

EÖTVÖS LORÁND TUDOMÁNYEGYETEM

---

FRAKNÓI ÁDÁM

**Adatcsomag-továbbító programok  
beágyazása programozható céleszközökre:  
elmélet, algoritmusok és kiértékelés**

TDK-dolgozat

ELTE Természettudományi Kar  
Matematika MSc

Témavezető:

**BÉRCZI-KOVÁCS ERIKA**

Operációkutatási tanszék, TTK  
Egyetemi adjunktus

Külső témavezetők:

**VASS BALÁZS**

BME VIK TMIT  
Egyetemi tanársegéd

**RÉTVÁRI GÁBOR**

BME VIK TMIT  
Tudományos főmunkatárs



Budapest, 2022

# Tartalomjegyzék

|                                                   |           |
|---------------------------------------------------|-----------|
| <b>1. Bevezetés</b>                               | <b>1</b>  |
| <b>2. Technikai háttér</b>                        | <b>3</b>  |
| 2.1. RMT . . . . .                                | 3         |
| 2.2. dRMT . . . . .                               | 6         |
| <b>3. Elméleti eredmények</b>                     | <b>10</b> |
| 3.1. Matematikai fogalmak . . . . .               | 11        |
| 3.2. Modellek . . . . .                           | 13        |
| 3.3. Elméleti eredmények összefoglalása . . . . . | 14        |
| 3.4. Bonyolultságok . . . . .                     | 15        |
| 3.5. Approximációs algoritmusok . . . . .         | 17        |
| 3.5.1. Előütemezés . . . . .                      | 17        |
| 3.5.2. Eredmények . . . . .                       | 18        |
| 3.6. Több ütemezés beágyazása . . . . .           | 21        |
| 3.6.1. Matematikai fogalmak . . . . .             | 21        |
| 3.6.2. Bonyolultságok . . . . .                   | 23        |
| 3.6.3. Periodicitás . . . . .                     | 24        |
| <b>4. Szimulációk</b>                             | <b>30</b> |
| 4.1. Ütemezés hossza . . . . .                    | 31        |
| 4.2. $P$ periódus minimalizálása . . . . .        | 31        |
| <b>5. Konklúzió, nyitott kérdések</b>             | <b>34</b> |

# Köszönetnyilvánítás

Szeretnék köszönetet mondani témavezetőmnek, Bérczi-Kovács Erikának, aki a témát ajánlotta számomra. A bizonyításaimat mindig figyelmesen átnézte, a hiányosságokra felhívta a figyelmemet, illetve számos tanáccsal látott el a dolgozat készítése során.

Köszönetet szeretnék mondani Vass Balázsnak, aki folyamatosan rendelkezésre állt a technikai háttér megértésében, modellek kitalálásában, szimulációk tervezésében. A felmerülő kérdéseimre mindig készségesen válaszolt.

Továbbá szeretném megköszönni Rétvári Gábornak, akitől származik a probléma felvetése, illetve folyamatos iránymutatással látott el, hogy mely problémák lehetnek érdekesek egy mérnök számára.

# 1. fejezet

## Bevezetés

A kommunikációs hálózatok csomópontjaiban adatcsomag-továbbító céleszközök (switchek és routerek) helyezkednek el, melyek elsődleges célja az, hogy a bejövő adatcsomagokat feldolgozza és eldöntse, hogy milyen irányba menjenek tovább. A korábban gyártott tradicionális switchek előre meghatározott funkciókkal rendelkeztek, aminek a hátránya az volt, hogy erősen kötött volt a viselkedésük, így az internet viselkedését néha csak lecserélésük által lehetett módosítani. Ezt a problémát orvosolva jelentek meg az úgynevezett programozható switchek, amik manapság már egyre népszerűbbek.

A különböző alkalmazások (hálózati telemetria [10], in-network computing [5], terhelés elosztás (load-balancing) [9, 11], stb.) által elvárt viselkedéseket egy speciális programnyelvben, például P4-ben [1, 3] fogalmazhatjuk meg, melyet egy fordítóprogram a programozható switchünkre lefordít. A fordítandó programkód reprezentálható egy körmentes irányított gráffal, amelyet be szeretnénk ágyazni a switch erőforrásaira úgy, hogy az minél hatékonyabb és energiatakarékosabb legyen. Ezt a feladatot matematikailag is modellezhetjük egy alap ütemezési feladattal továbbá egyéb, komplex feltételekkel.

Belátható, hogy ezen gráfbeágyazási feladatok  $\mathcal{NP}$ -nehéz problémák, így IP megoldókkal, illetve mohó algoritmusokkal próbálnak optimális, vagy optimálist közelítő megoldásokat keresni. Az IP megoldó lassú, van amikor órákba telik akár egy közepes méretű gráfra is megtalálni az optimális megoldást [15]. Ezzel szemben a heurisztikus megközelítések, például mohó algoritmusok polinomiális időben garantáltan találnak egy beágyazást, habár nem olyan jót, mint az IP megoldók.

Nagy mennyiségben gyártásban lévő programozható switch az RMT (Reconfigurable Match Tables) architektúrájú INTEL Tofino switch [7]. Azonban az RMT architektúrájának van néhány problémája, melyeket egy új, úgynevezett dRMT (disagg-

regulated RMT) architektúra kifejlesztésével próbálták orvosolni [4], habár gyártásba eddig még nem került. Azt vizsgálom a dolgozatomban, hogy a gráfbeágyazások különböző feltételek mellett milyen hatékonyan végezhetők el a dRMT architektúrán. Ehhez kidolgoztam a modellek elméletét, konstans közelítő algoritmusokat terveztem, implementáltam, majd szimulációkat végeztem rajtuk összehasonlítva a [4] cikk heurisztikáival.

A 2. fejezetben részletesen körüljárjuk az RMT és dRMT architektúra technikai hátterét. A 3. fejezetben matematikai eszközökkel vizsgáljuk meg a problémát, meghatározzuk a különböző modellek bonyolultsági osztályait, approximációs algoritmusokat adunk. A 4. fejezetben több szimulációt is bemutatunk, megmutatjuk, hogy az előző fejezetben adott mohó algoritmus nem csak konstans közelítő, hanem a gyakorlatban hatékonyabb is, mint a [4]-ben adott heurisztikák. Az 5. fejezetben megemlítek több, a dolgozat írásakor még nyitott, tovább kutatandó kérdést a témával kapcsolatban.

Az elért eredményeink egy része megjelent a CONEXT konferencián szervezett EuroP4 workshopon [16].

## 2. fejezet

# Technikai háttér

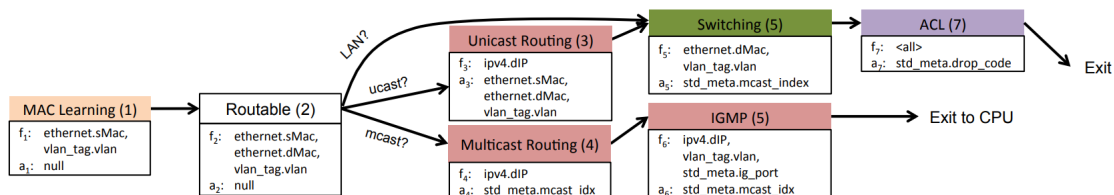
### 2.1. RMT

A szükséges hálózati viselkedéseket (pl. adatsomagok továbbítási portjának kiválasztása, adatfolyam statisztikák készítése stb.) rendszerint egy úgynevezett P4 programnyelvben [1, 3] definiálják, amit aztán a P4 fordító (amit minden hardverre a switch gyártók külön megírnak) az adott switchre alkalmazza.

Egy fordító feladata, hogy feldolgozza a P4 programot, és az adott hardver megkötéseit figyelembe véve (például elérhető memóriaterület és memóriatípusokat, munkaállomások száma), találjon egy minél hatékonyabb és energiatakarékosabb úgynevezett beágyazást az RMT architektúrába [8].

Egy P4 program meghatároz egy vezérlésfolyamot (control flow), amit reprezentálhatunk egy olyan körmentes irányított gráffal, ahol a csúcsok úgynevezett match-action tábláknak felelnek meg, az élek pedig függőségeknek.

Ebben a gráfban egy-egy út egy-egy lehetséges feldolgozását mutatja egy adatsomagnak. A táblákban bejegyzések szerepelnek, amik két részből állnak: egyeztetésekből (match) és műveletekből (action). Az előbbiek párhuzamosan keresnek, a legjobb egyezés műveletét végrehajtják. Az utóbbi utasításokat tartalmaz, pél-



2.1. ábra. Egy lehetséges vezérlésfolyamat (control flow). [8]

dául az adatcsomagon módosítson egy mezőt, megmondja, hogy melyik tábla felé haladjon tovább az adatcsomag, stb. A táblákat élek kötik össze, melyek megelőzési feltételeket határoznak meg. Például a 2.1. ábrán a Unicast Routing táblában lévő feladatokat csak akkor végezhetjük el, ha már végeztünk a Ratable-ben lévő összes feladattal.

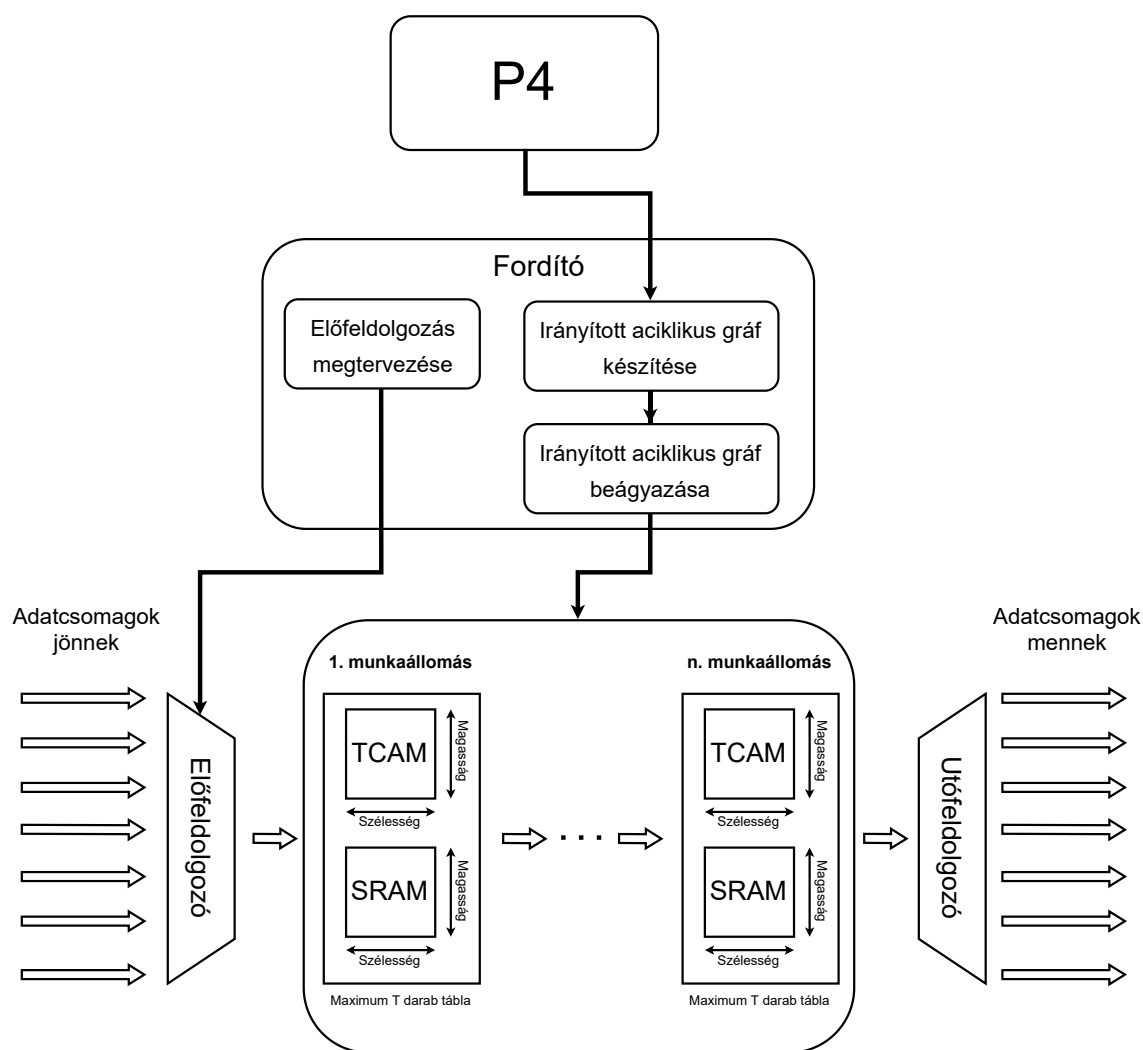
Ezt a vezérlésfolyamot és a hozzá tartozó körmentes irányított gráfot a P4 fordító az első lépésében el is készíti. Ezután egyrészt ezt a gráfot a fordító megpróbálja beágyazni a csőszerűen (szekvenciálisan) felépített munkaállomásokba úgy, hogy azok teljesítsék a hardver szerkezeti megkötéseit, illetve minél kevesebb munkaállomást használjon fel. (Ezt a beágyazást fogjuk majd matematikailag vizsgálni.) Másrészt pedig megtervezi az adatcsomagok előfeldolgozásának menetét. (2.2. ábra)

A feldolgozandó adatcsomagok folyamatosan jönnek az előfeldolgozóba, majd végighaladnak a munkaállomásokon, és legvégén az utófeldolgozón keresztül továbbhaladnak.

A match-action táblákat a munkaállomásokban szereplő memóriákban tárolják. Minden munkaállomásban kétfajta memóriatípus is elérhető: TCAM és SRAM. Az előbbi kitűnő az előtag- (prefix) és helyettesítő karakteres (wildcard) egyezések keresésében, azonban drágább és jelentős energiaköltséggel jár, míg az utóbbi ideális a pontos (exact) vagy tartományi (range) egyezésekben, illetve a műveletek kódjának tárolásában.

Leegyszerűsítve, a beágyazásra az alábbi szerkezeti megkötések vannak:

- **Memória:** Minden memóriának van egy szélessége és egy magassága. Minden bejegyzést el kell helyeznünk a memória valamelyik sorában úgy, hogy a memória minden sorában a szélességek összege nem haladhatja meg az adott memóriatípus korlátját.
- **Horizontal-split** (h-split): ha egy tábla nem férne bele egy munkaállomásba, akkor a táblát horizontálisan elvághatjuk, így a bejegyzéseket több munkaállomásba helyezzzük el.
- **Crossbar:** Minden munkaállomásban van egy korlát a táblák számára, az egyeztetések szélességeinek összegére és a műveletek szélességeinek összegére.
- **Word-packing:** Néhány bejegyzést *együtt* lehet ragasztani, ezzel memóriát spórolva.



2.2. ábra. Egy switch felprogramozásának menete.

- **Késleltetés:** Ha két munkaállomásban lévő táblák között nincsen él, akkor egymás utáni órajelben végre lehet hajtani a munkaállomásokat. Ha egy táblából egy másik táblába megy egy él, és az előbbi egy olyan mezőn hajt végre műveletet, amit az utóbbi felhasznál az egyeztetés során, akkor egy bizonyos számú (pl. legalább 12) órajelnek el kell telnie azon két munkaállomás között, ahová beágyasztuk őket. (Például a 2.1. ábrán Unicast Routing és Switching táblák között ilyen van.)

Ezen megkötések mellett a cél minimalizálni a munkaállomások számának, az energiahasználatot és a szekvencia hosszát.

A [15] cikkben matematikai eszközökkel definiálták ezen modelleket, majd megvizsgálták ezen feladatok bonyolultság osztályait, illetve approximációs algoritmusokat adtak rá.

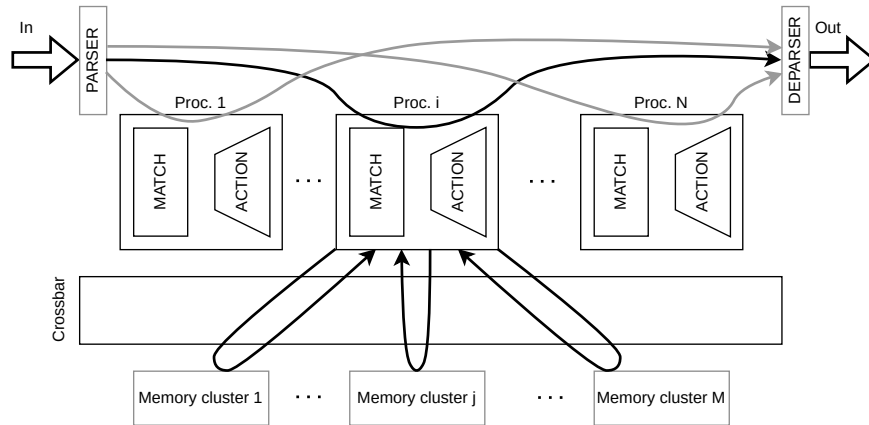
## 2.2. dRMT

A dRMT (disaggregated Reconfigurable Match-Action Table) [4] egy új architektúra programozható switchekre, ami az RMT switchek két fontos, a csőszerű szekvenciális architektúrából fakadó problémájára kínál megoldást:

1. Az RMT-ben a memória a munkaállomáshoz tartozik, ami csak az adott munkaállomásból érhető el. Ez párhuzamosan elvégezhető kereséseknél, illetve nagy tábláknál is problémás lehet. Tekintsünk néhány táblát, amiket párhuzamosan el lehet végezni. Ha a táblák mérete túl nagy lenne a munkaállomás memóriájához képest, akkor kénytelenek vagyunk több munkaállomásba elhelyezni őket, azonban ezáltal a hozzáadott munkaállomások számítási kapacitásai nem lesznek kihasználva. Ugyanígy, ha egy tábla memóriaigénye nagyobb, mint amekkora egy munkaállomás memóriája, akkor több munkaállomásba kell elhelyezni ezt a match-action táblát.
2. Az RMT szekvenciálisan hajtja végre az egyezéseket (match), majd a műveleteket (action), miközben a csomagok áthaladnak a csövön. Azaz az első munkaállomásban egy match után action jön, a második munkaállomásban is egy match után action jön és így tovább. Ez is a hardveres erőforrások kihasználatlanságához vezethet, ha a matchek és actionök kiegyensúlyozatlanul fordulnak elő. Például, ha néhány actiont nem kell matchnek megelőznie, akkor is le van foglalva a memóriaigénye matchnek. Továbbá, ha egy program nem fér bele a hardver által elérhető munkaállomások számába, (mert például az aciklikus gráfunkban lévő leghosszabb út hossza nagyobb, mint a munkaállomások száma) akkor lehetőség van arra, hogy még egyszer végigmenjen a program a szekvencián azért, hogy minden feladatot elvégezhessen. Ráadásul akkor is végig kell mennie a programnak még egyszer a szekvencián, ha csupán egy extra munkaállomásra lenne szükség, ezzel akár megfelelheti (vagy ha  $k$ -szor megy végig a szekvencián, akkor  $k$ -ad részére csökkentheti) a teljesítményt a maximális teljesítményhez képest.

A dRMT úgy oldja meg mindkét problémát, hogy különválasztja a hardveres erőforrásokat (2.3. ábra):

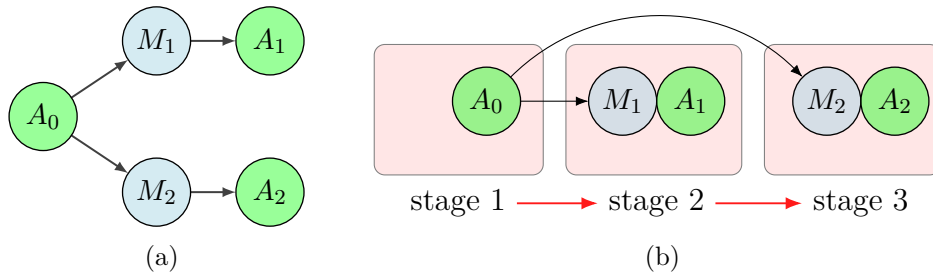
1. **Memória:** A memóriát szétválasztja a processzortól, amit egy úgynevezett crossbaron keresztül ér el a processzor.
2. **Processzor:** Az RMT-ben szereplő csőszerű szekvenciális munkaállomásokat felváltják az úgynevezett match-action processzorok halmaza. Ahelyett, hogy



2.3. ábra. A dRMT architektúra felépítése.

minden munkaállomás foglalkozik egy kicsit a csomaggal, majd továbbadja a következő munkaállomásnak, most egy csomaggal csak egy processzor fog foglalkozni. A csomag mindaddig a processzornál található, amíg annak a teljes programjával nem végez. (Egy processzor természetesen több csomaggal is foglalkozhat egyszerre.)

A memória és processzor szétválasztása jelentős flexibilitást nyújt. Megoldja a memóriát illetően a párhuzamos keresési problémát és a nagy táblák problémáját, a gráf beágyazást illetően pedig akár kevesebb munkaállomás/processzor segítségével is megoldhatók ugyanazok a feladatok. Például egy példa program beágyazása az RMT architektúrába 2.4. ábrán látható 3 munkaállomást felhasználva, míg ugyanez a program a dRMT architektúrában  $P = 2$  periódussal megoldható (2.5. ábra), azaz 50%-al több csomagot is fel tud dolgozni ugyanannyi processzorral.



2.4. ábra. Egy példa program és annak beágyazása az RMT architektúrában 3 munkaállomás felhasználásával.

| Processzor \ órajel | 0     | 1     | 2     | 3            | 4            | 5            | 6            |
|---------------------|-------|-------|-------|--------------|--------------|--------------|--------------|
| 0                   | $A_0$ | $M_1$ | $M_2$ | $A_1 \& A_2$ |              |              |              |
| 1                   |       | $A_0$ | $M_1$ | $M_2$        | $A_1 \& A_2$ |              |              |
| 0                   |       |       | $A_0$ | $M_1$        | $M_2$        | $A_1 \& A_2$ |              |
| 1                   |       |       |       | $A_0$        | $M_1$        | $M_2$        | $A_1 \& A_2$ |

2.5. táblázat. A példa program egy lehetséges ütemezése a dRMT architektúrában 2 processzoron. Megfelelő paraméterek mellett ez egy optimális beágyazás  $P = 2$  periódussal a dRMT architektúrába.

Feltehetjük, hogy a P4 program kimenetéből a fordító segítségével rögtön megkapjuk az aciklikus gráfunkat, aminek a csúcsai lesznek a match és action csúcsok, az élein pedig adott egy kapacitás függvény, hogy mennyi szünetnek kell eltelnie a két csúcs között. Ezt szükséges most a dRMT architektúrába beágyazni (avagy ütemezni) nem megsértve a P4 program megelőzési feltételeit vagy a dRMT architektúra szerkezeti korlátait. A megelőzési feltételek alatt azt értjük, hogy ha egy match vagy action csúcs után egy másik valamilyen csúcsba él megy, akkor a legkorábban csak a  $\Delta M$ -ik illetve  $\Delta A$ -ik órajel után hajtható végre.

A match és action indítására az alábbi szerkezeti korlátok vannak:

- (i) Egy processzor legfeljebb  $\overline{M}$  darab párhuzamos keresést kezdeményezhet, egyenként legfeljebb  $b$  bittel. (Például ha egy match-action táblázatban keresendő string  $2,5 \cdot b$  bit hosszú, akkor három párhuzamos  $b$  bites vektor elküldésével meg tudja ezt tenni, feltéve, hogy  $3 \leq \overline{M}$ ).
- (ii) Egy processzor legfeljebb  $\overline{A}$  darab action mező módosítást hajthat végre párhuzamosan.
- (iii) Egy processzoron legfeljebb IPC (inter-packet concurrency) darab match indítható minden órajelben.
- (iv) Egy processzoron legfeljebb IPC (inter-packet concurrency) darab action indítható minden órajelben.

Ezen kívül még számos korlát van (habár ezekkel már nem fogunk foglalkozni), például:

- (v) **Memória blokkok:** Ugyanúgy lehetnek TCAM és SRAM típusú memóriák, illetve egy memória blokkra ugyanazok a korlátozások vonatkoznak, mint az RMT architektúrában.
- (vi) **Memória elérés:** Minden órajelben egy match-action táblához tartozó memóriablokkhoz legfeljebb egy processzor egy csomagja férhet hozzá.
- (vii) **Crossbar:** A processzor és a memória közötti információáramlásért felel a crossbar, aminek szintén vannak erőforrás-korlátai.

A dolgozat során azonban fel fogjuk tenni, hogy egy processzor (és egyféle memória) áll rendelkezésre, ezért a megszorításokat a következő szekcióban rögtön ennek a függvényében mondjuk majd ki. Az IPC értéke jellemzően 1 vagy 2 szokott lenni.

A dolgozatban főként az  $IPC = 1$  értékre fogunk fókuszálni, ugyanis a [4] cikkben a szimulációk már azt mutatták, hogy  $IPC = 2$  esetén az ILP megoldók könnyedén megtalálták az optimális megoldásokat.

A [4] cikkben az ODG-nek egyetlen ütemezését ismételték  $P$  periódusonként, ugyanis a gyakorlatban egyszerűbb tárolni és számolni, ha periodikusan ismétlődnek a feladatok. Így egy-egy processzor ciklikus rendezését eltolva  $P$  processzorral órajelenként tudunk egy-egy csomagtovábbítást kezdeni és végezni. Céluk egyrészt ennek a  $P$  periódusnak, másrészt az ütemezés hosszának minimalizálása volt. A következő fejezetben mi is fel fogjuk tenni ezen feltételeket, azonban a végén tesszünk egy kitérőt, és annyit módosítunk ezen, hogy meg fogjuk engedni azt is, hogy egy helyett egyszerre akár  $k$  darab ütemezést is váltogassunk, amivel javulást lehet elérni a hatékonyságban.

## 3. fejezet

# Elméleti eredmények

Ebben a fejezetben felépítünk egyre bővülő modelleket azzal a céllal, hogy jobban megértsük az erőforrás igényüket, amikor egy P4 programot a dRMT struktúrába ágyaznánk be. Minden modell során megpróbáljuk megállapítani a bonyolultsági osztályát, milyen approximációs algoritmus adható rá, vagy milyen approximációk adására nincs esély, ha  $\mathcal{P} \neq \mathcal{NP}$ .

Minden modellünkben a P4 programot egy  $D = (V, E)$  aciklikus irányított gráffal reprezentáljuk, ahol  $V = V_a \cup V_m$ , azaz a csúcshalmaz fel van osztva match és action csúcsokra. Az  $E$  irányított élek a függőségeket reprezentálják a csúcsok között. Ha az  $e = (u, v)$  töve match vagy action csúcs, akkor a  $v$  csúcs végrehajtásához legalább  $\Delta M$  vagy  $\Delta A$  órajelet szükséges várni az  $u$  elkezdése után. Továbbá minden processzor egyszerre legfeljebb  $\overline{M}$  párhuzamos tábla keresést kezdeményezhet, és legfeljebb  $\overline{A}$  action mezőt módosíthat párhuzamosan.  $\Delta M, \Delta A, \overline{M}, \overline{A}$  paraméterek pozitív egész számok. (példa: 2.4. ábra)

A [4] cikkben olyan periodikus ütemezéseket vizsgálnak, amikor egyetlen csomagot ismételnék periodikusan egy processzoron. Ez azzal ekvivalens, hogy a csomag  $t, t+P, t+2P, \dots$  órajeleiben szereplő csúcsok halmazának teljesíteniük kell a megfelelő erőforrás  $(\overline{M}, \overline{A})$  és késleltetési feltételeket  $(\Delta A, \Delta M)$  minden  $t \in \{1, \dots, P\}$  modulo osztályra.

BASIC modellben nincs további megkötés, mint a fentiek, továbbá feltesszük, hogy minden csúcs szélessége 1. Belátható, hogy ebben a modellben az optimális  $P$  periódusszám megtalálható polinom időben. (3.4.1. tétel)

IPC1 modellben a BASIC fenti megkötései mellett megjelenik az IPC (Inter-Packet Concurrency) feltétel, azaz minden órajel legfeljebb IPC (ami ebben a modellben 1) darab match és action csúcsot indíthat. A WIDTH modell annyiban különbözik a BASIC modelltől, hogy a match és action csúcsok szélességei tetszőle-

ges pozitív egészek lehetnek. Belátható, hogy mindkét modell  $\mathcal{NP}$ -nehéz, továbbá nem is közelíthető meg, kivéve, ha  $\mathcal{NP} = \mathcal{P}$ . (3.4.3. tétel, 3.4.4. tétel) Az IPC1 modellre azonban megadunk lineáris idejű 3-approximációs mohó algoritmust. (3.5.7. tétel)

A két modellt egyesítve kapjuk a WIDTH-IPC1 modellt, ami szintén  $\mathcal{NP}$ -nehéz és nem közelíthető meg, kivéve, ha  $\mathcal{NP} = \mathcal{P}$ . Azonban ebben a modellben is megadható egy kvázi lineáris idejű 4-approximációs mohó algoritmust. (3.5.4. tétel)

Ha az IPC értékét 2-re változtatjuk, akkor kapjuk a WIDTH-IPC2 modellt.

A [4] cikkben a WIDTH-IPC1 és WIDTH-IPC2 modellekkel foglalkoztak.

A következő két alfejezetben (3.1., 3.2) matematikai eszközökkel precízen és részletesebben definiáljuk a modelleket, majd utána összefoglaljuk a dolgozatban szereplő eredményeket (3.3). Később megvizsgáljuk a modelleket bonyolultsági osztályait (3.4), majd pedig approximációs algoritmusokat adunk különböző modellekre (3.5). Az utolsó alfejezetben (3.6) elrugaszkodunk attól a feltételezéstől, hogy mindig egyetlen csomagot ismétlünk periodikusan, és így vizsgáljuk az elméleti legjobb ütemezéseket. Ezzel akár még hatékonyabb ütemezéseket lehetne elérni, habár az ütemezések tárolására több memóriára lenne szükség.

## 3.1. Matematikai fogalmak

Az alábbi alfejezetben definiáljuk a fogalmakat a dRMT modellhez.

Egy dRMT feladatot az alábbi bemenethalmaz ír le: adott egy  $D = (V, E)$  irányított aciklikus gráf, ahol a  $V$  csúcshalmaz fel van osztva a  $V_m$  és  $V_a$  match és action csúcshalmazokra. Továbbá adott egy  $W : V \rightarrow \mathbb{N}^+$  szélességi függvény,  $\Delta M, \Delta A \in \mathbb{N}^+$  késleltetési konstansok,  $\overline{M}, \overline{A}, IPC \in \mathbb{N}^+ \cup \{\infty\}$  szerkezeti korlátok.

A  $\Delta M$  és  $\Delta A$  konstansok indukálnak egy  $l : V \rightarrow \{\Delta M, \Delta A\}$  **késleltetési függvényt** minden csúcsra az alábbi módon:  $l(v_m) = \Delta M$  és  $l(v_a) = \Delta A$  minden  $v_m \in V_m$  és  $v_a \in V_a$ -ra.

**3.1.1. Definíció.** Egy tetszőleges  $S : V \rightarrow \mathbb{N}^+$  függvényt **ütemezésnek** nevezzük. (Itt  $\mathbb{N}^+$  az órajeleket reprezentálja.) Ha teljesül még rá az is, hogy minden  $(i, j) \in E$  élre  $l(v_i) \leq S(j) - S(i)$ , akkor **megengedett ütemezésnek** hívjuk.

Egy megengedett ütemezést szeretnénk beágyazni periodikusan úgy, hogy közben bizonyos szerkezeti korlátok is teljesülnek. Erről szól az alábbi definíció:

**3.1.2. Definíció.** Egy  $S$  megengedett ütemezésre és  $P \in \mathbb{N}^+$ -re legyen  $\mathcal{S}$  az  $S_i$  ütemezések halmaza, ahol  $S_i(v) := S(v) + iP$  (ahol  $i \in \mathbb{N}$ ). Azt mondjuk, hogy  $S$

**végrehajtható  $P$ -periodikus ütemezés**, ha  $\mathcal{S}$  minden órajelben a csúcsok kielégítik az erőforrás és IPC korlátot.

- **Erőforrás korlát**, (i) és (ii): Minden match és action csúcsnak van egy **szélessége**, amit a  $W$  függvény ír le. Minden órajelben a match és action csúcsok szélességeinek összege legfeljebb  $\overline{M}$  és  $\overline{A}$  lehet. Formálisan

$$\forall t \in \mathbb{N}^+ : \sum_{S \in \mathcal{S}} \sum_{\substack{v_m \in V_m \\ S(v_m)=t}} W(v_m) \leq \overline{M}$$

és ugyanez az action csúcsokra.

- **IPC korlát**, (iii) és (iv): Az  $IPC$  érték egy felső korlát arra, hogy egy órajelben hány ütemezés (avagy csomag) foglalkozhat match illetve action csúcsokkal. Formálisan

$$\forall t \in \mathbb{N}^+ : \#\{S \in \mathcal{S} \mid \exists v_m \in V_m : S(v_m) = t\} \leq IPC$$

és ugyanez az action csúcsokra.

A dRMT feladat kimenetének egy végrehajtható  $P$ -periodikus ütemezésnek kell lennie, ahol  $P \in \mathbb{N}^+$ .

Egy ütemezés fontos tulajdonsága még az ütemezés hossza, ugyanis amellett, hogy sok csomagot akarunk elvégezni, nem akarjuk azt, hogy egy csomag elvégzése túl sok időt vegyen igénybe.

**3.1.3. Definíció.** Legyen  $T = T(\mathcal{S}) := \max_{v \in V} (S(v) + l(v))$  egy  $\mathcal{S}$  ütemezés **hossza**.

Amikor egy megengedett  $\mathcal{S}$  ütemezésről akarjuk eldönteni, hogy ő végrehajtható  $P$ -periodikus ütemezés-e, akkor elegendő az alábbi feltételt leellenőrizni, nem kell hozzá az  $\mathcal{S}$  halmazt megkonstruálni és abban leellenőrizni a feltételeket. Ugyanez az állítás megtalálható a [4]-ben is. (3.5. tétel)

**3.1.4. Állítás.** Legyen  $P \in \mathbb{N}^+$  és  $\mathcal{S}$  egy megengedett ütemezés. Az ebből konstruált  $\mathcal{S}$  pontosan akkor végrehajtható  $P$ -periodikus  $k$ -ütemezés, ha a  $\mathcal{S}$ -ben az órajeleket mod  $P$  ekvivalencia osztályokra bontva minden ekvivalencia osztályra teljesül az erőforrás korlát és IPC korlát.

*Bizonyítás.* Meggondolható, hogy az  $\mathcal{S}$   $P$ -periodikus ütemezésben minden  $t > T(\mathcal{S})$  órajelben pontosan azon csúcsok fognak szerepelni, mint az  $\mathcal{S}$  ütemezés  $t \pmod{P}$  szerinti ekvivalencia osztályában.  $\mathcal{S}$  minden  $t \leq T(\mathcal{S})$  órajelében pedig ugyanezen csúcsoknak csak egy részhalmaza fog szerepelni. Így valóban elegendő csak  $\mathcal{S}$  modulo osztályaira vizsgálni az erőforrás korlátot és IPC korlátot.  $\square$

**3.1.5. Példa.** A 2.5. táblázatban  $P = 2$ , így ha a 0. processzort tekintjük, akkor elegendő az első sorban a páros, majd páratlan órajeleket megvizsgálni. Mindkettő teljesíti például az  $IPC = 1$  korlátot, ugyanis a páros számú órajelekben legfeljebb 1 db órajel tartalmaz match csúcsot, és legfeljebb 1 db órajel tartalmaz action csúcsot, valamint ugyanez elmondható a páratlan számú órajelekről is. (Ha azonban az  $A_1$  csúcsot áttennénk a 2-es órajelbe, akkor már csak az  $IPC = 2$  korlátot teljesítené.)

Elsődleges célunk az, hogy olyan végrehajtható  $P$ -periodikus ütemezést találjunk, aminek minél nagyobb a hatékonysága, azaz minél kisebb  $P$  periódussal rendelkezik. Másodlagos célunk pedig az, hogy minimalizáljuk  $T$ -t. Ezek alapján az alábbi módon definiáljuk az optimális megoldásokat:

**3.1.6. Definíció.**

$$OPT := \min \left( P \mid S \text{ egy végrehajtható } P\text{-periodikus ütemezés} \right)$$

$$OPT^T := \min \left( T(S) \mid S \text{ egy végrehajtható } P\text{-periodikus ütemezés} \right)$$

## 3.2. Modellek

Definiáljuk a vizsgált modelleket, melyek egy processzort és egyféle memóriát használnak. Mindegyik modellben legyen  $\overline{M}, \overline{A} \in \mathbb{N}^+$  és  $\Delta M, \Delta A \in \mathbb{N}^+$ .

**BASIC:**  $(IPC = \infty) + (W \equiv 1)$

Ebben a modellben egy olyan megengedett ütemezést keresünk, ahol minden csúcs szélessége legyen 1, és minden órajelben a csúcsoknak teljesíteniük kell az erőforrás korlátot, azaz legfeljebb  $\overline{M}$  és  $\overline{A}$  match és action csúcs szerepelhet egy órajelben. Azonban egy órajelben akár több csomag csúcsaival is foglalkozhatunk.

**WIDTH:**  $(IPC = \infty) + (W : V \rightarrow \mathbb{N}^+)$

Ez a modell annyiban különbözik a BASIC modellel, hogy mostantól egy csúcs szélessége egy tetszőleges természetes szám lehet, és egy órajelbe a match és action csúcsok szélességeinek összege legfeljebb  $\overline{M}$  és  $\overline{A}$  lehet.

**IPC1:**  $(IPC = 1) + (W \equiv 1)$

A BASIC modell feltételein felül bejön az, hogy egy órajelben legfeljebb egy csomag csúcsaival foglalkozhatunk. A 3.1.4. állítás miatt egy megengedett ütemezésben elegendő azt ellenőrizni, hogy minden ekvivalencia osztályban legfeljebb egy órajelben indítunk match és action csúcsokat.

**WIDTH-IPC1:**  $(IPC = 1) + (W : V \rightarrow \mathbb{N}^+)$

Az előző két modell pluszfeltételeit egybe rakva kapjuk ezt a modellt. Ezt használták a [4] cikkben is.

**WIDTH-IPC2:**  $(IPC = 2) + (W : V \rightarrow \mathbb{N}^+)$

Annyit módosítunk az előző modellhez képest, hogy mostantól egy órajelben legfeljebb két csomag csúcsaival foglalkozhatunk, azaz  $IPC = 2$ . Ezt a modellt is használták a [4] cikkben is.

### 3.3. Elméleti eredmények összefoglalása

Ebben a fejezetben összefoglaljuk, hogy milyen eredményeket tudunk az egyes modellekről.

A BASIC modellben világos, hogy az  $OPT$  értéke maximum  $\lceil \frac{|V_m|}{M} \rceil$  és  $\lceil \frac{|V_a|}{A} \rceil$  lehet. Kiderül azonban, hogy ennek a két értékének a maximuma el is érhető egy egyszerű mohó algoritmussal. (3.4.1. Tétel)

Azonban az  $OPT^T$  meghatározása már a BASIC modellben is  $\mathcal{NP}$ -nehéz probléma. Illetve ha bővül a modellünk az  $IPC = 1$  feltétellel vagy  $W : V \rightarrow \mathbb{N}^+$  feltétellel, az  $OPT$  meghatározása máris  $\mathcal{NP}$ -nehézzé válik.

Belátjuk, hogy WIDTH modell egy speciális esete ekvivalensek az RMT architektúra 1D1R modelljével [15]. (3.4.3. tétel) Az IPC1 és WIDTH-IPC1 modellek egy speciális esete pedig ekvivalens lesz egy ismert ütemezési problémával. (3.4.4. tétel)

Megadunk az IPC1 és WIDTH-IPC1 modellekre egy 3- és 4-közelítő algoritmust az  $OPT$ -ra nézve, melynek futási ideje kvázi lineáris. (3.5.4., 3.5.7. Tétel)

Összefoglalva az eredményeket a 3.1. táblázat tartalmazza.

A 3.6. alfejezetben általánosítjuk a problémát a WIDTH-IPC1 modell esetén, elrugaskodunk attól a feltételtől, hogy mindig egy megengedett ütemezést ágyazunk be periodikusan. Megengedjük azt, hogy  $k$  darab megengedett ütemezést ágyazzunk be periodikusan, vagy akár végtelen sokat is. Ezen feltételek mellett az optimális megoldást  $OPT_k$ -val és  $OPT_\infty$ -nel jelöljük. (Az eddig használt jelölésre igaz lesz, hogy  $OPT = OPT_1$ .)

Habár az  $OPT_k$  értékek nem feltétlen monoton csökkennek, könnyedén belátható, hogy  $OPT_1 \geq OPT_k \geq OPT_\infty$  minden  $k \in \mathbb{N}^+$  értékre, illetve  $OPT_1 \leq 2 \cdot OPT_\infty$ . (3.6.22. állítás, 3.6.23. állítás)

Megmutatjuk, hogy minden dRMT feladatra létezik  $K \in \mathbb{N}^+$ , hogy  $OPT_K = OPT_\infty$ , azonban ez a  $K$  tetszőlegesen nagy is lehet, ugyanis mutatunk egy példát,

| Modell neve:                                                                            | BASIC         | IPC1                   | WIDTH                  | WIDTH-IPC1             | WIDTH-IPC2           |
|-----------------------------------------------------------------------------------------|---------------|------------------------|------------------------|------------------------|----------------------|
| Bonyolultság                                                                            | $\mathcal{P}$ | $\mathcal{NP}$ -nehéz  | $\mathcal{NP}$ -nehéz  | $\mathcal{NP}$ -nehéz  | ?                    |
| Rossz hír: Nem közelíthető jobban, mint ... (, kivéve ha $\mathcal{P} = \mathcal{NP}$ ) | OPT           | $4/3 \cdot \text{OPT}$ | $3/2 \cdot \text{OPT}$ | $3/2 \cdot \text{OPT}$ | ?                    |
| Jó hír: Konstans közelíthető                                                            | OPT           | $3 \cdot \text{OPT}$   | ?                      | $4 \cdot \text{OPT}$   | $8 \cdot \text{OPT}$ |

3.1. táblázat. Eredmények összefoglalva. Rossz hír: a dRMT beágyazási probléma  $\mathcal{NP}$ -nehéz még akkor is, ha csak approximálni akarjuk. Jó hír: a BASIC modell polinomiális időben megoldható, a többi modell pedig konstans közelíthető kvázi lineáris időben, még az [4]-ben vizsgált WIDTH-IPC1 és WIDTH-IPC2 modellek is. aminél  $\text{OPT}_1 = \dots = \text{OPT}_{k-1} > \text{OPT}_k$ . (3.6.20. tétel, 3.6.21. tétel)

## 3.4. Bonyolultságok

Ebben az alfejezetben megvizsgáljuk, hogy az egyes modellek milyen bonyolultsági osztályba tartoznak.

**3.4.1. Tétel.** *A BASIC modellben*

$$\text{OPT} = \max \left( \left\lceil \frac{|V_m|}{M} \right\rceil, \left\lceil \frac{|V_a|}{A} \right\rceil \right).$$

Továbbá megadható egy végrehajtható OPT-periodikus ütemezés  $O(|E| + |V|)$  időben.

*Bizonyítás.* Az OPT alsó becslése könnyen látszik, ugyanis  $|V_m|$  match csúcsot kell beágyaznunk, amihez szükséges legalább  $\left\lceil \frac{|V_m|}{M} \right\rceil$  órajel. Figyelembe véve az action csúcsokat is, következik, hogy  $\text{OPT} \geq \max \left( \left\lceil \frac{|V_m|}{M} \right\rceil, \left\lceil \frac{k|V_a|}{A} \right\rceil \right)$

A másik irányhoz legyen  $P := \max \left( \left\lceil \frac{|V_m|}{M} \right\rceil, \left\lceil \frac{|V_a|}{A} \right\rceil \right)$ .

Vegyük a gráf egy tetszőleges  $v_1, v_2, \dots, v_n$  topologikus sorrendjét, (azaz  $v_i v_j \in E$  esetén  $i < j$ ). Ezt  $O(|E| + |V|)$  időben meg tudjuk tenni. Menjünk végig ebben a sorrendben a csúcsokon és ágyazzuk be a csúcsokat az alábbi módon: Legyen  $S(v_1) := 1$ .  $j > 1$  esetén, legyen  $\delta_j := \max\{S(v_i) + l(v_i) \mid v_i v_j \in E\}$ , ha  $v_j$  befoka legalább 1 egyébként  $\delta_j := S(v_{j-1})$ . Ha  $v_j \in V_m$ , akkor  $S(v_j) := \min\{k \geq \delta_j \mid \#\{i : i < j, v_i \in V_m, S(v_i) \equiv k \pmod{P} \leq \overline{M}\}\}$ . (Ha  $v_j \in V_a$ , akkor hasonlóan adódik a képlet.)

Vegyük észre, hogy ez egy megengedett ütemezés lesz, továbbá  $P$  megválasztása miatt teljesülnek az erőforrás korlátok is.

Az összes  $\delta_j$  kiszámolása  $O(|E|)$  idő. A  $v_i$  csúcs esetén  $S(v_i)$  meghatározásához tároljuk le mindig egy kétirányú éllistában, hogy a  $0, 1, \dots, P-1$  maradékosztályokra, mi az előző és következő maradékosztály, ami még nincs tele. Ha  $S(v_i)$  megtelt, akkor ennek megfelelően frissítsük az előző és következő órajelet, ami még nincs tele az ilyen típusú csúcsokkal. Ilyen módon  $S(v_i)$   $O(1)$  időben meghatározható. Így az algoritmus futási ideje  $O(|E| + |V|)$ .

□

Ezzel beláttuk, hogy BASIC modellben a minimális  $P$  periódusszám meghatározása  $\mathcal{P}$ -ben. Azonban  $OPT^T$  meghatározása (azaz a legrövidebb ütemezés hossza) már  $\mathcal{NP}$ -nehéz probléma, akármelyik modellt is tekintjük:

**3.4.2. Tétel.** *Annak eldöntése, hogy  $P, T \in \mathbb{N}^+$  esetén bármelyik 3.2. alfejezetben definiált modellben létezik-e  $S$  végrehajtható  $P$ -periodikus ütemezés, úgy hogy  $T(S) \leq T$ ,  $\mathcal{NP}$ -teljes. Továbbá nem létezik  $4/3$ -nél jobban közelítő algoritmus az  $OPT^T$ -re nézve, kivéve ha  $\mathcal{P} = \mathcal{NP}$ .*

*Bizonyítás.* Speciális esetként feltehetjük, hogy csak match csúcsaink vannak és  $\Delta M = 1$ . Továbbá az IPC értéke nem számít, hiszen csak egy darab  $S$  ütemezést keresünk.

Az így kapott modell pedig egy klasszikus ütemezési feladattal ekvivalens,  $P|prec, p_j = 1|C_{max}$ , ami pedig visszavezethető a KLIKK problémára. [12] A bizonyítás megtalálható a [14] 3.4.2. tételben, ahol a gépek száma most  $\overline{M}$ -al egyenlő. □

**3.4.3. Tétel.** *Annak eldöntése, hogy rögzített  $P \in \mathbb{N}^+$  esetén WIDTH modellben egy dRMT feladat esetén létezik-e végrehajtható  $P$ -periodikus ütemezés,  $\mathcal{NP}$ -teljes probléma. Továbbá nem létezik  $3/2$ -nél jobb közelítő algoritmus, kivéve ha  $\mathcal{P} = \mathcal{NP}$ .*

*Bizonyítás.* Tegyük fel, hogy csak match csúcsaink vannak és  $\Delta M = 1$ . A kapott feladat ekvivalens az RMT problémakör 1D1R modelljével [15]. □

**3.4.4. Tétel.** *Annak eldöntése, hogy rögzített  $P \in \mathbb{N}^+$  esetén az IPC1 és WIDTH-IPC1 modellekben egy dRMT feladat esetén létezik-e végrehajtható  $P$ -periodikus ütemezés,  $\mathcal{NP}$ -teljes probléma. Továbbá nem létezik  $4/3$ -nál jobban közelítő algoritmus az  $OPT$ -ra nézve, kivéve ha  $\mathcal{P} = \mathcal{NP}$ .*

*Bizonyítás.* Feltehető, hogy csak match csúcsok vannak és  $W \equiv 1$ . Ezért és az  $IPC=1$  feltétel miatt minden órajelben legfeljebb egy ütemezés fordulhat elő. Ebből következik, hogy olyan ütemezést szeretnénk találni, amely legfeljebb  $P$  órajelet felhasználva végez. (Egy ütemezés akkor használ fel egy órajelet, ha van abban az

órajelben legalább egy csúcsa.) Ha nem találunk ilyen végrehajtható  $P$  periodikus ütemezést. Ha találunk egy ilyen, abból már könnyen készíthetünk  $P$ -periodikus ütemezést (például a 3.5.3. lemma ezt mondja ki).

Ezzel a problémát visszavezettük az  $P|prec, p_j = 1|C_{max}$ , ami visszavezethető a KLIKK problémára. [12, 14]  $\square$

A nem megközelíthetőségben a WIDTH-IPC1 modellben tudunk még javítani a következőképpen:

**3.4.5. Tétel.** *A WIDTH-IPC1 modell esetén nem létezik  $3/2$ -nél jobban közelítő algoritmus az OPT-ra nézve, kivéve ha  $\mathcal{P} = \mathcal{NP}$ .*

*Bizonyítás.* Feltehető, hogy  $\Delta M = 1$ . Ugyanúgy, mint a 3.4.3. tétel bizonyításában, most elegendő egy ütemezés találása, ami minél kevesebb órajelet használ fel. Ez pedig ekvivalens az RMT problémakör 1D1R modelljével [15].  $\square$

## 3.5. Approximációs algoritmusok

Ebben a szekcióban megadunk egy 3- és 4-közelítő algoritmust az IPC1 és WIDTH-IPC1 modellben. Ehhez először egy új fogalomra, az előütemezés definiálására lesz szükségünk.

### 3.5.1. Előütemezés

**3.5.1. Definíció.** *Jelölje  $A$  és  $M$  az  $S$  ütemezésben azon órajelek számát, ahol létezik egy action illetve match csúcs. Jelölje  $N$  pedig azon órajelek számát, ahol létezik egy match vagy action csúcs. Formálisan*

$$\begin{aligned} A &= A(S) := \#\{n \mid \exists v_a \in V_a : n = S(v_a)\} \\ M &= M(S) := \#\{n \mid \exists v_m \in V_m : n = S(v_m)\} \\ N &= N(S) := \#\{n \mid \exists v \in V : n = S(v)\} \end{aligned}$$

Ha a szövegkörnyezetből egyértelműen kiderül, hogy mire vonatkozik, akkor  $A$ ,  $M$ ,  $N$ -nek fogjuk csak rövidíteni.

Minden megengedett ütemezéshez hozzá szeretnénk rendelni egy olyan úgynevezett előütemezést, amiben nem szerepelnek szünetek, azaz addig nincs üres járatú (action és match csúcs nélküli) órajel, amíg az összes match és action csúcs nem szerepelt.

**3.5.2. Definíció.** Egy  $S$  ütemezést nevezzünk **előütemezésnek**, ha  $A(S) + M(S) = N(S)$ , továbbá létezik olyan  $f : \{1, 2, \dots, N(S)\} \rightarrow \mathbb{N}^+$  szigorúan monoton növekvő függvény és  $S'$  megengedett ütemezés, hogy  $f(S(v)) = S'(v)$  minden  $v \in V$ -re. Ekkor  $S'$  ütemezést  $S$  ütemezés **kiterjesztésének** nevezzük.

Az előütemezés során nem engedjük meg, hogy egy órajelbe match és action csúcs is legyen. Vegyük észre, hogy véges sok előütemezés van.

**3.5.3. Lemma.** Legyen  $S$  egy előütemezés, legyen  $IPC = 1$  és legyen  $P := \max(A, M)$ . Ekkor  $P$  az a legkisebb egész szám, amire tudunk konstruálni egy  $S'$  végrehajtható  $P$ -periodikus ütemezést, amely az  $S$  előütemezés kiterjesztése.

*Bizonyítás.* Meghatározzuk az  $f(1), \dots, f(N)$  értékeit. Legyen  $f(1) := 1$ . Ha létezik egy él csúcs befoka  $S$ -ben nem nulla, akkor  $\delta := \max\{f(S(v)) + l(v) \mid vw \in E, S(w) = i\}$ , egyébként  $\delta := f(i - 1) + 1$ .

$$f(i) := \min\{k \geq \delta \mid \nexists j < i : f(j) \equiv k \pmod{P} \text{ és } S^{-1}(i), S^{-1}(j) \text{ ugyanolyan típusú}\}$$

Vegyük észre, hogy az  $f(i)$  minimumában szereplő halmaz nem az üres halmaz, hiszen  $P \geq M$  és  $P \geq A$ , így a korábbi órajelekben az azonos típusú csúcsok nem tudják lefedni az összes  $P$  modulo osztályt.

Az  $IPC = 1$  korlát miatt az  $A$  és  $M$  értékek alsó korlátok, mert egy előütemezés kiterjesztésében ugyanannyi órajel tartalmaz match és action csúcsot.  $\square$

### 3.5.2. Eredmények

**3.5.4. Tétel.** A  $WIDTH-IPC1$  modellben létezik 4-approximációs algoritmus az  $OPT_1$ -re nézve, melynek futási ideje  $O(|V| \log(|V|) + |E|)$ .

*Bizonyítás.* A 3.5.3. lemmát felhasználva a célunk az, hogy megadjunk egy előütemezést, ahol minimalizáljuk a  $\max(A, M)$  kifejezést, hiszen a lemma értelmében ezután már tudunk  $\max(A, M)$ -periodikus ütemezést készíteni. Megadjunk egy mohó algoritmust, amiről belátjuk, hogy 4-approximációs algoritmus az előbbi kifejezésre nézve.

Az algoritmus a csúcsoknak egy  $V'$  részhalmazából fogja beágyazni a csúcsokat az órajelekbe. Kezdetben legyen  $V' := V$ . Jelölje  $m$  és  $a$  azon match és action csúcsoknak listáját, melyeknek a befoka 0 a  $V'$  által kifeszített részgráfban, továbbá ezen halmazok legyenek szélesség szerint csökkenő sorrendbe rendezve. Egy fázisban

az algoritmus ágyazzon be néhány csúcsot  $m$  és  $a$  listákból egy vagy két órajelbe, majd ugorjon a következő órajelre és kezdje előlről a fázist miután az  $m$  és  $a$  listákat frissítette. Jelölje  $i$  a legkisebb üres órajelet.

Jelölje  $w_m$  és  $w_a$  a csúcsok szélességeinek összegét rendre  $m$ -ban és  $a$ -ban. Egy fázisban csináljuk a következőt:

Ha  $w_m < \overline{M}/2$  és  $w_a < \overline{A}/2$ , ágyazzuk be az összes csúcsot  $m$ -ból az  $i$ -ik órajelbe, az összes csúcsot  $a$ -ból az  $i+1$ -ik órajelbe. Ugorjunk a következő órajelre:  $i := i+2$ .

Ha  $w_m \geq \overline{M}/2$  és  $w_a < \overline{A}/2$ , akkor ágyazzuk be mohón  $m$  csúcsait, amíg csak lehet, és ugorjunk a következő fázisba és a következő órajelre.

Hasonlóan, ha  $w_m < \overline{M}/2$  és  $w_a \geq \overline{A}/2$ , ágyazzuk be mohón  $a$  csúcsait, amíg csak lehet, és ugorjunk a következő fázisba és a következő órajelre.

Végül, ha  $w_m \geq \overline{M}/2$  és  $w_a \geq \overline{A}/2$ , akkor tetszőleges választhatunk  $m$  és  $a$  közül és ágyazzuk be újra mohón az  $i$ -ik órajelbe, úgy mint az előbb.

Lássuk be, hogy ez 4-approximációs algoritmus. Osszuk fel az órajeleket négy csoportra: jelölje  $HF_m$  és  $HF_a$  azon órajeleket, melyek legalább félig tele vannak match, illetve action csúcsokkal. Hasonlóan, jelölje  $NHF_m$  és  $NHF_a$  azon órajeleket, melyek nincsenek félig tele. Vegyük észre, hogy  $|NHF_m| = |NHF_a|$  és  $NHF_m[j] = NHF_a[j] - 1$ .

Feltehető, hogy  $M \geq A$ . A 3.5.3. lemmából azt kapjuk, hogy a kapott  $PS$  előütemezésből konstruálható egy  $S$  végrehajtható  $P$ -periodikus ütemezés.

Jelölje  $P_o$  az optimális periódust a problémára. Tudjuk, hogy  $P_o \geq \sum_{v \in V_m} W(v)/\overline{M}$ . Mivel

$$\sum_{v \in V_m} W(v)/\overline{M} \geq \sum_{v \in PS^{-1}(HF_m)} W(v)/\overline{M} \geq |HF_m|/2,$$

így azt kapjuk, hogy  $|HF_m| \leq 2P_o$ .

**3.5.5. Állítás.** Minden  $v$  beágyazott csúcsra  $NHF_m[i]$ -ben vagy  $NHF_a[i]$ -ben ( $i \geq 2$ ) létezik egy út egy csúcsból a  $NHF_m[i-1]$ -ből vagy  $NHF_a[i-1]$ -ből a  $v$ -be.

*Bizonyítás.* Vizsgáljuk azt a fázist, amikor a  $NHF_m[i-1]$  és  $NHF_a[i-1]$  órajelekbe írunk. Vegyük észre, hogy ekkor minden csúcs  $V'$ -ben elérhető a  $NHF_m[i-1]$ -ből vagy  $NHF_a[i-1]$ -ből.  $\square$

**3.5.6. Állítás.** Létezik egy legalább  $|NHF_m|$  hosszú út a  $D$  gráfban.

*Bizonyítás.* Alkalmazva a 3.5.5. állítást visszafelé tetszőleges  $v_{|NHF_m|}$  beágyazott csúcsra  $NHF_m[|NHF_m|]$ -ben vagy  $NHF_a[|NHF_m|]$ -ben kapható egy út  $v_{|NHF_m|-1}$ -

ből, ami  $NHF_m[|NHF_m| - 1]$ -be vagy  $NHF_a[|NHF_m| - 1]$ -be van beágyazva, stb. Ezen utakat összekapcsolva megkapunk egy  $\mathcal{P}$  utat, amivel az állítást igazoltuk.  $\square$

Vegyük észre, hogy bármilyen  $Q$  útra  $|V(Q) \cap V_m| \leq P_o$  és  $|V(Q) \cap V_a| \leq P_o$  így  $|V(Q)| \leq 2P_o$ . Ezért  $|NHF_m| \leq |V(\mathcal{P})| \leq 2P_o$  és így  $|M| = |HF_m| + |NHF_m| \leq 4P_o$ , ami bizonyítja a tételt.

Könnyen meggondolható, hogy az algoritmus futási ideje  $O(|V| \log |V| + |E|)$ .  $\square$

---

**Algoritmus 1:** WIDTH-IPC1 modell mohó algoritmus (3.5.4. Tétel)

---

**Input:** ODG  $D = (V, E)$ ;  $W : V \rightarrow \mathbb{N}^+$ ;  $\overline{M}, \overline{A}$   
**Output:**  $PS : V \rightarrow \mathbb{N}^+$   
**begin**  
1      $i := 1$ ;  $V' := V$   
2     **while**  $V' \neq \emptyset$  **do**  
3          $a :=$  list of action nodes with 0 indegrees, descending order of width  
4          $m :=$  list of match nodes with 0 indegrees, descending order of width  
5          $w_a :=$  sum of widths in  $a$   
6          $w_m :=$  sum of widths in  $m$   
7         current\_usage := 0  
8         **if**  $w_m \geq 1/2 \overline{M}$  and  $w_a \geq 1/2 \overline{A}$  **then**  
9             Go to line 12 or 19  
10        **if**  $w_a \geq 1/2 \overline{A}$  and  $w_m < 1/2 \overline{M}$  **then**  
11            Go to line 19  
12        **while**  $m[0] + \text{current\_usage} \leq \overline{M}$  **do**  
13            current\_usage +=  $m[0]$   
14             $PS[m[0]] := i$   
15             $V' := V' \setminus \{m[0]\}$   
16             $m := m - m[0]$   
17         $i := i + 1$   
18        **if**  $w_m \geq 1/2 \overline{M}$  **then**  
19            Go to line 12 or 19  
19        **while**  $a[0] + \text{current\_usage} \leq \overline{A}$  **do**  
20            current\_usage +=  $a[0]$   
21             $PS[a[0]] := i$   
22             $V' := V' \setminus \{a[0]\}$   
23             $a := a - a[0]$   
24         $i := i + 1$   
25     **return**  $PS$

---

**3.5.7. Tétel.** A IPC1 modellben létezik 3-approximációs algoritmus, melynek futási ideje  $O(|V| + |E|)$ .

*Bizonyítás.* Lehet ugyanaz az algoritmusunk, mint a 3.5.4. tételben szereplő algoritmus annyi módosítással, hogy nem szükséges a  $w$  és  $a$  elemeket szélesség szerint csökkenő sorrendbe helyeznünk, ezzel a futási időnk lecsökken. Továbbá az  $\overline{M}/2$  és  $\overline{A}/2$  helyett a limitek legyenek  $\overline{M}$  és  $\overline{A}$ , azaz ha meg tudunk telíteni egy órajelet match vagy action csúcsokkal, akkor telítsünk meg és ugorjunk a következő fázisra, egyébként pedig ágyazzuk be az összes match és action csúcsot a következő két órajelbe. Jelöljük  $F_m$ -mel és  $F_a$ -val a tele lévő órajeleket. Ekkor egy élesebb becslést is felírhatunk  $|F_m| \leq P_o$ , ami 3-approximációt ad.  $\square$

**3.5.8. Tétel.** *A WIDTH-IPC2 modellben létezik 8-approximációs algoritmus, melynek futási ideje  $O(|V| \log(|V|) + |E|)$ .*

*Bizonyítás.* Jelölje  $P_1^{opt}$  és  $P_2^{opt}$  az optimális értéket a WIDTH-IPC1 és WIDTH-IPC2 modellekben ugyanazon feladat bemenetek mellett (kivéve IPC értékét). Mivel egy végrehajtható  $P$ -periodikus ütemezés a WIDTH-IPC2 modellben áttanszformálható egy végrehajtható  $2P$ -periodikus ütemezéssé a WIDTH-IPC1 a modellben, így  $P_1^{opt} \leq 2P_2^{opt}$ . Jelölje  $P^*$  a 4-approximációs algoritmus által adott ütemezést WIDTH-IPC1 modellben (3.5.4. Tétel). Ekkor  $P^* \leq 4P_1^{opt} \leq 8P_2^{opt}$ .  $\square$

## 3.6. Több ütemezés beágyazása

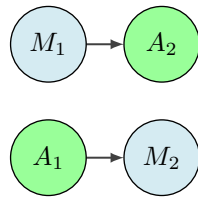
Általánosítsuk a problémát olyan módon, amiből jobban látszik, hogy hogyan lehet processzorunkat a lehető legjobban kihasználni. Ne feltétlenül egy darab ütemezést ismételjünk periodikusan, hanem engedjünk meg több ütemezés periodikus ismétlését is. Sőt, akár el is engedhetjük a periodicitást, azaz feltesszük, hogy végtelen sok ütemezésünk lehet a beágyazásban.

Ha például tekintjük a 3.2. ábrán lévő gráfot, akkor az eddigiek alapján  $P = 2$  lenne a legjobb megoldás. Azonban ha két ütemezést ágyazunk be periodikusan, akkor az optimális megoldás  $P = 4$  helyett  $P = 3$  lesz, ezzel a hatékonyságot 33%-al növelve, ahhoz képest, ha csak egy ütemezést ismételnénk periodikusan.

### 3.6.1. Matematikai fogalmak

**3.6.1. Definíció.** *Ütemezések egy  $k$  elemű halmazát, ahol  $k \in \mathbb{N}^+ \cup \{\infty\}$ ,  $k$ -ütemezésnek nevezünk. Jelölés:  $\mathcal{S}$ . Egy  $\mathcal{S}$   $k$ -ütemezést **megengedett  $k$ -ütemezésnek** hívunk, ha minden  $S \in \mathcal{S}$  ütemezés megengedett ütemezés.*

**3.6.2. Definíció.** *Egy megengedett ütemezést **végrehajtható ütemezésnek** nevezünk, ha minden órajelben a csúcsok kielégítik az erőforrás korlátot. Egy  $\mathcal{S}$  meg-*



(a)

| Órajel:      | 0      | 1                | 2      |
|--------------|--------|------------------|--------|
| 1. ütemezés: | $A_1$  | $M_1$<br>$M_2$   | $A_2$  |
| 2. ütemezés  | $M'_1$ | $A'_1$<br>$A'_2$ | $M'_2$ |

(b)

3.2. ábra.  $P = 4$  helyett  $P = 3$  periódusszám is elérhető ennél a gráfnál, ha két ütemezést ágyazunk be periodikusan.

engedett  $k$ -ütemezést ( $k \in \mathbb{N}^+ \cup \infty$ ) **végrehajtható  $k$ -ütemezésnek** nevezünk, ha minden  $S \in \mathcal{S}$  ütemezés végrehajtható ütemezés, továbbá  $\mathcal{S}$  kielégíti az IPC korlátot.

A dRMT feladat kimenetének egy végrehajtható  $k$ -ütemezésnek kell lennie, ahol  $k \in \mathbb{N}^+ \cup \infty$ . Ahhoz, hogy meg tudjuk fogalmazni a célfüggvényünket, vezessük be az alábbi definíciókat.

**3.6.3. Definíció.** Legyen  $CT(\mathcal{S}, t) := \#\{S \mid S \in \mathcal{S}, \forall v \in V : S(v) + l(v) \leq t\}$  a **végrehajtott feladatok** száma a  $t$ -ik időpillanatban  $\mathcal{S}$ -ben.

Egy végrehajtható ütemezés hatékonyságát definiáljuk a következőképp.

**3.6.4. Definíció.** Legyen  $\mathcal{S}$  egy  $\infty$ -ütemezés. Legyen  $Q(\mathcal{S}) := \limsup_{t \in \mathbb{N}^+} \frac{CT(\mathcal{S}, t)}{t}$  az  $\mathcal{S}$  **hatékonysága**.

Ez lényegében azt fejezi ki, hogy hány feladatot tudunk elvégezni egy időegység alatt, ha kellően sok ideig fut a processzorunk.

A következő definíció a 3.1.2. definíció általánosítása  $k$ -ütemezésekre.

**3.6.5. Definíció.** Legyen  $k, P \in \mathbb{N}^+$  és  $\mathcal{S} = \{S_1, \dots, S_k\}$  egy megengedett  $k$ -ütemezés. Legyen  $\mathcal{S}' := (S_{(n,m)} \mid n \leq k, m \in \mathbb{N})$ , ahol  $S_{(n,m)}(v) := S_n(v) + m \cdot P$ . Ekkor  $\mathcal{S}$  ütemezést **végrehajtható  $P$ -periodikus  $k$ -ütemezésnek** nevezzük, ha  $\mathcal{S}'$  egy végrehajtható  $\infty$ -ütemezés. Továbbá ennek a hatékonysága legyen  $Q(\mathcal{S}, P) := Q(\mathcal{S}')$ .

Vegyük észre, hogy  $Q(\mathcal{S}, P) = \frac{k}{P}$ , hiszen  $\ell \in \mathbb{N}$  esetén  $CT(\mathcal{S}', \ell P) \leq \ell k \leq CT(\mathcal{S}', T(\mathcal{S}) + \ell P)$ , továbbá a  $CT$  függvény monoton nő a második változójában.

A 3.1.4. állítás is általánosítható (lényegében ugyanazzal a bizonyítással) az alábbi módon:

**3.6.6. Állítás.** Legyen  $k, P \in \mathbb{N}^+$  és  $\mathcal{S} = \{S_1, \dots, S_k\}$  egy megengedett  $k$ -ütemezés.  $\mathcal{S}$  pontosan akkor végrehajtható  $P$ -periodikus  $k$ -ütemezés, ha az órajeleket mod  $P$  ekvivalencia osztályokra bontva minden ekvivalencia osztályra teljesül az erőforrás korlát és IPC korlát.

Célunk az, hogy olyan végrehajtható ütemezést találjunk, aminek minél nagyobb a hatékonysága, azaz minél kisebb periódussal rendelkezik. Ezek alapján bevezetjük a következő definíciókat.

**3.6.7. Definíció.**

$$OPT_k := \min \left( \frac{1}{Q(\mathcal{S}, P)} \mid \mathcal{S} \text{ egy végrehajtható } P\text{-periodikus } k\text{-ütemezés} \right)$$

$$OPT_\infty := \inf \left( \frac{1}{Q(\mathcal{S})} \mid \mathcal{S} \text{ egy végrehajtható } \infty\text{-ütemezés} \right)$$

Vegyük észre, hogy az  $OPT_k$  definíciójában a minimum jól definiált, hiszen minden fix  $k$ -ra létezik egy triviális  $k(|V_m| + |V_a|)$ -periodikus  $k$ -ütemezés, és (szintén fix  $k$  esetén) véges sok  $(P, k)$  pár van, ahol  $\frac{k}{P} \geq \frac{1}{|V_m| + |V_a|}$ . Azt is vegyük észre, hogy ez összhangban van a korábbi definícióval, azaz  $OPT = OPT_1$ .

**3.6.8. Megjegyzés.** Ha létezik végrehajtható  $P$ -periodikus  $k$ -ütemezés, akkor  $OPT_\infty \leq P/k$ . Hasonlóan  $\forall k \in \mathbb{N}^+ : OPT_\infty \leq OPT_k$ .

### 3.6.2. Bonyolultságok

A következő tétel a 3.4.1. tétel általánosítása  $k$ -ütemezésekre.

**3.6.9. Tétel.** A BASIC modellben

$$OPT_k = \frac{\max \left( \left\lceil \frac{k|V_m|}{M} \right\rceil, \left\lceil \frac{k|V_a|}{A} \right\rceil \right)}{k},$$

ha  $k \in \mathbb{N}^+$ , illetve

$$OPT_\infty = \max \left( \frac{|V_m|}{M}, \frac{|V_a|}{A} \right).$$

Továbbá  $k \in \mathbb{N}^+$  esetén megadható egy  $OPT_k$ -periodikus  $k$ -ütemezés  $O(|E| + k|V|)$  időben.

*Bizonyítás.* Az  $OPT_k$  pontos értékének bizonyítása megegyezik az  $OPT_1$  pontos értékének bizonyításával (a 3.4.1. tétel), annyi a különbség, hogy most a  $k$  darab ütemezést vesszük sorban és ágyazzuk be mohón az órajelekbe.

Legyen  $k := \overline{M} \cdot \overline{A}$ , ekkor  $OPT_\infty \leq OPT_k = \max\left(\frac{|V_m|}{\overline{M}}, \frac{|V_a|}{\overline{A}}\right)$ .

Az  $OPT_\infty$  alsó becsléséhez vegyük észre, hogy  $t$  órajel alatt a match csúcsok szélességeinek összege maximum  $t\overline{M}$  lehet, az action csúcsoknál pedig  $t\overline{A}$ . Ebből azt kapjuk, hogy

$$CT(\mathcal{S}, t) \leq \min\left(\left\lfloor \frac{t\overline{M}}{|V_m|} \right\rfloor, \left\lfloor \frac{t\overline{A}}{|V_a|} \right\rfloor\right),$$

amiből

$$OPT_\infty = \frac{1}{\limsup_t \frac{CT(\mathcal{S}, t)}{t}} \geq \max\left(\frac{|V_m|}{\overline{M}}, \frac{|V_a|}{\overline{A}}\right).$$

□

Az  $IPC = 1$  modellek (IPC1, WIDTH-IPC1) esetén is átmegy a bizonyítása a 3.4.4. és 3.4.5. tételeknek, ha egy fix  $k \in \mathbb{N}^+$  esetén az  $OPT_k$  értékét tekintjük.

**3.6.10. Tétel.** *Annak eldöntése, hogy rögzített  $k, P \in \mathbb{N}^+$  esetén az IPC1 modellben egy dRMT feladat esetén létezik-e végrehajtható  $P$ -periodikus  $k$ -ütemezés,  $\mathcal{NP}$ -teljes probléma. Annak eldöntése is  $\mathcal{NP}$ -teljes, hogy létezik-e  $\mathcal{S}$  végrehajtható  $\infty$ -ütemezés, amelyre  $Q(\mathcal{S}) \leq \frac{k}{P}$ . Továbbá  $\ell \in \mathbb{N}^+ \cup \{\infty\}$  esetén nem létezik  $4/3$ -nál jobban közelítő algoritmus az  $OPT_\ell$ -re nézve, kivéve ha  $\mathcal{P} = \mathcal{NP}$ .*

**3.6.11. Tétel.**  $\ell \in \mathbb{N}^+ \cup \{\infty\}$  esetén WIDTH-IPC1 modell esetén nem létezik  $3/2$ -nél jobban közelítő algoritmus az  $OPT_\ell$ -re nézve, kivéve ha  $\mathcal{P} = \mathcal{NP}$ .

### 3.6.3. Periodicitás

Célunk, hogy belássuk a 3.6.20. tételt. Ehhez számos definícióra és lemmára lesz szükségünk. Először is bizonyos ütemezéseket ekvivalensnek akarunk tekinteni, így vezessük be a következő definíciókat:

**3.6.12. Definíció.** *Egy  $\mathcal{S}$  végrehajtható ütemezést **minimális ütemezésnek** nevezünk, ha nem létezik olyan  $\mathcal{S}'$  végrehajtható ütemezés, amire  $A' \leq A, M' < M$  vagy  $A' < A, M' \leq M$ .*

**3.6.13. Definíció.** *Legyen  $A, M \in \mathbb{N}^+$ . Ekkor a*

$$C(A, M) := \left\{ \mathcal{S}' \text{ végrehajtható ütemezés} \mid A' = A, M' = M \right\}$$

*halmazt **ütemezési osztálynak** (vagy röviden csak osztálynak) nevezzük. Azt mondjuk, hogy az  $\mathcal{S}$  ütemezés használja az  $C(A, M)$  osztályt, ha  $\mathcal{S} \cap C(A, M) \neq \emptyset$ .*

Vegyük észre, hogy véges sok minimális ütemezési osztály.

Azt mondjuk, hogy  $\mathcal{S}$  végrehajtható  $k$ -ütemezés **használja** az  $S$  előütemezést, ha létezik olyan  $S' \in \mathcal{S}$  ütemezés, az  $S$  előütemezése kiterjesztése. Így már a 3.5.3. lemma is általánosítható az alábbi módon:

**3.6.14. Lemma.** *Legyenek  $S_1, \dots, S_k$  előütemezések, legyen  $IPC = 1$  és legyen*

$$P := \max \left( \sum_{i=1}^k A_i, \sum_{i=1}^k M_i \right).$$

*Ekkor  $P$  az a legkisebb egész szám, amire tudunk konstruálni egy  $\mathcal{S}$  végrehajtható  $P$ -periodikus  $k$ -ütemezést, amely csak az  $S_1, \dots, S_k$  előütemezéseket használja, ráadásul mindegyiket pontosan egyszer.*

*Bizonyítás.* Vegyük sorban  $S_1, \dots, S_k$  előütemezéseket és csináljuk ugyanazt, mint a 3.5.3. lemmában.  $\square$

Ha adottak az  $S_1, S_2$  előütemezések úgy, hogy  $A_1 \leq M_1 \leq M_2$ , akkor tudunk konstruálni egy végrehajtható  $M_1$ -periodikus 1-ütemezést, azonban az utóbbi 3.6.14. lemma miatt nem tudunk konstruálni  $P$ -periodikus  $k$ -ütemezést úgy, hogy  $\frac{P}{k} \leq M_1$ , amely kizárólag az  $S_1, S_2$  előütemezéseket használja. Így elegendő az alábbi esetet vizsgálni:

**3.6.15. Lemma.** *Legyen  $S_1, S_2$  előütemezés és tegyük fel, hogy  $A_1, M_2 < M_1$  és  $A_1, M_2 < A_2$ . Legyen*

$$P := \frac{A_2 M_1 - A_1 M_2}{\text{lko}(M_1 - A_1, A_2 - M_2)},$$

*illetve*

$$k := \frac{M_1 - A_1 + A_2 - M_2}{\text{lko}(M_1 - A_1, A_2 - M_2)}.$$

*Tudunk olyan végrehajtható  $P$ -periodikus  $k$ -ütemezést konstruálni, mely kizárólag az  $S_1$  és  $S_2$  előütemezéseket használja, azonban nincs olyan végrehajtható ütemezés, melynek a hatékonysága  $k/P$ -nél jobb lenne. Továbbá, minden végrehajtható  $P'$ -periodikus  $k'$ -ütemezésnél, aminek a hatékonysága  $k/P$  és kizárólag az  $S_1$  és  $S_2$  előütemezéseket használja,  $P' = nP$  és  $k' = nk$ , ahol  $n \in \mathbb{N}^+$ .*

*Bizonyítás.*  $P \in \mathbb{N}^+$ , mert

$$P = \frac{(M_1 - A_1)M_2 + (A_2 - M_2)M_1}{\text{lko}(M_1 - A_1, A_2 - M_2)} =$$

$$= \frac{(M_1 - A_1)A_2 + (A_2 - M_2)A_1}{\text{luko}(M_1 - A_1, A_2 - M_2)}.$$

Vegyünk  $S_1$ -nek  $\frac{A_2 - M_2}{\text{luko}(M_1 - A_1, A_2 - M_2)}$  darab, míg  $S_2$ -nek  $\frac{M_1 - A_1}{\text{luko}(M_1 - A_1, A_2 - M_2)}$  darab példányát. A 3.6.14. lemma miatt tudunk egy  $\mathcal{S}$  végrehajtható  $P$ -periodikus  $k$ -ütemezést konstruálni, hiszen azt kapjuk, hogy  $M = P = A$ .

Vizsgáljuk azt az esetet, amikor  $S_1$ -nek  $c_1$  példányát,  $S_2$ -nek pedig  $c_2$  példányát vesszük. Szimmetria okok miatt feltehetjük, hogy

$$c_1 A_2 + c_2 A_1 < c_2 M_1 + c_1 M_2.$$

Azt kell belátnunk, hogy

$$\frac{P}{k} \leq \frac{\max(c_1 A_2 + c_2 A_1, c_2 M_1 + c_1 M_2)}{c_1 + c_2} = \frac{c_2 M_1 + c_1 M_2}{c_1 + c_2}.$$

Helyettesítsük be a  $P$  és  $k$  definícióját, szorozzuk meg mindkét oldalt a nevezővel, majd egyszerűsítsünk. Átrendezés után azt kell kapnunk, hogy

$$(A_2 - M_2)(M_1 - M_2)c_1 \leq (M_1 - A_1)(M_1 - M_2)c_2.$$

Mivel  $M_2 > M_1$  ezért leoszthatjuk mindkét oldalt  $M_1 - M_2$ -vel. Átrendezés után megkapjuk a feltételezésünket. Egyenlőség pedig pontosan akkor van, ha  $c_1$  és  $c_2$  az  $\frac{A_2 - M_2}{\text{luko}(M_1 - A_1, A_2 - M_2)}$  és  $\frac{M_1 - A_1}{\text{luko}(M_1 - A_1, A_2 - M_2)}$  többszöröse.  $\square$

**3.6.16. Lemma.** *Legyen  $\mathcal{S}$  egy végrehajtható  $P$ -periodikus  $k$ -ütemezés. Ekkor tudunk konstruálni egy  $\mathcal{S}'$  végrehajtható  $P$ -periodikus  $k$ -ütemezést, amely csak minimális ütemezésekből áll.*

*Bizonyítás.* Minden nem minimális előütemezést cseréljünk ki minimális előütemezésre. A 3.6.14. lemma miatt ez is egy végrehajtható  $P$ -periodikus  $k$ -ütemezés lesz, ugyanis

$$\max \left( \sum_{i=1}^k A_i, \sum_{i=1}^k M_i \right) \geq \max \left( \sum_{i=1}^k A'_i, \sum_{i=1}^k M'_i \right)$$

és ha tudunk ütemezést konstruálni  $P$  periódussal, akkor minden  $P' > P$ -re szintén tudunk ütemezést konstruálni  $P'$  periódussal.  $\square$

**3.6.17. Lemma.** *Legyen  $\mathcal{S}_1$  egy végrehajtható  $P_1$ -periodikus  $k_1$ -ütemezés, amely  $c > 2$  darab minimális ütemezési osztályt használ. Ekkor tudunk konstruálni egy  $\mathcal{S}_2$  végrehajtható  $P_2$ -periodikus  $k_2$ -ütemezést, amely legfeljebb  $c - 1$  minimális ütemezési osztályt használ, továbbá  $P_1/k_1 \geq P_2/k_2$ .*

*Bizonyítás.* Tegyük fel, hogy minden minimális ütemezési osztályra  $A \leq M$ . Ekkor legyen  $S$  egy ütemezés abból az osztályból, ahol  $M$  minimális. Az  $IPC = 1$  feltétel miatt teljesülnie kell annak, hogy  $M \leq P_1/k_1$ . Azonban a 3.6.14. lemma miatt konstruálható egy  $S$   $M$ -periodikus 1-ütemezés. Ez kizárólag egy osztályból tartalmaz ütemezést. Amennyiben  $A \geq M$  minden osztályra, akkor ugyanez az érvelés elmondható.

Így feltehetjük, hogy létezik két minimális ütemezési osztály, melyre  $A_1 < M_1$  illetve  $A_2 > M_2$ . Legyen  $f_1 := A_2 - M_2$ ,  $f_2 := M_1 - A_1$ , és legyen  $S_i$  egy minimális ütemezés az  $i$ -ik osztályból. Vegyünk  $S_1$ -nek  $f_1$  darab,  $S_2$ -nek  $f_2$  darab példányát. A 3.6.14. lemma miatt tudunk egy  $\mathcal{S}_2$  végrehajtható  $P_2$ -periodikus  $k_2$ -ütemezést konstruálni, ahol  $P_2 := f_1 M_1 + f_2 M_2 = A_2 M_1 - A_1 M_2 = f_1 A_1 + f_2 A_2$  és  $k_2 := f_1 + f_2$ .

Ha  $P_1/k_1 \geq P_2/k_2$ , akkor készen vagyunk, mert  $\mathcal{S}_2$  két osztályból tartalmaz csak ütemezést. Tegyük fel, hogy  $P_1/k_1 < P_2/k_2$ . Ekkor  $e_i$  jelölje azt, hogy hány ütemezést tartalmaz  $\mathcal{S}_1$  az  $i$ -ik osztályból, ahol  $1 \leq i \leq c$ . (Azaz  $\sum_{i=1}^c e_i = k_1$ .) Tegyük fel, hogy  $e_1 f_2 \leq e_2 f_1$ . (A másik esetben ugyanez az érvelés elmondható lesz.)  $S_1$ -nek vegyünk 0 darab,  $S_2$ -nek  $e_2 f_1 - e_1 f_2$  darab,  $S_i$ -nek  $3 \leq i \leq c$  esetén  $f_1 e_i$  darab példányát. A 3.6.14. lemma miatt tudunk konstruálni egy  $\mathcal{S}_3$   $P_3$ -periodikus  $k_3$ -ütemezést, ahol  $P_3 := f_1 P_1 - e_1 P_2$  és  $k_3 := f_1 k_1 - e_1 f_1 - e_1 f_2$ .

Azt kell belátnunk, hogy

$$\frac{P_1}{k_1} \geq \frac{P_3}{k_3} = \frac{f_1 P_1 - e_1 P_2}{f_1 k_1 - e_1 f_1 - e_1 f_2}$$

ami igaz, mert átrendezés és egyszerűsítés után azt kapjuk, hogy  $P_1/k_1 < P_2/k_2$ .

$\mathcal{S}_3$  legfeljebb  $c - 1$  osztályból tartalmaz ütemezést, mert 0 példányt vettünk  $S_1$ -ből, így készen vagyunk.  $\square$

**3.6.18. Megjegyzés.** *Nem precízen megfogalmazva vettük  $\mathcal{S}_1$ -nek  $f_1$  példányát, és kitöröltünk  $\mathcal{S}_2$ -nek  $e_1$  darab példányát, aminek a hatékonysága rosszabb volt az  $\mathcal{S}_1$  hatékonyságánál.*

**3.6.19. Következmény.** *A WIDTH-IPC1 modellben minden dRMT problémára a  $K := \operatorname{argmin}_{k \in \mathbb{N}^+} OPT_k$  jól definiált, illetve  $OPT_K$  megkonstruálható legfeljebb 2 minimális ütemezési osztály segítségével.*

*Bizonyítás.* A 3.6.16. lemma miatt elegendő azon ütemezéseket vizsgálni, melyek kizárólag minimális ütemezési osztályokat használnak, továbbá a 3.6.17. lemma miatt leredukáltuk a problémát arra, amikor amikor legfeljebb két minimális ütemezési osztályt használhatunk. Mivel véges előütemezés illetve előütemezési párok vannak,

továbbá a 3.6.15. lemmából tudjuk az optimális megoldást ezekre, ezért ezen véges sok eset közül kiválasztva a legjobb hatékonyságút megkapjuk a  $K$  értékét, sőt,  $OPT_K$ -ra konstrukciót is kapunk.  $\square$

**3.6.20. Tétel.** *A WIDTH-IPC1 modellben minden dRMT problémára létezik egy  $K$ , hogy  $OPT_K = OPT_\infty$ .*

*Bizonyítás.* Tegyük fel, hogy  $OPT_\infty < OPT_k$  minden  $k \in \mathbb{N}^+$  esetén. Speciálisan  $OPT_\infty < OPT_K$  ahol  $K$  a 3.6.19. következmény szerint legyen definiálva. Az  $OPT_\infty$  definíciója miatt léteznie kell egy  $\mathcal{S}$  végrehajtható  $\infty$ -ütemezésnek úgy, hogy  $\frac{1}{Q(\mathcal{S})} < OPT_K$ . A hatékonyság definíciója miatt létezik egy olyan  $P$ , amely osztható  $K$ -val és  $\frac{CT(\mathcal{S}, P)}{P} < OPT_K$ . Azon ütemezéseket véve, amelyek a  $P$ -ik órajelben készen vannak, konstruálható egy  $P$ -periodikus  $k$ -ütemezés, aminek a hatékonyságának reciproka kisebb, mint  $OPT_K$ , ami ellentmond  $K$  definíciójának.  $\square$

Ez jó hír, hogy mindig található egy  $P$ -periodikus  $K$ -ütemezés, amely tényleg az elméleti minimum. Azonban nem jó hír, hogy ez a  $K$  akármilyen nagy is lehet (természetesen gráftól függően), erről szól a következő állítás:

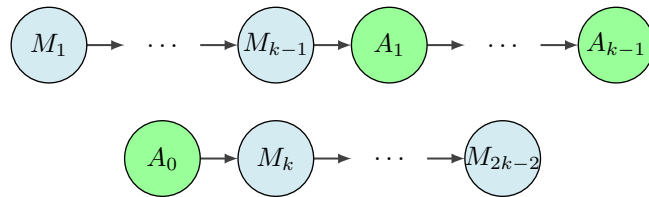
**3.6.21. Állítás.** *Legyen  $\overline{M} = \overline{A} = 2$ ,  $\Delta A = \Delta M = IPC = 1$ . Ekkor minden  $2 \leq k < \infty$  esetén létezik egy  $D = (V, E)$  gráf, ami esetén*

$$OPT_1 = OPT_2 = \dots = OPT_{k-1} = \left(1 + \frac{1}{k^2 - 1}\right) OPT_k.$$

*Bizonyítás.* Vegyük a következő aciklikus gráfot (3.3. ábra):

$V_m := \bigcup_{i=1}^{2k-2} \{M_i\}$ ,  $V_a := \bigcup_{i=0}^{k-1} \{A_i\}$  és

$E := \bigcup_{i=1}^{k-2} \{(M_i, M_{i+1}), (M_{k+i-1}, M_{k+i}), (A_i, A_{i+1})\} \cup \{(M_{k-1}, A_1), (A_0, M_1)\}.$



3.3. ábra.  $3k - 2$  csúcsú gráf, amely esetén  $OPT_1 = \dots = OPT_{k-1} > OPT_k$

Ekkor két minimális ütemezési osztály van:  $S_1$  és  $S_2$ , ahol  $A(S_1) = k-1$ ,  $M(S_1) = 2k-2$  és  $A(S_2) = k$ ,  $M(S_2) = k-1$ . (3.4. táblázat)

A 3.6.15. lemmából tudjuk, hogy  $OPT_k = k - \frac{1}{k}$ , amit úgy kaphatunk, hogy egyszer az első és  $k-1$ -szor a második minimális ütemezési osztályt vesszük. Ezzel szemben könnyen ellenőrizhető, hogy  $OPT_1 = \dots = OPT_{k-1} = k$ .  $\square$

| Órajel: | 1     | 2              | ... | k-1                     | k                       | k+1            | ... | 2k-2                    | 2k-1       |
|---------|-------|----------------|-----|-------------------------|-------------------------|----------------|-----|-------------------------|------------|
| $S_1$ : | $M_1$ | $M_2$          |     | $M_{k-1}$               | $A_0$<br>$A_1$          | $A_2$<br>$M_k$ |     | $A_{k-1}$<br>$M_{2k-3}$ | $M_{2k-2}$ |
| $S_2$ : | $A_0$ | $M_1$<br>$M_k$ |     | $M_{k-2}$<br>$M_{2k-3}$ | $M_{k-2}$<br>$M_{2k-2}$ | $A_1$          |     | $A_{k-2}$               | $A_{k-1}$  |

3.4. táblázat. A két minimális ütemezési osztály a 3.3. ábra esetén.

**3.6.22. Állítás.** *Mindegyik modellben minden dRMT problémára  $OPT_\infty \leq OPT_k \leq OPT_1$  tetszőleges  $k \in \mathbb{N}^+$  esetén.*

*Bizonyítás.*  $OPT_1$ -hez tartozik egy  $\mathcal{S}$   $OPT_1$ -periodikus 1-ütemezés, ezt  $k$ -szor megismételve tudunk készíteni egy  $\mathcal{S}'$   $k \cdot OPT_1$ -periodikus  $k$ -ütemezést tetszőleges  $k \in \mathbb{N}^+$ -ra, így  $OPT_k \leq OPT_1$ .  $OPT_\infty \leq OPT_k$  nyilvánvaló, hiszen minden  $k$ -ütemezés definiál egy  $\infty$ -ütemezést, melynek hatékonysága ugyanannyi, mint a  $k$ -ütemezése.  $\square$

**3.6.23. Állítás.** *A WIDTH-IPC1 modellben minden dRMT problémára  $2 \cdot OPT_\infty \geq OPT_1$ .*

*Bizonyítás.* Vegyünk egy tetszőleges  $\mathcal{S}$   $\infty$ -ütemezést. Ekkor az  $IPC = 1$  feltétel miatt minden  $S \in \mathcal{S}$ -re  $A(S) \geq OPT_1$  vagy  $M(S) \geq OPT_1$ , különben létezne  $OPT_1$ -nél kisebb periodikus 1-ütemezés. Ebből viszont következik, hogy  $2 \cdot \frac{t}{OPT_1} \geq CT(\mathcal{S}', t)$ , ugyanis az  $IPC = 1$  feltétel miatt egy órajelben legfeljebb egy ütemezés foglalkozhat match csúccsal és legfeljebb ütemezés foglalkozhat action csúccsal.  $\square$

## 4. fejezet

# Szimulációk

A [4] cikkben található egy szimulációs program példákkal együtt, ami segítségével létre lehet hozni WIDTH-IPC1 és WIDTH-IPC2 dRMT ütemezéseket.

A szimulációjuk röviden összefoglalva a következőképp néz ki: Rögzítenek egy  $P$  periódus értéket, és egy végrehajtható ütemezést keresnek minimalizálva a beágyazás hosszát,  $T$ -t. Lefuttatnak többféle heurisztikát erre a  $P$  értékre, így egy végrehajtható, vagy majdnem végrehajtható ütemezést kapva. Ezután felírnak egy IP feladatot, aminek a kezdeti értékének beadják a legjobb heurisztikát, majd egy IP megoldón futtatják a feladatot egy ideig. (Kizárólag olyan megoldást keresnek, ahol az ütemezés hossza nem nagyobb, mint a kritikus út hosszának másfélszerese.) Ha nem talál végrehajtható ütemezést, akkor nagyobb  $P$ -vel próbálkozik, ha talál végrehajtható ütemezést, akkor kisebb  $P$ -vel próbálkozik. A megfelelő  $P$  értéket pedig bináris kereséssel keresik meg. Kezdetben  $P$ -re megadható egy triviális alsó korlát, amit a szélességek összege és  $\overline{A}, \overline{M}$  értékek határoznak meg.

A dolgozatban kétféle javítással is prezentálunk: egyrésztől megadtunk egy jobb heurisztikát (3.5.4), amiről bizonyított, hogy konstans közelítő algoritmus. Másrésztől ha csak annyi a célunk, hogy egy végrehajtható ütemezést keressünk, a  $T$  hossza nem annyira számít, akkor arra jobb módszer is van. Az IP felírás során, (vagy az rnd\_sieve algoritmus során) ha feltesszük, hogy  $\Delta A = \Delta M = 1$ , akkor lényegesen kevesebb paramétere lesz az IP felírásnak, így az IP megoldó is gyorsabb lesz.

A futási időket egy közönséges számítógépen mértük 64 GB RAM-mal 24 szálon 2,40 GHz-en. A programkód elérhetőek az alábbi oldalon: <https://github.com/fraknoiadam/drmt>.

A [4] cikk programkódjában számos módosítást végre kellett hajtani, hogy megkapjuk a kívánt eredményeket, továbbá implementáltam a 3.5.4. tételben szereplő algoritmust.

Számos szimulációt három valódi, switch.p4 [2] programból generált gráfon vizsgáltuk: Ingress, Egress és Combined (Ingress+Egress), ahogy azt a [4] cikkben is használták.

## 4.1. Ütemezés hossza

Megvizsgáltuk, hogy a három gráfon különböző  $P$  értékek mellett milyen végrehajtható ütemezést lehet generálni, aminek hossza ( $T$ ) minél rövidebb, és ezt mennyi idő alatt lehet megtalálni. Az eredmények megtalálhatók a 4.1. táblázatban. Azt tapasztaltuk, hogy futási idők nem növekedtek meg drasztikusan az Egress gráfot kivéve. Illetve azt a következtetést vonhatjuk le, hogy az alacsonyabb  $P$  érték nincs szignifikáns hatással a legkisebb hosszra, ami elérhető.

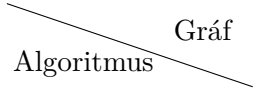
|                 | Egress |     |     | Ingress |     |     | Combined |     |     |
|-----------------|--------|-----|-----|---------|-----|-----|----------|-----|-----|
| P               | 11     | 12  | 13  | 17      | 18  | 19  | 21       | 22  | 23  |
| optimális $T_P$ | 217    | 208 | 206 | 245     | 246 | 244 | 243      | 244 | 243 |
| idő [mp]        | 1203   | 76  | 107 | 106     | 59  | 23  | 118      | 25  | 109 |

4.1. táblázat.  $P$  vs  $T$ : Alacsonyabb  $P$  periódusérték érték nem okoz magasabb  $T_P$  hossz értéket.

## 4.2. $P$ periódus minimalizálása

Ebben a fejezetben összehasonlítjuk a különböző algoritmusok milyen jó  $P$  periódus értékeket találnak meg. Minél kisebb  $P$  periódus minél nagyobb hatékonyságot jelent. Egyrésről vizsgáljuk a [4] cikkben szereplő dRMT modellhez tartozó ILP megoldót, másrésről szintén ebben a cikkben szereplő rnd\_sieve algoritmust, továbbá mi mohó algoritmusunkat (3.5.4. tétel). A legjobb  $P$  értékeket és futási idejüket a 4.2. táblázat tartalmazza. Ugyanezen eredményeket tartalmazza a 4.3. ábra, csak most az ILP megoldó eredményéhez tekintjük, hogy milyen jó teljesítményű a két heurisztika.

**Rnd\_sieve.** Lerögzítenek egy fix  $P$  periódust és itt keresnek egy végrehajtható ütemezést. Egy futtatás során vesznek egy topologikus sorrendet, majd egyesével megpróbálják beágyazni a csúcsokat  $P$ -periodicitással. Mindig veszik a legkisebb órajelet (vagy időnként az utána lévő), ahová beágyazhatják a csúcsot. Ezen kívül több trükköt bevetnek, hogy minél több véletlent belevigyenek az algoritmusba. Ha

| <div style="text-align: center;">  </div> | Egress      |        | Ingress     |        | Combined     |        |
|----------------------------------------------------------------------------------------------------------------------------|-------------|--------|-------------|--------|--------------|--------|
|                                                                                                                            | $ V  = 104$ |        | $ V  = 224$ |        | $ V  = 328$  |        |
|                                                                                                                            | $ E  = 291$ |        | $ E  = 930$ |        | $ E  = 1221$ |        |
|                                                                                                                            | P           | t [mp] | P           | t [mp] | P            | t [mp] |
| [4] rnd_sieve                                                                                                              | 13          | 0,36   | 21          | 1,78   | 28           | 1,65   |
| 3.5.4. Tétel algoritmus                                                                                                    | 13          | 0,007  | 19          | 0,28   | 23           | 10,5   |
| [4] ILP                                                                                                                    | 11          | 0,13   | 17          | 3,4    | 21           | 30,6   |
| ILP által adott alsó becslés                                                                                               | 7           | -      | 15          | -      | 21           | -      |

4.2. táblázat. Legjobb  $P$  értékek és futási idejük különböző algoritmusok esetén

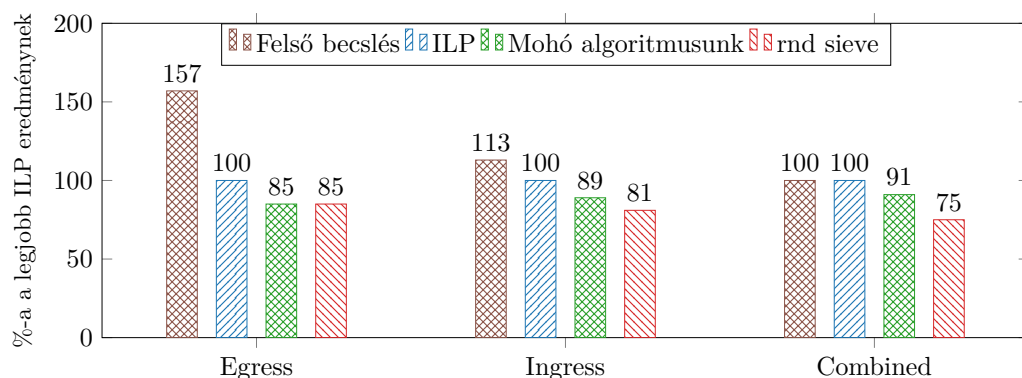
nem akadnak el az erőforrás feltételek miatt, akkor ezzel egy jó beágyazást kaptak. Előnye ennek az algoritmusnak, hogy  $IPC = 2$  esetén is működik. Az algoritmus futtatás során,  $\Delta A = \Delta M = 1$  késleltetési függvényeket vettük, ezzel is gyorsabbá téve az algoritmust.

**Mohó algoritmusunk.** A mi algoritmusunk során nem rögzítjük le  $P$  értékét, hanem azt majd az algoritmus végén kiszámoljuk, hogy milyen  $P$  értéket kapnánk ebből az  $IPC = 1$  korlátot is figyelembe véve. Az algoritmus során itt is lehet használni véletleneket, például topologikus sorrend vételével is. Hátránya az algoritmusunknak, hogy felteszi az  $IPC = 1$  korlátot.

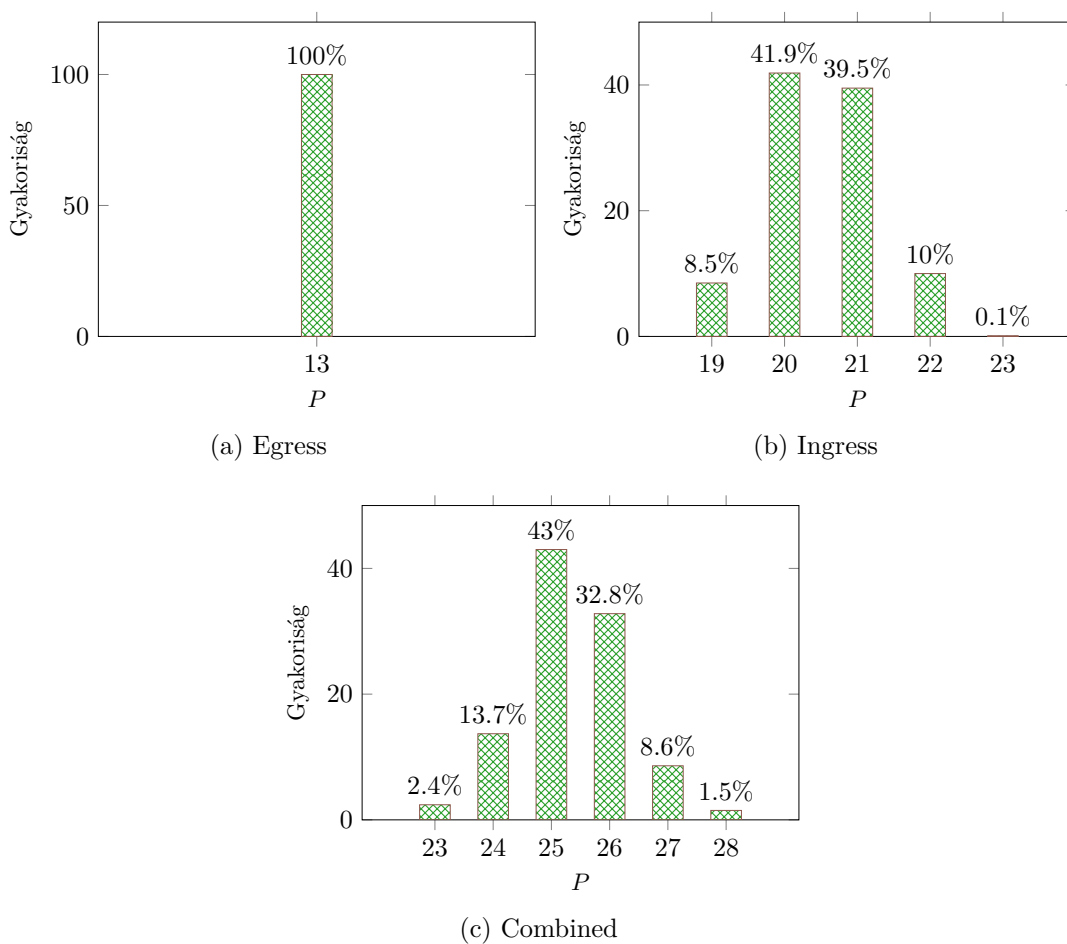
**[4] ILP.** A cikkben kétféle ILP felírást használnak. Egyrésztől megadták a dRMT probléma egy felírását, másrésztől még az RMT problémakörnek is egy felírását, amiből generálható egy dRMT megoldás, habár ez nem feltétlen lesz ugyanolyan jó, mint az előbbi ILP felírás által adott megoldás. (A három gráf esetén mindkettő ILP ugyanazt az optimális  $P$  értéket adta, azonban kisebb gráfoknál az sokszor nem igaz.) Jelen futási idők esetén az RMT-hez tartozó ILP felírást vettem.

Az Egress és Ingress gráfnál is van egy 57%-os és 13%-os lyuk, ahol nem tudjuk az optimális  $P$  értéket. Ennek meghatározása reménytelennek tűnik, az ILP-t futtatva több, mint 24 órán át sem kaptunk jobb alsó vagy felső becsléseket ezen gráfokra.

A mi mohó algoritmusunkat a véletlen eseményekkel felhasználva 1000-szer futtattuk le a három gráfon (Egress, Ingress, Combined). Az eloszlásuk a 4.4. ábrán láthatóak. Az átlagos futási idő 7 ms, 24 ms és 41 ms volt. Ebből azt kapjuk, hogy a legjobb  $P$  érték megtalálása várható értékben 0.007, 0.28 és 10.5 másodperc volt. Ezzel szemben az rnd\_sieve algoritmus átlagos futási ideje a három gráfon, amíg talált egy jó megoldást az adott táblázatban szereplő  $P$  értékre 0.36, 1.78 és 1.65 másodperc volt.



4.3. ábra. Különböző algoritmusok teljesítménye a legjobb ILP megoldásához képest.



4.4. ábra. A 3.5.4. tétel mohó algoritmusunk által adott  $P$  értékek eloszlása az Egress, Ingress és Combined gráfokon 1000 futtatás során. Egy futtatás átlagos futási ideje 7 ms, 24 ms és 41 ms volt.

## 5. fejezet

# Konklúzió, nyitott kérdések

A P4 szekvenciák beágyazása valódi céleszközökre egy kritikus pontja a programozható adatsomag-továbbítás témakörének. Az adatátviteli sebesség és P4 program függvényében a hatékonyság maximalizálása kiemelt fontosságú. A dolgozatban megvizsgáltuk az elméleti hátterét a problémának, meghatároztuk a feladatok bonyolultságát, felső és alsó korlátokat adtunk a különböző modellekben optimális átviteli sebesség közelíthetőségére. Eredményeink hozzájárulhatnak létező P4 fordító teljesítményének növekedéséhez, és segítik hatékony jövőbeli csomagtovábbító programnyelvek és fordítók tervezését. Most létező magasabb szintű nyelvek a  $\mu$ P4 [13] és Lyra [6] egy magasabb szintű absztrakciós szintet nyújtanak a programozóknak, könnyebbé téve azt, hogy a program több eszközön is futhasson.

A dolgozat megírása végén számos további izgalmas nyitott kérdés és kutatási irányt látunk, ezeket alább foglalom össze.

A modellek többségének meghatároztuk a bonyolultsági osztályát, kivéve a WIDTH-IPC2 modellnek. Vajon  $\mathcal{NP}$ -nehéz probléma-e ebben a modellben legkisebb  $P$  periódus megtalálása?

Hasonlóan a modellek többségére adtunk approximációs algoritmust, kivéve a WIDTH modellre. Erre a modellre milyen approximációs algoritmus adható?

Beláttuk, hogy egy ütemezés hosszának ( $T$ ) a minimalizálása is  $\mathcal{NP}$ -nehéz probléma. Ezt a problémakört is tovább lehet vizsgálni, például létezik-e fix  $P$  esetén  $T$ -t közelítő algoritmus a modellekben? Vagy legalább garantálható-e mindegyik modellben, hogy ha a hatékonyságból egy kicsit engedünk, akkor már tudunk  $T$ -re nézve közelítő algoritmust adni?

A 3.6. alfejezetben a WIDTH-IPC1 modell esetén bevezettük a  $k$ -ütemezés fogalmát, ami nagyobb hatékonyság reményében megengedi  $k$  darab ütemezés váltogatását. Itt is számtalan kérdés felvetődik. Egyrészről ezt a fogalmat a többi modellre is

be lehet vezetni. Mit mondhatunk a feladat bonyolultságáról a többi modellben, ha nem az  $OPT_1$ -re vagyunk kíváncsiak, hanem az  $OPT_k$ -ra? Illetve a többi modellben is igaz az, hogy mindig létezik egy  $K$  pozitív egész, hogy  $OPT_K = OPT_\infty$ . Másrészről jelenleg van egy becslésünk az  $OPT_k$  és  $OPT_j$  között, azonban lehet-e ennél élesebb becslést mondani? Például tudunk-e 33%-nál nagyobb hatékonyságnöveledést elérni, ha egy  $\infty$ -ütemezést tekintünk és nem 1-ütemezést, azaz igaz-e az, hogy  $\frac{4}{3} \cdot OPT_\infty \geq OPT_1$ ? Hasonlóan, ha például 100-ütemezés helyett 101-ütemezéseket veszünk, akkor van olyan eset, amikor meg tudjuk lényegesen növelni a hatékonyságot, vagy létezik egy jó alsó korlát, aminél jobban már nem segít a 101-ütemezés.

Ezen kérdések mellett az egyszerűsített modellek vizsgálatának is fontos szerepe van, ugyanis ezzel segíteni tudjuk a hardver tervezőket, mely megkötések okoznak igazából nehézséget, és melyek nem okoznak nehézséget; hogyan érdemes tervezni az architektúrát, hogy utána egy-egy beágyazás elvégzése már könnyebb legyen.

# Irodalomjegyzék

- [1] Programming protocol-independent packet processors. <https://p4.org/>.
- [2] Switch.p4. <https://github.com/p4lang/switch/blob/master/p4src/switch.p4>. Accessed: 2020-08-19.
- [3] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, et al. P4: Programming protocol-independent packet processors. *ACM SIGCOMM CCR*, 44(3):87–95, 2014.
- [4] Sharad Chole, Andy Fingerhut, Sha Ma, Anirudh Sivaraman, Shay Vargaftik, Alon Berger, Gal Mendelson, Mohammad Alizadeh, Shang-Tse Chuang, Isaac Keslassy, et al. dRMT: Disaggregated Programmable Switching. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pages 1–14, 2017.
- [5] Huynh Tu Dang, Marco Canini, Fernando Pedone, and Robert Soulé. Paxos made switch-y. volume 55, pages 19–24. ACM SIGCOMM, 2016.
- [6] Jiaqi Gao, Ennan Zhai, Hongqiang Harry Liu, Rui Miao, Yu Zhou, Bingchuan Tian, Chen Sun, Dennis Cai, Ming Zhang, and Minlan Yu. Lyra: A cross-platform language and compiler for data plane programming on heterogeneous ASICs. In *ACM SIGCOMM*, page 435–450, 2020.
- [7] Intel. Intel Tofino 3: Product Brief. <https://www.intel.com/content/www/us/en/products/network-io/programmable-ethernet-switch/tofino-3-brief.html>. Accessed: 2022-09.
- [8] Lavanya Jose, Lisa Yan, George Varghese, and Nick McKeown. Compiling packet programs to reconfigurable switches. In *USENIX NSDI*, pages 103–115, 2015.

- 
- [9] Naga Katta, Mukesh Hira, Changhoon Kim, Anirudh Sivaraman, and Jennifer Rexford. Hula: Scalable load balancing using programmable data planes. In *ACM SOSR*, pages 1–12, 2016.
  - [10] Changhoon Kim, Anirudh Sivaraman, Naga Praveen Katta, Antonin Bas, Advait Dixit, and Lawrence J Wobker. In-band network telemetry via programmable dataplanes. 2015.
  - [11] Rui Miao, Hongyi Zeng, Changhoon Kim, Jeongkeun Lee, and Minlan Yu. Silkroad: Making stateful Layer-4 load balancing fast and cheap using switching ASICs. In *ACM SIGCOMM*, pages 15–28, 2017.
  - [12] Pinedo Michael L. *Scheduling Theory, Algorithms, and Systems*. Springer, New York, 2012.
  - [13] Hardik Soni, Myriana Rifai, Praveen Kumar, Ryan Doenges, and Nate Foster. Composing dataplane programs with  $\mu$ P4. In *ACM SIGCOMM*, page 329–343, 2020.
  - [14] Jordán Tibor. Ütemezés: elmélet és algoritmusok. <https://web.cs.elte.hu/~jordan/utemezes/UtemezesJegyzet2020okt.pdf>, 2020.
  - [15] Balázs Vass, Erika Bérczi-Kovács, Costin Raiciu, and Gábor Rétvári. *Compiling Packet Programs to Reconfigurable Switches: Theory and Algorithms*, page 28–35. Association for Computing Machinery, New York, NY, USA, 2020.
  - [16] Balázs Vass, Ádám Fraknói, Erika Bérczi-Kovács, and Gábor Rétvári. Compiling packet programs to drmt switches: Theory and algorithms. In *Proceedings of the 5th P4 Workshop in Europe*, EuroP4’22, page 1–7, New York, NY, USA, 2022. Association for Computing Machinery.