÏC POTTIER. Minimal solutions of linear diophantine systems: bounds and algorithms, in: R.V. Book (ed.) *Proc. 4th Conference on Rewriting Techniques and Applications, Como, Italy*, pp. 162–173. Lecture Notes in Computer Science **488**, Springer, 1991.

[25] RÓNYAI LAJOS, IVANYOS GÁBOR ÉS SZABÓ RÉKA. *Algoritmusok*, Typotex, Budapest, 1998. ISBN: 963 913 216 0

[26] GUILLERMO SANCHEZ, JESUS LOPEZ-FIDALGO. Mathematical Techniques for Solving Analytically Large Compartmental Systems, *Health Physics Journal* **85**(2):184–193, 2003. URL: `http://web.usal.es/~guillerm/biokmod.htm` 2005. április 2.

[27] NIGEL P. SMART. *The algorithmic resolution of Diophantine equations*, London Mathematical Society Student Texts **41**, Cambridge University Press, Cambridge, 1998. ISBN: 052 164 633 2

[28] ISTVÁN SZALKAI. A new general algorithmic method in reaction syntheses using linear algebra, *Journal of Mathematical Chemistry* **28**(1–3):1–34, 2000.

[29] SZILI LÁSZLÓ, TÓTH JÁNOS. *Matematika és Mathematica*, ELTE Eötvös Kiadó, Budapest, 1996. ISBN: 963 463 004 9

[30] ALISON S. TOMLIN, TAMÁS TURÁNYI, MICHAEL J. PILLING. Low temperature combustion and autoignition, in R. G. Compton (ed.) *Mathematical tools for the construction, investigation and reduction of combustion mechanisms*, pp. 293-437. Comprehensive Chemical Kinetics **35**, Elsevier, 1997.

[31] TAMÁS TURÁNYI. KINAL — A program package for kinetic analysis of complex reaction mechanisms, *Computers & Chemistry* **14**(3):253-254, 1990.

[32] VIZVÁRI BÉLA. *Egészértékű programozás*, egyetemi jegyzet, ELTE TTK, Tankönyvkiadó, Budapest, 1992.

[33] A. I. VOLPERT, S. I. KHUDYAEV. *Analysis in classes of discontinuous functions and the equations of mathematical physics*, Nauka, Moscow, 1975.

[34] BENJAMIN M. M. DE WEGER. *Algorithms for Diophantine equations*, CWI Tract **65**, Centrum voor Wiskunde en Informatica (CWI), Amsterdam, 1989. ISBN: 906 196 375 3

[35] STEPHEN WOLFRAM. *The Mathematica Book, Fifth Edition*, Wolfram Media, Champaign, 2003. ISBN: 1 5795 5022 3. URL: http://documents.wolfram.com/mathematica/book/

[36] —. *Differential Equations for Chemical Kinetics*, Mathematica Information Center, Wolfram Research Inc., http://library.wolfram.com/examples/chemicalkinetics/ 2005. április 2.

[37] —. *Petri Nets World*, Theoretical Foundations of Computer Science group, University of Hamburg, Germany, http://www.informatik.uni-hamburg.de/TGI/PetriNets/ 2005. május 14.

[38] —. *The home site for the Systems Biology Markup Language* http://sbml.org 2005. május 14.