NYILATKOZAT

Név: Fraknói Ádám

ELTE Természettudományi Kar, szak: Matematika BSc

NEPTUN azonosító: ASQQJE

Szakdolgozat címe:
Approximációs algoritmusok gráfbeágyazási feladatokra

A **szakdolgozat** szerzőjeként fegyelmi felelősségem tudatában kijelentem, hogy a dolgozatom önálló szellemi alkotásom, abban a hivatkozások és idézések standard szabályait következetesen alkalmaztam, mások által írt részeket a megfelelő idézés nélkül nem használtam fel.

Budapest, 2022. 05. 30.

Fraknói Ádám

a hallgató aláírása

Eötvös Loránd Tudományegyetem
Természettudományi Kar

Fraknói Ádám

Approximációs algoritmusok gráfbeágyazási feladatokra

Szakdolgozat
Matematika BSc

Témavezető:
Bérczi-Kovács Erika

Külső témavezető:
Vass Balázs
BME VIK TMIT

Budapest, 2022

Tartalomjegyzék

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 elkészítésében. 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.

Bevezetés

A kommunikációs hálózatok csomópontjaiban switchek helyezkednek el, melyek elsődleges célja az, hogy a bejövő adatcsomagokat feldolgozza és eldöntse, hogy milyen irányba menjenek továbbább. 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 kívánt viselkedéseket egy speciális programnyelvben 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.

Belátható, hogy ezen gráfbeágyazási feladatok NP-nehéz problémák, így az optimumot polinomiális időben meghatározni reménytelen. 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. 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. Azonban az RMT architektúrának van néhány problémája, melyeket egy új, úgynevezett dRMT architektúra kifejlesztésével próbálták orvosolni, habár gyártásba eddig még nem került. A dolgozatom első felében az RMT architektúrát, második felében a dRMT architektúráját járom körbe. Azt vizsgálom, hogy a gráfbeágyazások különböző feltételek mellett milyen hatékonyan végezhetők el.

Az RMT fejezetben a modelleket és hozzájuk tartozó eredményeket a [6] cikkből vettem. A dRMT fejezetben a modelleket a [2] cikk alapján dolgoztuk ki témavezetőimmel, a benne elért eredményeket és méréseket pedig saját magam készítettem.

1. fejezet

RMT architektúra

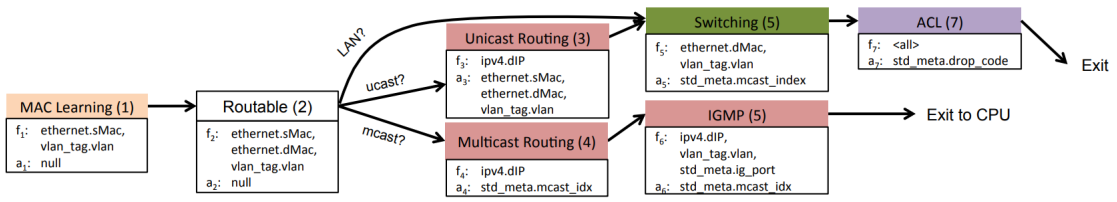
1.1. Technikai háttér

A szükséges hálózati viselkedéseket (pl. adatcsomagok továbbítási portjának kiválasztása, adatfolyam statisztikák készítése stb.) rendszerint egy úgynevezett P4 programnyelvben [1] 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 [4].

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 adatcsomagnak. A táblákban bejegyzések szerepelnek, amik két részből állnak: egyeztetésekből (match) és műveletekből (action). Az előbbiek párhuzamosan keresnek,



1.1. ábra. Egy lehetséges vezérlésfolyamat (control flow). [4]

a legjobb egyezés műveletét végrehajtják. Az utóbbi utasításokat tartalmaz, például 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 az 1.1. ábrán a Unicast Routing táblában lévő feladatokat csak akkor végezhetjük el, ha már végeztünk a Routable-ben lévő összes feladattal.

Ezt a vezérlésfolyamot és a hozzá tartozó körmentes irányított gráfot a P4 fordító az első lépésében el is készíti. Ezután egyrészről 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.) Másrészről pedig megtervezi az adatcsomagok előfeldolgozásának menetét.

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.

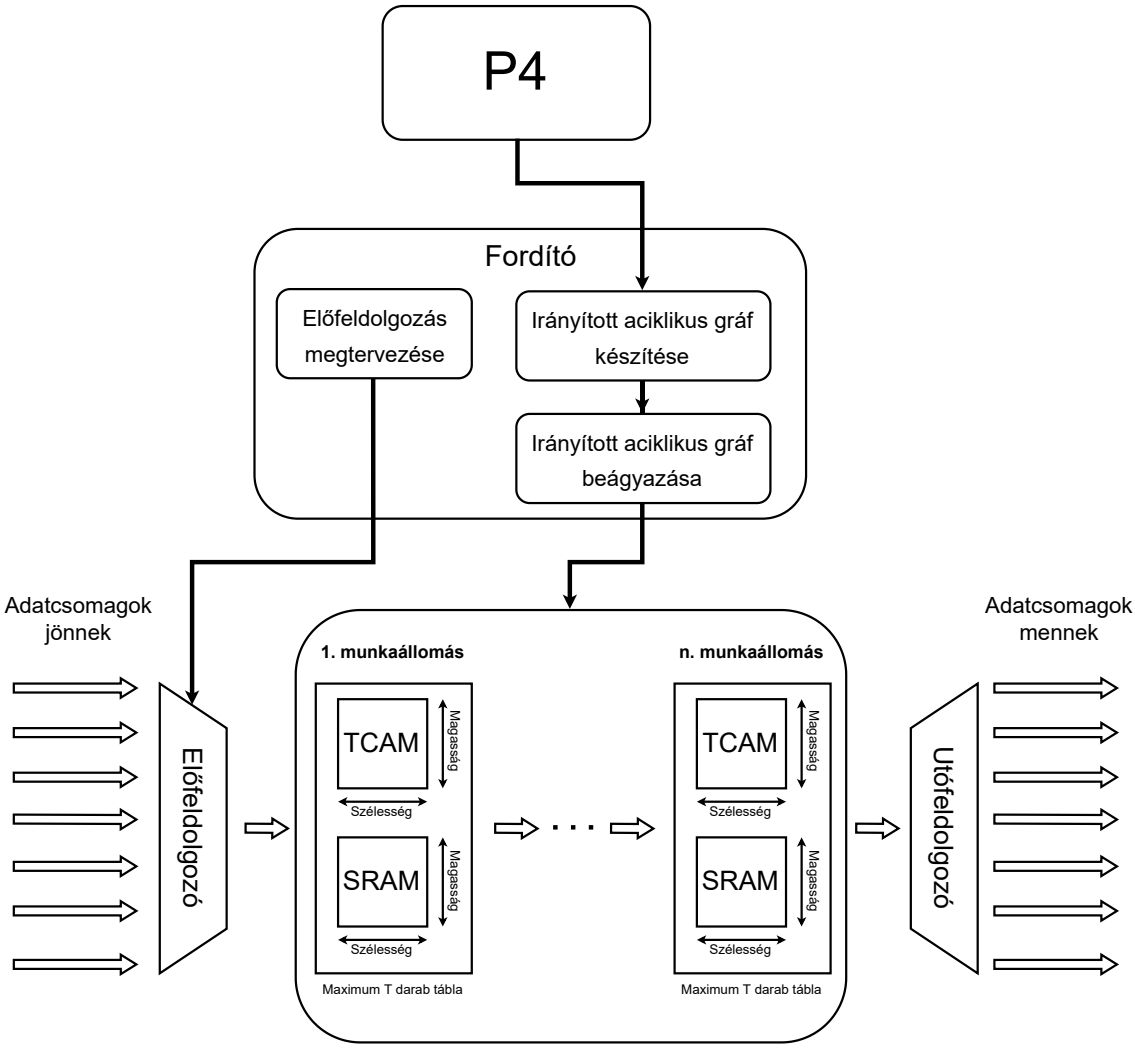
Leegyszerűsítve, a beágyazásra az alábbi szerkezeti megkötések vannak:

- **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 helyezzük el.

- **Crossbar:** Minden munkaállomásban van egy korlát a táblák számára, az egyeztetések szélességeinek összegére és a műveletek szélességeinek összegére.

- **Word-packing:** Néhány bejegyzést *egymás mellé lehet ragasztani*, ezzel memóriát spórolva.



1.2. 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 beágyaztuk őket. (Például az 1.1. ábrán Unicast Routing és Switching táblák között ilyen van.)

Ezen megkötések mellett szeretnénk minimalizálni a munkaállomások számát, az energiahasználatot és a szekvencia hosszát.

A megkötések közül nem fogjuk mindet modellezni, kizárólag a memóriára vonatkozó megkötéseket, a h-splitet, illetve a crossbarból a legelső korlátot. Továbbá egyedül a munkaállomások számára fogunk minimalizálni. A modelleket és hozzájuk tartozó bizonyításokat a [6] cikk alapján dolgozom fel.

1.2. Fontos fogalmak

Egy RMT feladatot az alábbi bemenethalmaz ír le: adott egy $G(V, E)$ irányított aciklikus gráf, egy $H : V \to \mathbb{N}^+$ függvény a bejegyzések számáról, egy $W : V \to \mathbb{N}^+$ szélességi függvény, egy $V_{TCAM} \subseteq V$ halmaz, illetve $\overline{W_{TCAM}}, \overline{W_{SRAM}}, \overline{H_{TCAM}}, \overline{H_{SRAM}}, \overline{T} \in \mathbb{N}^+$ konstans korlátok a szélességek összegére, bejegyzések számára és táblák számára.

Minden csúcsot el kell helyeznünk valamelyik munkaállomáson és valamelyik memória típusba is. Ezek alapján vezessük be a következő definíciót:

1.2.1. Definíció. *Egy tetszőleges* $A : V \to \mathbb{N}^+ \times \{SRAM, TCAM\} \times (\mathbb{N}^+ \times \mathbb{N}^+)$ *függvényt* ***beágyazásnak*** *nevezünk. (Ez azt reprezentálja, hogy a tábla elejét melyik munkaállomásába ágyazzuk be, az SRAM és TCAM a memória típusokat használja-e, továbbá a munkaállomáson belül melyik sor melyik oszlopába kezdjük el beágyazni.) Használjuk az* $A(v) = \Big(S(v), M(v), (R(v), C(v))\Big)$ *jelölést, ahol* $S(v), R(v), C(v) \in \mathbb{N}^+$, $M(v) \in \{SRAM, TCAM\}$. *Ha teljesül* S*-re az, hogy minden* $(i, j) \in E$ *élre* $S(i) < S(j)$, *akkor* A*-t* ***megengedett beágyazásnak*** *hívjuk.*

Egy megengedett beágyazásban a csúcsoknak még az alábbi szerkezeti korlátokat kell teljesíteniük:

- **Memória típusa:** A V_{TCAM} halmaz csúcsait kizárólag a TCAM típusú memóriába helyezhetjük el. Formálisan,

$$\forall v \in V_{TCAM} : M(v) = TCAM.$$

- **Szélességi és magassági korlát:** Mindkét memóriatípusnak van egy szélessége ($\overline{W}$) és egy magassága ($\overline{H}$). A táblákat (V csúcsait) úgy kell elhelyezni, hogy azok ne fedjék egymást, illetve ne lógjanak ki memóriából. (Azaz egy 2D ládapakolást kell megvalósítanunk.)

Mivel egy csúcsban a bejegyzések száma lehet több, mint $\overline{H}$, ezért bizonyos modellekben megengedjük a következőt:

- **Horizontális vágás:** Egy tetszőleges csúcsot a bejegyzései mentén szétvághatunk két másik csúcsra. Formálisan tetszőleges $v \in V$ csúcs és $1 \leq j \leq B(v)-1$ egész szám esetén $G(V,E)$ gráfból készíthetünk egy $G'(V',E')$ gráfot úgy, hogy

$$\begin{aligned} V' =& v_1 \cup v_2 \cup V \setminus v \\ E' =& \{v_1v_2\} \cup \{wv_1 | w \in V \setminus v; wv \in E\} \cup \{v_2w | w \in V \setminus v; vw \in E\} \\ & \cup \{w_iw_j | w_i, w_j \in V \setminus v; w_iw_j \in E\} \\ B(v_1) =& j; B(v_2) = B(v) - j \\ W(v_1) =& \sum_{i=1}^{j} j \cdot W(v); W(v_2) = \sum_{i=j+1}^{B(v)} (B(v) - j) \cdot W(v) \end{aligned}$$

  Mivel a gráf eredeti csúcsait folytonosan szeretnénk beágyazni, ezért a kettévágás során $S(v_2), (R(v_2), C(v_2))$ reprezentálja azt, hogy mi az utolsó munkaállomás, és azon belül sor és oszlop, ameddig tart ennek a táblának feldolgozása. (Azaz azon köztes munkaállomásokat, ahol kizárólag v bejegyzései lesznek teljes egészében beágyazva, csak gondolatban foglaljuk le.) Ezzel elkerülve azt, hogy ha egy táblát csak nagyon sok munkaállomásba tudunk elhelyezni, akkor mindegyik munkaállomást le kelljen írni a kimenetben. Így csak a csúcs elejét és végét kell meghatároznunk, onnan pedig egyértelműen kitalálható, hogy hogyan néz ki a beágyazás.

  Egy csúcsot akár többször is kettévághatunk, azonban az algoritmus gyorsítása miatt elvégezhetünk több vágást is egyszerre, így a formális definícióban a következőt is megengedjük:

$$\begin{aligned} B(v_1) =& j; B(v_2) = k \\ W(v_1) =& \sum_{i=1}^{j} j \cdot W(v); W(v_2) = \sum_{i=j+1}^{B(v)} k \cdot W(v) \end{aligned}$$

  , ahol $j + k \leq B(v)$, továbbá feltéve, hogy $S(v_1)$ és $S(v_2)$ között van elegendő munkaállomás, ahova bepakolhatjuk a kimaradt bejegyzéseket.

Azonban a következő szerkezeti korlát miatt ez még mindig nem lesz egy jó definíció, ezért csináljuk még a következőt: ha eddig nem volt egyáltalán horizontális vágás a gráfban, akkor rendezzük sorba V csúcsait: $w_1, \ldots, w_n$, majd vezessük be a $T : V \to \mathbb{R} : T(w_i) = i$ függvényt. Ha már volt horizontális vágás, akkor az ott szereplő jelöléssel $T(v_1) = T(v_2) := T(v)$. Így már bevezethetjük a következő definíciót:

- **Táblák száma:** Minden munkaállomáson legfeljebb $\overline{T}$ táblát ágyazhatunk be. Formálisan, definiáljuk a $\mathcal{V}_t := \{v \in V | S(v) = t\}$ halmazt $\forall t \in \mathbb{N}^+$-re. Ekkor annak kell igaznak lenni, hogy

$$\forall t \in \mathbb{N}^+ : \#\{Im(T(\mathcal{V}_t))\} \leq \overline{T}.$$

A vágást megengedve az RMT feladat egy olyan 2 dimenziós ládapakolási feladattá alakult, ahol szükség esetén az eredeti téglalapokat vízszintesen elvághatjuk akár többször is, akár egy ládán belül is.

1.2.2. Definíció. *Egy megengedett beágyazást* ***végrehajtható beágyazásnak*** *nevezünk, ha teljesíti a fent felsorolt szerkezeti korlátokat.*

Az RMT feladat kimenetének egy végrehajtható beágyazásnak kell lennie. Célunk, hogy minél kevesebb munkaállomást használjunk fel.

Jelölje $N(A) = \max_t(\exists v \in V : S(v) = t)$ az A beágyazásban használt munkaállomások számát. Ekkor az optimális megoldást a következőképp jelöljük:

1.2.3. Definíció.

$$OPT := \min\left(N(A)\middle| A \textit{ egy végrehajtható beágyazás}\right)$$

A modellek során feltesszük, hogy $\max_{v \in V_{TCAM}} W(v) \leq \overline{W_{TCAM}}$, $\max_{v \in V} W(v) \leq \max(\overline{W_{TCAM}}, \overline{W_{SRAM}})$, illetve ha nincs horizontális vágás, akkor $\max_{v \in V} H(v) \leq \overline{B}$. Így a definícióban szereplő minimum jóldefiniált, hiszen ekkor mindig létezik végrehajtható beágyazás.

1.3. Modellek

Ebben a szekcióban definiáljuk a vizsgált modelleket. Ha nem szerepel a modellben, hogy $\overline{W_{SRAM}}$, akkor legyen $V_{TCAM} = V$. Ilyenkor $\overline{W_{TCAM}}$ és $\overline{H_{TCAM}}$ helyett csak annyit írunk, hogy $\overline{W}$ és $\overline{H}$. Ha egyáltalán nem szerepel a modellben $\overline{W}$, akkor legyen $\overline{W} = 1$ és $W(v) = 1$ minden $v \in V$-re. Továbbá ha nem definiáljuk külön, akkor $\overline{T} = \infty$, illetve a horizontális vágást megengedjük. A modellek a következők:

1. $\overline{H} = \infty$, nincs horizontális vágás
2. $\overline{H} \in \mathbb{N}^+$, nincs horizontális vágás
3. $\overline{H} \in \mathbb{N}^+$
4. $\overline{H}, \overline{W} \in \mathbb{N}^+$
5. $\overline{H_{SRAM}}, \overline{H_{TCAM}}, \overline{W_{SRAM}}, \overline{W_{TCAM}} \in \mathbb{N}^+$
6. $\overline{H_{SRAM}}, \overline{H_{TCAM}}, \overline{W_{SRAM}}, \overline{W_{TCAM}}, \overline{T} \in \mathbb{N}^+$

1.4. Eredmények

Ebben a szekcióban összefoglalom a főbb eredményeket a modellekről.

Az 1. modell lineáris időben megoldható, hiszen a G gráfban a leghosszabb út hossza lesz a minimális munkaállomások száma. Azonban az összes többi modell mindegyike már NP-teljes[1] probléma. Sőt, a 2. modellben nem létezik 3/2-nél jobb, a 3., 4. és 5. modellben pedig 5/4-nél jobb approximációs algoritmus, kivéve ha $P = NP$. Azonban ezen modellek esetén megadható egy mohó algoritmus, ami segítségével közelíthetjük az optimális megoldásunkat. A 2. modellben megadható egy 3-közelítő algoritmus, ami ha megengedjük a horizontális vágást (3. modell), akkor máris 2-közelítő lesz. Kicsit módosítva az algoritmust a 4. modellben kapunk szintén egy 3-közelítő algoritmust. Attól függően, hogy $\overline{W_{TCAM}} \geq \overline{W_{SRAM}}$ vagy $\overline{W_{TCAM}} < \overline{W_{SRAM}}$, az 5. és 6. modellben, 5- vagy 8-közelítő, illetve 6- vagy 9-közelítő algoritmust kapunk. Ez azért van, mert $\overline{W_{TCAM}} < \overline{W_{SRAM}}$ esetén előfordulhat, hogy van olyan $v \in V \setminus V_{TCAM}$ csúcs, amire $\overline{W_{TCAM}} < W(v) \leq \overline{W_{SRAM}}$. Összefoglalva az eredményeket az 1.3. táblázat tartalmazza.

1.5. Bonyolultságok

Ebben a szekcióban megvizsgáljuk, hogy az egyes modelleknek mik a bonyolultsági osztályai.

Használjuk a későbbiekben a következő jelöléseket: $n := |V|$ és $m := |E|$.

[1]Egész pontosan akkor NP-teljes, ha a feladat eldöntési változatát tekintjük, azaz létezik-e legfeljebb N munkaállomást használó beágyazás. Ha azt a változatot tekintjük, hogy OPT értéket szeretnénk meghatározni, akkor csak annyit állíthatunk, hogy NP-nehéz, mert azt nem tudjuk, hogy ez a változat NP-ben van-e.

Modell	Komplexitás	Nem approximálható jobban, mint ... (kivéve ha P=NP)	Approximáció
1.	P	1	1
2.	NPC	3/2	3
3.	NPC	5/4	2
4.	NPC	5/4	3
5.	NPC	5/4	5 vagy 8
6.	NPC	5/4	6 vagy 9

1.3. táblázat. Főbb eredmények összefoglalva.

1.5.1. Állítás. *Az 1. modellben egy RMT feladatra $O(n+m)$ időben megadható egy optimális megoldás.*

Bizonyítás. Elegendő egy megengedett beágyazást találni, mert a munkaállomásokra semmilyen megkötésünk nincs. Vegyünk egy topologikus sorrendjét a csúcsoknak. (Ez $O(n+m)$ időt vesz igénybe.) Majd vegyük sorban a csúcsokat és ágyazzuk be a lehető legkisebb munkaállomásba. (Ez is $O(n+m)$ időt vesz igénybe összesen.) Mivel az összes őt megelőző csúcsot már beágyaztuk, ezért a legvégén ez egy megengedett beágyazás lesz. Ez optimális lesz, mivel a kritikus út (leghosszabb út a gráfban) hosszánál kevesebb munkaállomással nem megoldható a feladat.

Algoritmus 1: Csúcsok mélységének meghatározása

```
   Input: G(V, E)
   begin
1      [v_1, v_2, ..., v_n] := TopologikusRendezes(G(V, E))
2      for i = 1, ..., n do
3          L(i) := üres lista
4          if v_i befoka 0 then
               l(v_i) := 1
5          else
               l(v_i) := max{l(v_j) | (v_j, v_i) ∈ A} + 1
6          L(l(v_i)).append(v_i)
7      return L
```

□

A következő tételben fel fogjuk használni, hogy a `PARTITION` probléma NP-

teljes [3]. Azaz ha adott egy $\mathcal{H}$ pozitív egész számokból álló halmaz, akkor annak eldöntése, hogy $\mathcal{H}$ elemei partícionálhatók-e $\mathcal{H}_1$ és $\mathcal{H}_2$ halmazokra (azaz $\mathcal{H}_1 \cup \mathcal{H}_2 = \mathcal{H}, \mathcal{H}_1 \cap \mathcal{H}_2 = \emptyset$) úgy, hogy a $\mathcal{H}_1$-ben lévő számok összege megegyezzen a $\mathcal{H}_2$-ben lévő számok összegével, NP-teljes.

1.5.2. Tétel. *Annak eldöntése, hogy a 2. modellben egy RMT feladat és $k \in \mathbb{N}^+$ esetén létezik-e végrehajtható beágyazás legfeljebb k munkaállomás felhasználásával, NP-teljes probléma. Továbbá nem létezik $3/2$-nél jobb közelítő algoritmus, kivéve ha $P = NP$.*

Bizonyítás. A probléma NP-ben van, hiszen egy adott beágyazásról polinomiális időben eldönthető, hogy végrehajtható beágyazás-e.

Az `PARTITION` problémát vissza akarjuk vezetni egy 2. modellben lévő RMT feladatra. Legyen adva a $\mathcal{H} = \{h_1, \ldots, h_n\}$ halmazunk, amit $\mathcal{H}_1$ és $\mathcal{H}_2$ osztályokra szeretnénk osztani úgy, hogy a két halmazban lévő összeg ugyanannyi legyen.

Tekintsük a következő RMT feladatot: $V := \bigcup_{i=1}^{n} v_i$, $E := \emptyset$, $\overline{H} := \frac{1}{2}\sum_{i=1}^{n} h_i$, $H(v_i) = h_i$, ahol $1 \leq i \leq n$.

Pontosan akkor létezik 2 munkaállomást használó végrehajtható beágyazás erre a feladatra, ha a $\mathcal{H}$ halmazt partícionálni tudjuk $\mathcal{H}_1$ és $\mathcal{H}_2$ halmazokra. Ezzel polinomiálisan visszavezettük az `PARTITION` problémát az RMT problémára, így az NP-teljességgel készen vagyunk.

Ha létezne $3/2$-nél jobban közelítő algoritmus, akkor speciálisan erről az RMT feladatról is el tudnánk dönteni, hogy létezik-e 2 munkaállomást használó végrehajtható beágyazás. Ez utóbbi viszont csak akkor lehetséges, ha $P = NP$. □

A következő tételben azt használjuk fel, hogy az `EQUAL CARDINALITY PARTITION` (ECP) probléma NP-teljes [3]. Az ECP probléma arról szól, hogy ha adott $2k$ darab pozitív egész számból álló $\mathcal{H}$ halmaz, akkor annak eldöntése, hogy ezek két k-k elemből álló $\mathcal{H}_1, \mathcal{H}_2$ partíció osztályra oszthatók-e úgy, hogy a $\mathcal{H}_1$-ben lévő számok összege megegyezzen a $\mathcal{H}_2$-ben lévő számok összegével, NP-teljes.

1.5.3. Tétel. *Annak eldöntése, hogy egy RMT feladat és $k \in \mathbb{N}^+$ esetén a 3. modellben (így a 4.,5. és 6. modellekben is) létezik végrehajtható beágyazás legfeljebb k munkaállomás felhasználásával, NP-teljes probléma. Továbbá nem létezik $5/4$-nél jobb közelítő algoritmus, kivéve ha $P = NP$.*

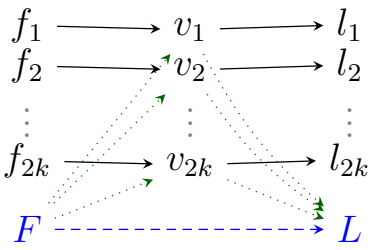
Bizonyítás. A probléma NP-ben van, hiszen egy adott beágyazásról most is polinomiális időben eldönthető, hogy végrehajtható beágyazás-e.

Az ECP problémát vissza akarjuk vezetni egy 3. modellben lévő RMT feladatra. Legyen adva a $\mathcal{H} = \{h_1, \ldots, h_{2k}\}$ $2k$ elemű halmazunk, amit fel szeretnénk osztani $\mathcal{H}_1 = \{h_{i_1}, \ldots, h_{i_k}\}$ és $\mathcal{H}_2 = \{h_{i_{k+1}}, \ldots, h_{i_{2k}}\}$ diszjunkt k elemű osztályokra úgy, hogy $\sum_{j=1}^{k} h_{i_j} = \sum_{j=k+1}^{2k} h_{i_j}$.

Legyen a $K := \frac{1}{2}\sum_{i=1}^{2k} h_i$. (Feltehető, hogy $K \in \mathbb{N}^+$.) Legyen a gráfunk a következő (lásd: 1.4. ábra):

$$V := \{F, L\} \cup \bigcup_{i=1}^{2k} \{f_i, v_i, l_i\}$$

$$E := \{(F, L)\} \cup \bigcup_{i=1}^{2k} \{(f_i, v_i), (F, v_i), (v_i, l_i), (h_i, L)\}$$

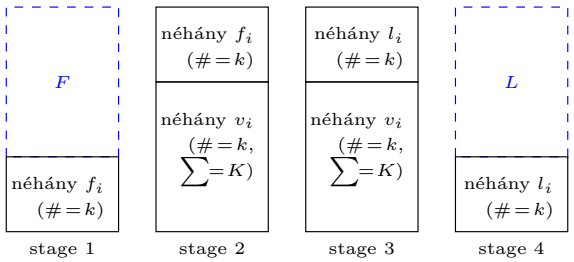


1.4. ábra.

Legyen $H(f_i) = H(l_i) = 1$ és $H(v_i) = h_i$ minden $1 \leq i \leq 2k$-ra, továbbá $H(F) = H(L) = K + k$.

Ha a $\mathcal{H}$ halmaz ugyanakkora méretű és összegű $\mathcal{H}_1$ és $\mathcal{H}_2$ halmazokra partícionálható, akkor ez az RMT feladat 4 munkaállomásba végrehajthatóan beágyazásható: F-et ágyazzuk be az 1., L-et a 4. munkaállomásba; $\mathcal{H}_1$ halmazhoz tartozó v_i elemeket ágyazzuk be a 2. munkaállomásba, míg $\mathcal{H}_2$ halmazhoz tartozó v_i elemeket a 3. munkaállomásba; Ha $h_i \in \mathcal{H}_1$, akkor f_i-t ágyazzuk be az 1., l_i-t a 3. munkaállomásba, ha $h_i \in \mathcal{H}_2$, akkor a 2. és 4. munkaállomásba. Így ez egy végrehajtható beágyazás.

Ugyanez fordítva is igaz, tegyük fel, hogy adott egy végrehajtható beágyazás, mely négy munkaállomást használt. $v_1, \ldots, v_{2k}$ elemeket muszáj volt a 2. és 3. munkaállomásba beágyazni, mert befokuk és kifokuk is pozitív. Ebből következik, hogy f_i csak az 1. és 2., l_i csak a 3. és 4. munkaállomásba kerülhet, továbbá az F teljes egészében csak az 1. munkaállomáson lehet (különben az összes v_i-nek a 3. munkaállomáson kellene lennie), illetve L teljes egészében a 4. munkaállomáson.



1.5. ábra.

Így az 1. és 4. munkaállomáson is k-k darab bejegyzésnek való hely maradt, ahova csak az f_i és l_i elemek mehetnek. Így 2. és 3. munkaállomáson is k-k darab v_i-nek kell lennie, továbbá a $\overline{H} = K + k$ korlát miatt K-K-nak kell lennie $H(v_i) = h_i$-k összegének a két halmazban. Ez viszont megadja nekünk az ugyanakkora méretű és összegű $\mathcal{H}_1$ és $\mathcal{H}_2$ halmazokat.

Azaz $\mathcal{H}$ halmaz pontosan akkor partícionálható megfelelő $\mathcal{H}_1$ és $\mathcal{H}_2$ halmazokra, ha ennek az RMT feladatnak van megoldása. Ezzel polinomiálisan visszavezettük az `ECP` problémát az RMT problémára, így az NP-teljességel készen vagyunk.

Ha létezne 5/4-nél jobban közelítő algoritmus, akkor speciálisan erről az RMT feladatról is el tudnánk dönteni, hogy létezik 4 munkaállomást használó végrehajtható beágyazás. Ez utóbbi viszont csak akkor lehetséges, ha $P = NP$.

Mivel a 4.,5. és 6. modelleknek egy speciális esete a 3. modellnek, ezért ezek is NP-teljes problémák és ők sem közelítők jobban, mint 5/4, kivéve ha $P = NP$. □

1.6. Approximációs algoritmusok

Mindegyik modellre megadunk egy közelítő algoritmus, melyek (kvázi) lineáris időben találnak egy végrehajtható beágyazást egy RMT feladatra.

Mindegyik algoritmusnak ugyanaz lesz az alapja: szintenként beágyazzuk a csúcsokat. Ez $O(n+m)$ időt vesz igénybe. Az, hogy egy szinten belül hogyan ágyazzuk be őket, az már modellspecifikus.

1.6.1. Állítás. *A 2. modellben egy RMT feladatra $O(n\log(n)+m)$ időben megadható olyan végrehajtható beágyazás, amely legfeljebb $3 \cdot OPT$ munkaállomást használ.*

Bizonyítás. Ebben a modellben a specifikus algoritmus egy egyszerű mohó algoritmus lesz. Rendezzük méret szerint csökkenő sorrendbe a csúcsokat, majd egyesével vegyük a csúcsokat, és ha a bejegyzések száma több lenne, mint a $\overline{H}$ korlát, akkor kezdjünk új munkaállomást. Így minden szinten az utolsó munkaállomás kivételével

Algoritmus 2: Szintenkénti beágyazás

Input: $G(V,E)$; H; $\overline{H_{SRAM}}$,$\overline{H_{TCAM}}$;W;$\overline{W_{SRAM}}$;$\overline{W_{TCAM}}$;$\overline{V}$
begin
1 $\mathcal{L}$:=`CsucsokMelysege(`G`)`
2 $D(0) := 0$, $i := 1$
3 **while** $\mathcal{L}(i) \neq \emptyset$ **do**
4 $D(i) := D(i-1) + \lceil (\sum_{v \in \mathcal{L}(i)} H(v))/\overline{H} \rceil$
5 $(A[D(i-1)+1], \ldots, A[D(i)])$:= `ModellSpecifikusAlgoritmus(`
$\mathcal{L}(i), H, \overline{H_{SRAM}}, \overline{H_{TCAM}}, W, \overline{W_{SRAM}}, \overline{W_{TCAM}}, \overline{V}$)
6 $i := i + 1$
7 **return** *A beágyazás*

Algoritmus 3: 2. modell: mohó algoritmus

Input: $[v_1, \ldots, v_k]$; H; $\overline{H}$
begin
1 A :=üres lista
2 $j := 1$
3 $s := 0$
4 `Sort(`$[v_1, \ldots, v_k]$`)` `//` $H(v_1) \geq \ldots \geq H(v_k)$
5 **for** $i = 1, \ldots, k$ **do**
6 **if** $s + H(v) > \overline{H}$ **then**
7 $j := j + 1$
8 $s := 0$
9 $A[j]$.append$((v_i, \text{TCAM}, (s, 0)))$
10 $s := s + H(v)$
11 **return** *A beágyazás*

minden munkaállomásban legalább $\overline{H}/2$ bejegyzés van. Így ezen munkaállomások száma $\leq 2 \cdot OPT$, az utolsó munkaállomások száma, ami mivel megegyezik a kritikus út hosszával, $\leq OPT$. Innen megkapjuk, hogy 3-közelítő algoritmus ez. A rendezés összességében $O(n \log(n))$ időt vesz igénybe, így az algoritmus futási ideje összességében (a szintek meghatározásával együtt) $O(n \log(n) + m)$. □

1.6.2. Állítás. *A 3. modellben egy RMT feladatra $O(n+m)$ időben megadható egy olyan végrehajtható beágyazás, amely legfeljebb $2 \cdot OPT$ munkaállomást használ.*

Bizonyítás. A különbség az előző algoritmus képest az, hogy most nem kell sorba rendeznünk a csúcsokat, hiszen szükség esetén horizontális vágással egy csúcsból két csúcsot készíthetünk. Így minden szinten az utolsó munkaállomás kivételével

Algoritmus 4: 4. modell: mohó algoritmus

Input: $[v_1, \ldots, v_k]$; H; $\overline{H}$
begin
1 A :=üres lista
2 $j := 1$
3 $s := 0$
4 **for** $i = 1, \ldots, k$ **do**
5 **if** $s + H(v) > \overline{H}$ **then**
6 (v_i, v_i', j', s') :=`HorizontalisVagas`$(v_i, s + H(v) - \overline{H})$
7 $A[j]$.append$((v_i, \text{TCAM}, (s, 0)))$
`// Első sort adjuk meg, ahonnan indul a beágyazás`
8 $j := j + j'$
`// Közben` $j' - 1$ `munkaállomást teljes egészében felhasználtunk`
9 $s := s'$
`// Utolsó munkaállomásban` s' `sort kell még használnunk`
10 $A[j]$.append$((v_i', \text{TCAM}, (s' - 1, 0)))$
`// Utolsó sort adjuk meg, ahol végződik a beágyazás`
11 **else**
12 $s := s + H(v)$
13 $A[j]$.append$((v_i, \text{TCAM}, (s, 0)))$
14 **return** *A beágyazás*

minden munkaállomásban $\overline{H}$ bejegyzés szerepel, így ezen munkaállomások száma most csak $\leq OPT$. Minden szinten az utolsó munkaállomások száma (a kritikus út hossza miatt) továbbra is $\leq OPT$, így az algoritmusunk 2-közelítő. Mivel nem kell a csúcsokat sorba rendezni, és egy csúcsra legfeljebb egyszer alkalmazzuk a horizontális vágást, ezért $O(n + m)$ időben végez. □

1.6.3. Állítás. *Egy RMT feladatra a 4. modellben megadható olyan végrehajtható beágyazás $O(n \log(n) + m)$ időben, amely legfeljebb $3 \cdot OPT$ munkaállomást használ.*

Bizonyítás. Rendezzük a csúcsokat most szélesség szerinti csökkenő sorrendbe. Az általánosság elvesztése nélkül feltehetjük, hogy $\overline{W} = 1$. Először tekintsünk úgy, mintha egy memória végtelen magas lenne. Először vegyük azon csúcsokat, melyek szélessége $\geq 1/2$, és ágyazzuk be őket egy oszlopba. Ezután a k-ik lépésben ágyazzuk be azon csúcsokat k oszlopba, melyek szélessége $[1/(k+1), 1/k)$ halmazba tartozik. A csúcsokra hajtsunk végre horizontális vágásokat és így ágyazzuk be sorba őket. Ha az utolsó sorba még maradt hely, akkor a következő csúcsot véve horizontális elvágva ágyazzuk be őt a maradék helyre. Ekkor az utolsó munkaállomást kivéve

mindegyik munkaállomáson minden sorban a szélességek legalább felét kihasználtuk. Ez $\leq 2 \cdot OPT$. Az utolsó munkaállomások száma $\leq OPT$, így ez egy 3-közelítő algoritmus, továbbá $O(n \log(n) + m)$ végez. □

1.6.4. Állítás. *Egy RMT feladatra az 5. modellben megadható olyan végrehajtható beágyazás $O(n \log(n) + m)$, amely legfeljebb $8 \cdot OPT$ munkaállomást használ. Továbbá, ha $\overline{W_{TCAM}} \geq \overline{W_{SRAM}}$, akkor legfeljebb $5 \cdot OPT$ munkaállomást használ.*

Bizonyítás. Tegyük fel, hogy most az i-ik szinten vagyunk. Mindig szélesség szerint csökkenő sorrendben fogunk beágyazni, illetve szükség esetén horizontálisan elvágjuk a csúcsokat. Először ágyazzuk be V_{TCAM} csúcsait 1-től a t_i-ig terjedő munkaállomások TCAM memóriájába. Ezután azon v csúcsokat ágyazzuk be 1-től s_i-ig terjedő munkaállomásokba az SRAM memóriába, melyekre $W(v) > \overline{W_{TCAM}}$. (Ha nincs ilyen csúcs, akkor $s_i := 1$.) Ezután a maradék csúcsokat ágyazzuk be a $\min\{t_i, s_i\}$ munkaállomástól elkezdve folyamatosan a TCAM és SRAM memóriákba, beleértve a megmaradt helyeket (az üres sorokat) a t_i-ik és s_i-ik munkaállomásokban. Tegyük fel, hogy az utolsó munkaállomás sorszáma $\max(s_i, t_i) + l_i$.

Ekkor meggondolható, hogy $\sum_{\text{szintek}}(t_i - 1) \leq 2 \cdot OPT$, hiszen legalább félig tele vannak ezek a TCAM memóriák. Ugyanígy $\sum_{\text{szintek}}(s_i - 1) \leq 2 \cdot OPT$. Szokásos módon azt is tudjuk, hogy $\sum_{szintek}(l_i - 1) \leq 2 \cdot OPT$. Így eddig $\leq 6 \cdot OPT$ becslésnél tartunk. Minden munkaállomás szélessége félig tele van, kivéve az utolsó munkaállomást, a t_i-ik munkaállomást (mert abban a sorban, ahol abbahagytuk a TCAM-be való bepakolást, lehet egy legalább félig üres sor) és az s_i-ik munkaállomás (szintén abban a sorban, ahol abbahagytuk az SRAM-ba való bepakolást, előfordulhat egy legalább félig üres sor). Azonban ezek közül az egyik eltűntethető: ha van az utolsó munkaállomáson pl. TCAM-ben olyan sor, ami legalább félig tele van, akkor azt kicserélhetjük a t_i-ik munkaállomáson lévő félig legalább üres sorral. Ezzel csak az s_i és utolsó munkaállomás maradt. Ha nincs sem TCAM-ben sem SRAM-ban ilyen sor, akkor csak olyan sor lehet, ami legalább félig üres. Ha pl. TCAM-ben van a legalább félig üres sor, akkor ezt a sort a t_i-ik munkaállomás szintén legalább félig üres sora mellé rakva az utolsó munkaállomás fel is szabadul, így azt el is hagyhatjuk.

Azaz legfeljebb két munkaállomás van, ahol nincs minden sor legalább félig tele, így ezek száma $\leq 2 \cdot OPT$. Összességében pedig azt kaptuk, hogy $\leq (6 + 2) \cdot OPT$ munkaállomást használtunk.

Ha $\overline{W_{TCAM}} \geq \overline{W_{SRAM}}$, akkor $s_i = 1$, továbbá legfeljebb a TCAM memóriában maradt félig üres sor, amit vagy ki tudunk cserélni, vagy az utolsó munkaállomást el tudjuk tüntetni. Így kapjuk azt, hogy $\leq (2+2+1) \cdot OPT$ munkállomást használtunk maximum ez esetben. □

1.6.5. Állítás. *Egy RMT feladatra a 6. modellben megadható olyan végrehajtható beágyazás $O(n\log(n)+m)$, amely legfeljebb $9\cdot OPT$ munkaállomást használ. Továbbá, ha $\overline{W_{TCAM}} \geq \overline{W_{SRAM}}$, akkor legfeljebb $6\cdot OPT$ munkaállomást használ.*

Bizonyítás. Ugyanazt csináljuk, mint az előző modellben, annyi változtatással, hogy ha elértük a $\overline{T}$ korlátot, akkor új munkaállomást kezdünk a következő csúcsnál. Ezen munkaállomások száma, ahol $\overline{T}$ csúcsot használtunk és nincs félig tele szélesség szerint az összes sor, $\leq OPT$ darab lehet. (Meggondolható, hogy egy csúcsot legfeljebb egy ilyen munkaállomáson számolhatunk meg.) Így végeredményben $8+1$- és $5+1$-közelítő algoritmusokat kapunk. □

2. fejezet

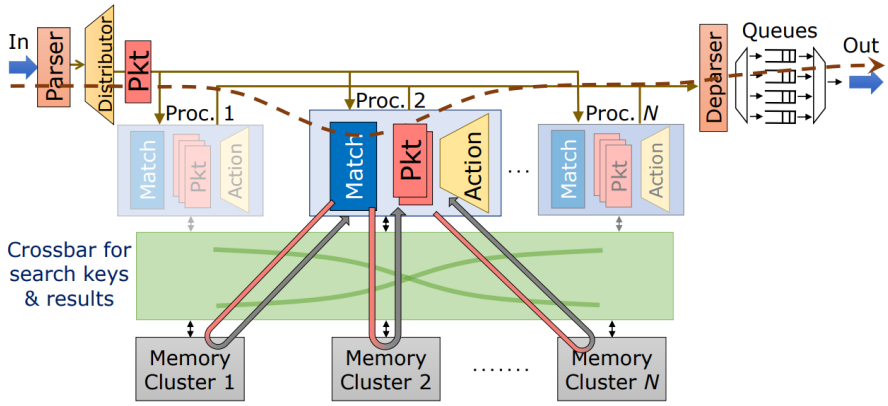
dRMT architektúra

2.1. Technikai háttér

A dRMT (disaggregated Reconfigurable Match-Action Table) [2] 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 végig-



2.1. ábra. A dRMT architektúra felépítése. [2]

menjen 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 megfelezheti (vagy ha k-szor megy végig a szekvencián, akkor k-ad részére csökkentheti) a teljesítményt a maximális teljesítményhez képest.

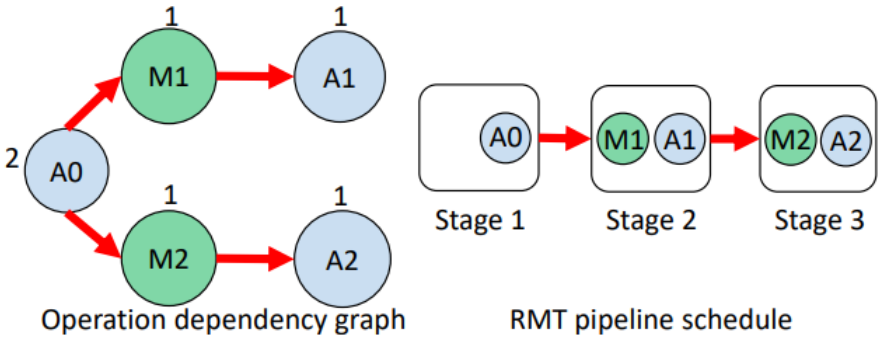
A dRMT úgy oldja meg mindkét problémát, hogy különválasztja a hardveres erőforrásokat:

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 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 (lásd: 2.2. és 2.3. ábra)

Feltehetjük, hogy a P4 program kimenetéből a fordító segítségével rögtön megkapjuk az aciklikus gráfunkat, aminek a csúcsai lesznek a match és action csúcsok,



2.2. ábra. Egy példa program és annak beágyazása az RMT architektúrában. [2]

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	$A_1\&A_2$

2.3. táblázat. A példa program egy lehetséges ütemezése a dRMT architektúrában.

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 egy match vagy action csúcs után egy másik valamilyen csúcsba él megy, akkor a legkorábban csak a ΔM-ik illetve ΔA-ik órajel után hajtódhat 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 [2] 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 egyrészről ennek a P periódusnak, másrészről az ütemezés hosszának minimalizálása volt. Mi ezen annyit módosítunk, hogy meg fogjuk engedni azt is, hogy egyszerre akár k ütemezést is ismételhessünk.

2.2. Fontos fogalmak

Az alábbi szekcióban definiáljuk a fontos 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 $W : V \to \mathbb{N}^+$ szélességi függvény, $\Delta M, \Delta A \in \mathbb{N}^+$ késleltetési konstansok, $\overline{M}, \overline{A}, IPC \in \mathbb{N}^+ \cup \{\infty\}$ szerkezeti korlátok.

A ΔM és ΔA konstansok indukálnak egy $l : V \to \mathbb{N}^+$ **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.

2.2.1. Definíció. *Egy tetszőleges $S : V \to \mathbb{N}^+$ függvényt **ütemezésnek** nevezünk. (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.*

Meg akarjuk tudni, hogy hogyan lehet processzorunkat a legjobban kihasználni, ekkor viszont nem feltétlen egy darab ütemezést fogunk periodikusan ismételni, hanem lehet, hogy többet. Sőt, ha elengedjük a periodicitást, akkor azt is megengedhetjük, hogy végtelen sok ütemezésünk legyen. Ennek fényében bevezetjük az alábbi definíciót:

2.2.2. 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.*

Egy megengedett k-ütemezésben csúcsoknak még az alábbi két szerkezeti korlátot kell teljesíteniük:

- **Erőforrás korlát**, (i) és (ii): Minden match és action csúcsnak van egy **szélessége**, amit a W függvény tartalmaz. 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

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

  és ugyanez az action csúcsokra.

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

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

2.2.5. Definíció. *Legyen* $k < \infty$*, ekkor* $T = T(\mathcal{S}) := \max_{S\in\mathcal{S},v\in V}(S(v) + l(v))$ *egy* $\mathcal{S}$ k*-ütemezés* ***hossza****.*

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

2.2.7. Definíció. *Legyen* $k, P \in \mathbb{N}^+$ *és* $\mathcal{S} = \{S_1, \ldots, S_k\}$ *egy megengedett* k*-ütemezés. Legyen* $\mathcal{S}' := (S_{(n,m)}|n \leq k, m \in \mathbb{N})$*, 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* ∞*-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 gyakorlatban könnyebben ellenőrizhető a következő feltétel:

2.2.8. Állítás. *Legyen* $k, P \in \mathbb{N}^+$ *és* $\mathcal{S} = \{S_1, \ldots, S_k\}$ *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 erőforráskorlát és IPC korlát.*

Bizonyítás. Meggondolható, hogy az $\mathcal{S}'$ ∞-ütemezésben minden $t > T(\mathcal{S})$ órajelben pontosan azon csúcsok fognak szerepelni, mint az $\mathcal{S}$ $t \pmod P$ szerinti ekvivalencia osztályában. $\mathcal{S}'$ minden $t \leq T(\mathcal{S})$ órajelében pedig ugyanezen csúcsoknak csak egy részhalmaza. □

Elsődleges célunk az, hogy olyan végrehajtható ütemezést találjunk, aminek minél nagyobb a hatékonysága. Másodlagos célunk az, hogy minimalizáljuk T-t. Ezek alapján az alábbi módon definiáljuk az optimális értékeket:

2.2.9. Definíció.

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

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

$$OPT_k^T := \min(T(\mathcal{S})|\mathcal{S} \text{ egy végrehajtható } P\text{-periodikus } k\text{-ütemezés})$$

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 $\frac{k}{P} \geq \frac{1}{|V_m|+|V_a|}$. Hasonlóan az OPT_k^T definíciója is értelmes.

2.2.10. Megjegyzés. *Az OPT_k és OPT_k^T nem feltétlen ugyanarra az ütemezésre veszik fel az optimális értéküket. Az $IPC = 1$ modellek esetében látni fogjuk (lásd: 2.6.12. tétel.)*

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

2.3. Modellek

Definiáljuk a vizsgált modelleket, melyek egy processzort és egyféle memóriát használnak. Ha nem definiáljuk külön, akkor $IPC = \infty$, $W \equiv 1$, $\Delta M = 1$, $\Delta A = 1$, illetve ha nem szerepel $\overline{A}$ a modellben, akkor $V_a = \emptyset$.

1. $\overline{M} = \infty$
2. $\overline{M} \in \mathbb{N}^+$
3. $\overline{M}, \overline{A} \in \mathbb{N}^+$
4. $(\overline{M} \in \mathbb{N}^+) + (\Delta M \in \mathbb{N}^+)$
5. $(\overline{M}, \overline{A} \in \mathbb{N}^+) + (\Delta M, \Delta A \in \mathbb{N}^+)$
6. $(\overline{M} \in \mathbb{N}^+) + (W : V \to \mathbb{N}^+)$
7. $(\overline{M}, \overline{A} \in \mathbb{N}^+) + (\Delta M, \Delta A \in \mathbb{N}^+) + (W : V \to \mathbb{N}^+)$
8. $(IPC = 1) + (\overline{M} \in \mathbb{N}^+)$
9. $(IPC = 1) + (\overline{M} \in \mathbb{N}^+) + (\Delta M \in \mathbb{N}^+)$
10. $(IPC = 1) + (\overline{M}, \overline{A} \in \mathbb{N}^+)$
11. $(IPC = 1) + (\overline{M}, \overline{A} \in \mathbb{N}^+) + (\Delta M, \Delta A \in \mathbb{N}^+)$
12. $(IPC = 1) + (\overline{M} \in \mathbb{N}^+) + (\Delta M \in \mathbb{N}^+) + (W : V \to \mathbb{N}^+)$
13. $(IPC = 1) + (\overline{M}, \overline{A} \in \mathbb{N}^+) + (\Delta M, \Delta A \in \mathbb{N}^+) + (W : V \to \mathbb{N}^+)$
14. $(IPC = k) + (\overline{M}, \overline{A} \in \mathbb{N}^+) + (\Delta M, \Delta A \in \mathbb{N}^+) + (W : V \to \mathbb{N}^+)$

2.4. Eredmények

Mit tudunk az egyes modellekről?

Az 1. modellben elegendő egy megengedett ütemezést találni (azaz minden csúcsra meg kell határozni a mélységét), utána pedig már $P = 0$ is megfelelő.

A 2-5. modellekben (azaz $IPC = \infty$ és $W \equiv 1$) található OPT_1 periódusú ütemezés, azonban OPT_1^T meghatározása már itt is NP-nehéz probléma. Azonban amint bejön a modellünkbe az $IPC = 1$ vagy $W : V \to \mathbb{N}^+$ feltétel, az OPT_1 meghatározása máris NP-nehézzé válik.

Belátjuk, hogy a 6. modell az OPT_1-re nézve, illetve a 12. modell ekvivalens az RMT architektúra 4. modelljével. A 8. modell ekvivalens lesz egy ismert ütemezési problémával.

Megadunk a 11. és 13. modellekre egy 4- és 6-közelítő algoritmust az OPT_1-re nézve.

Továbbá a 13. modellben megmutatjuk, hogy minden dRMT feladatra létezik $K \in \mathbb{N}^+$, hogy $OPT_K = OPT_\infty$, azonban azt is megmutatjuk, hogy ez a K tetszőlegesen nagy is lehet.

Összefoglalva az eredményeket a 2.4. táblázat tartalmazza.

Modell	$\overline{M} < \infty$	$\Delta M(, \Delta A) \in \mathbb{N}$	$V_a \neq \emptyset$ $\overline{A} < \infty$	IPC	$W \not\equiv 1$	P approx. $OPT_1 \cdot \ldots$	P nem approx. jobban, mint $OPT_1 \cdot \ldots$ (Kivéve ha P = NP)
1				∞		1	1
2	✓			∞		1	1
3	✓		✓	∞		1	1
4	✓	✓		∞		1	1
5	✓	✓	✓	∞		1	1
6	✓			∞	✓	3	$3/2$
7	✓	✓	✓	∞	✓	?	$3/2$
8	✓			1		$2 - 1/\overline{M}$	$4/3$
9	✓	✓		1		$2 - 1/\overline{M}$	$4/3$
10	✓		✓	1		4	$4/3$
11	✓	✓	✓	1		4	$4/3$
12 RMT 2	✓	✓		1	✓	3	$3/2$
13	✓	✓	✓	1	✓	6	$3/2$
14	✓	✓	✓	k	✓	?	?

2.4. táblázat.

2.5. Bonyolultságok

Ebben a szekcióban megvizsgáljuk, hogy az egyes modelleknek mik a bonyolultsági osztályai.

2.5.1. Tétel. *Az 5. modellben* $OPT_k = \max\left(\left\lceil \frac{k|V_m|}{\overline{M}} \right\rceil, \left\lceil \frac{k|V_a|}{\overline{A}} \right\rceil\right) \Big/ k$, *ha* $k \in \mathbb{N}^+$, *illetve* $OPT_\infty = \max\left(\frac{|V_m|}{\overline{M}}, \frac{|V_a|}{\overline{A}}\right)$.

Bizonyítás. Legyen $P := \max\left(\left\lceil \frac{k|V_m|}{\overline{M}} \right\rceil, \left\lceil \frac{k|V_a|}{\overline{A}} \right\rceil\right)$. Ágyazzuk be egyesével a gráf k példányát a következőképp: A soron lévő példánynak vegyük azon match és action csúcsait, melyeket még nem ágyaztunk be, viszont minden őt megelőző csúcsot már beágyaztunk. Vegyük a legkisebb órajelet, ahova be lehet ágyazni valamelyik csúcsot, majd ágyazzunk be ebbe az órajelbe minél több csúcsot úgy, hogy az teljesítse a szerkezeti korlátokat P periódus mellett is. Ekkor $\mathcal{S} := \{S_n | n \leq k\}$ egy végrehajtható P-periodikus k-ütemezés, amiből $OPT_k \leq \frac{P}{k}$. Ebből megkapjuk azt is, ha $k = \overline{M} \cdot \overline{A}$, akkor $OPT_\infty \leq OPT_k = \max\left(\frac{|V_m|}{\overline{M}}, \frac{|V_a|}{\overline{A}}\right)$

Másrészről t órajel alatt a match csúcsok szélességeinek összege maximum $t\overline{M}$ lehet, az action csúcsoknál pedig $t\overline{A}$. Ebből pedig azt kapjuk, hogy

$$\max\left(\left\lceil \frac{t|V_m|}{\overline{M}} \right\rceil, \left\lceil \frac{t|V_a|}{\overline{A}} \right\rceil\right) \geq CT(\mathcal{S}, t),$$

amiből $OPT_k \geq \frac{P}{k}$ és $OPT_\infty \geq \max\left(\frac{|V_m|}{\overline{M}}, \frac{|V_a|}{\overline{A}}\right)$. □

Ezzel beláttuk, hogy az 5. modellben OPT_1 meghatározása (azaz egy minimális periódusszámú ütemezés meghatározása) P-ben van. Azonban OPT_1^T meghatározása (azaz a legrövidebb ütemezés) már NP-nehéz probléma:

2.5.2. Tétel. *Annak eldöntése, hogy* $P, T \in \mathbb{N}^+$ *esetén bármelyik (de nem az 1.) modellben létezik-e végrehajtható* S *ütemezés, hogy* $T(S) \leq T$, *NP-teljes. Továbbá nem létezik* 4/3*-nél jobban közelítő algoritmus az* OPT_1^T*-ra nézve, kivéve ha* $P = NP$.

Bizonyítás. Speciális esetként vehetjük a 2. modellt, ugyanis most az IPC=1 feltétel nem zavar be, hiszen csak egy S ütemezést keresünk.

Ennek a bizonyítása pedig megegyezik az [5] 3.4.2. Tétel bizonyításával, ahol a gépek száma most $\overline{M}$-al egyenlő.

□

2.5.3. Tétel. *Annak eldöntése, hogy rögzített* $P \in \mathbb{N}^+$ *esetén a 6. modellben (így a 7. modellben is) egy dRMT feladat esetén létezik-e végrehajtható* P*-periodikus* 1*-ütemezés, NP-teljes probléma. Továbbá nem létezik* 3/2*-nél jobb közelítő algoritmus, kivéve ha* $P = NP$.

Bizonyítás. A feladat ekvivalens az RMT problémakör 2. modelljével, ugyanis meggondolható, hogy a ΔM feltétel nem fog bezavarni. □

2.5.4. Tétel. *Annak eldöntése, hogy rögzített $k, P \in \mathbb{N}^+$ esetén a 8. modellben egy dRMT feladat esetén létezik-e végrehajtható P-periodikus k-ütemezés, NP-teljes probléma. Annak eldöntése is NP-teljes, hogy létezik-e $\mathcal{S}$ végrehajtható ∞-ütemezés, amelyre $Q(\mathcal{S}) \leq \frac{k}{P}$. Továbbá $\ell \in \mathbb{N}^+ \cup \{\infty\}$ esetén nem létezik $4/3$-nél jobban közelítő algoritmus az OPT_ℓ-re nézve, kivéve ha $P = NP$.*

Bizonyítás. Az IPC=1 feltétel miatt, és hogy csak match csúcs van, minden órajelben legfeljebb egy ütemezés fordulhat elő. Ebből következik, hogy olyan ütemezést szeretnénk találni, amely legfeljebb $\frac{P}{k}$ órajelet felhasználva végez. (Egy ütemezés akkor használ fel egy órajelet, ha van abban az órajelben legalább egy csúcsa.) Ha nem találunk ilyet, akkor nincs ilyen végrehajtható (P periodikus) k-ütemezés. Ha találunk egy ilyet, abból már könnyen készíthetünk P-periodikus k-ütemezést (például a 2.6.5. lemma ezt mondja ki), ami meghatároz egyben egy ∞-ütemezést is. Ebből a meggondolásból feltehetjük, hogy $k = 1$.

Ezután a bizonyítás megint megegyezik az [5] 3.4.2. tétel bizonyításával, ahol a gépek száma most $\overline{M}$-al egyenlő. □

A 12. modell egy speciális esete a 8. modell, így már erről is tudjuk, hogy NP-teljes tetszőleges k-ütemezést vizsgálva. Azonban a nem megközelíthetőségben tudunk javítani:

2.5.5. Tétel. *$\ell \in \mathbb{N}^+ \cup \{\infty\}$ esetén a 12. modell esetén nem létezik $3/2$-nél jobban közelítő algoritmus az OPT_ℓ-re nézve, kivéve ha $P = NP$.*

Bizonyítás. Ugyanúgy, mint a 2.5.4. tétel bizonyításában, most elegendő egy ütemezés találása, ami minél kevesebb órajelet használ fel. Ez pedig ekvivalens az RMT problémakör 2. modelljével, ugyanis meggondolható, hogy a ΔM feltétel most sem fog bezavarni. □

2.6. Periodicitás

Ebben a szekcióban azt vizsgáljuk, hogy mennyivel kapunk rosszabb megoldást az elméleti minimumhoz képest azzal, ha periodikus ütemezéseket keresünk. Azaz valami összefüggést szeretnénk találni az OPT_k és OPT_∞ között. Az egész szekcióban végig a 13. modellben fogunk dolgozni.

2.6.1. 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 action vagy match action. Formálisan*

$$A = A(\mathcal{S}) := \#\Big\{n \;\Big|\; \exists v_a \in V_a : n = S(v_a)\Big\}$$
$$M = M(\mathcal{S}) := \#\Big\{n \;\Big|\; \exists v_m \in V_m : n = S(v_m)\Big\}$$
$$N = N(\mathcal{S}) := \#\Big\{n \;\Big|\; \exists v \in V : n = S(v)\Big\}$$

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

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

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

$$C(A, M) := \Big\{S' \text{ végrehajtható ütemezés} \;\Big|\; A' = A, M' = M\Big\}$$

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

Minden végrehajtható 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.

2.6.4. Definíció. *Egy S ütemezést nevezzünk **előütemezésnek**, ha létezik olyan $f : \{1, 2, \ldots, N(S)\} \to \mathbb{N}^+$ szigorúan monoton növő függvény és S' végrehajtható ütemezés, hogy $f(S(v)) = S'(v)$ minden $v \in V$-re.*
Azt mondjuk, hogy $\mathcal{S}$ végrehajtható k-ütemezés használja az S előütemezést, ha létezik olyan $S' \in \mathcal{S}$ ütemezés, aminek az előütemezése S.

Vegyük észre, hogy véges sok minimális ütemezési osztály, illetve előütemezés van.

2.6.5. Lemma. *Legyenek $S_1, \ldots, S_k$ 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 $S_1, \ldots, S_k$ előütemezéseket használja, ráadásul mindegyiket pontosan egyszer.

Bizonyítás. Építsük fel az $\mathcal{S}$ k-ütemezést a következő algoritmus szerint i = 1-től k-ig, illetve azon belül j = 1-től N_i-ig: Ütemezzük az S_i előütemezés j-ik órajelét azon minimális órajelbe, ami még nem sérti az $IPC = 1$ korlátot az újonnan készülő $\mathcal{S}'$ ütemezésben. Ilyen órajel a P definíciója miatt mindig létezik, így tudjuk venni ezeknek a minimumát.

Az $IPC = 1$ korlát miatt a $P' < P$ nem megoldható, különben létezne olyan órajel, amikor egynél több ütemezés tartalmazna match vagy action csúcsot. □

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 2.6.5. 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:

2.6.6. 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*

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

illetve

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

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 hatékonyság 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

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

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

Vegyünk S_1-nek $\frac{A_2 - M_2}{\text{lnko}(M_1 - A_1, A_2 - M_2)}$ darab, míg S_2-nek $\frac{M_1 - A_1}{\text{lnko}(M_1 - A_1, A_2 - M_2)}$ darab példányát. A 2.6.5. 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

$$\frac{P}{k} < \frac{\max(c_1A_2 + c_2A_1, c_2M_1 + c_1M_2)}{c_1 + c_2} = \frac{c_2M_1 + c_1M_2}{c_1 + c_2}.$$

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

$$(A_2 - M_2)(M_1 - M_2)c_1 < (M_1 - A_1)(M_1 - M_2)c_2.$$

Mivel $M_2 > M_1$ ezért leoszthatjuk mindkét oldalt $M_1 - M_2$-vel. Átrendezés után megkapjuk a feltételezésünket. □

2.6.7. 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 2.6.5. 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 konstrálni P periódussal, akkor minden $P' > P$-re szintén tudunk ütemezést konstruálni P' periódussal. □

2.6.8. 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 2.6.5. 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 2.6.5. lemma miatt tudunk egy $\mathcal{S}_2$ végrehajtható P_2-periodikus k_2-ütemezést konstruálni, ahol $P_2 := f_1M_1 + f_2M_2 = A_2M_1 - A_1M_2 = f_1A_1 + f_2A_2$ 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 vegyük 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 2.6.5. 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 egyszerüsítés 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. □

2.6.9. 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.*

2.6.10. Következmény. *A 13. 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 2.6.7. lemma miatt elegendő azon ütemezéseket vizsgálni, melyek kizárólag minimális ütemezési osztályokat használnak, továbbá a 2.6.8. lemma miatt leredukáltuk a problémát arra, amikor amikor legfeljebb két minimális ütemezési osztályt használhatunk. Mivel véges előütemezés illetve előütemezési párok vannak, továbbá a 2.6.6. 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. □

2.6.11. Tétel. *A 13. 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 2.6.10. következmény szerint legyen definiálva. Az OPT_∞ definíciója miatt léteznie kell egy $\mathcal{S}$ végrehajtható ∞-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 recproka kisebb, mint OPT_K, ami ellentmond K definíciójának. □

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

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

Bizonyítás. Vegyük a következő aciklikus gráfot: $V := \bigcup_{i=1}^{k} \{u_i, v_i, w_i\} \cup \{x\}$ és $E := \bigcup_{i=1}^{k-1} \{(u_i, u_{i+1}), (v_i, v_{i+1}), (w_i, w_{i+1})\} \cup \{(u_k, v_1), (x, w_1)\}$.

Ekkor két minimális ütemezési osztály van: $A_1 = k - 1, M_1 = 2k - 2$ és $A_2 = k, M_2 = k - 1$. A 2.6.6. lemmából tudjuk, hogy $OPT_k = k - \frac{1}{k}$, habár $OPT_1 = k$. □

2.7. Approximációs algoritmusok

Ebben a szekcióban megadunk egy approximációs algoritmust két modellre. A többi modellben az approximációs algoritmusokat onnan kapjuk, hogy már beláttuk, hogy ekvivalens egy ismert problémával, amire létezik már approximációs algoritmus.

2.7.1. Tétel. *A 11. és 13. modellben létezik 4- illetve 6-approximációs algoritmus az* OPT_1*-re nézve.*

Bizonyítás. A 2.6.5. 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 1-ütemezést készíteni. Megadunk egy mohó algoritmust, amiről belátjuk, hogy 4- illetve 6-approximációs algoritmus az előbbi kifejezésre nézve.

Legyen adott a $G(V, E)$ aciklikus gráfunk. Legyen $V' := V$, illetve legyen $\mathcal{M}$ és $\mathcal{A}$ azon match és action csúcsok halmaza, amiknek befoka 0. A 13. modell esetén $\mathcal{M}$ és $\mathcal{A}$ halmazokat rendezzük szélesség szerint csökkenő sorrendbe. Az i-ik lépésben az i-ik órajelbe ágyazzuk be $\mathcal{M}$ elemeit addig, amíg $\mathcal{M}$ nem az üres halmaz, vagy a következő csúcs hozzávételével a szélességek összege meg nem haladná $\overline{M}$ értékét. Ezután szintén az i-ik órajelbe ágyazzuk be $\mathcal{A}$ elemeit ugyanígy $\overline{A}$ figyelembevételével. A lépés végén először V', $\mathcal{M}$ és $\mathcal{A}$ halmazokból távolítsuk el azon csúcsokat, amiket most beágyaztunk. Ezután vegyük hozzá $\mathcal{M}$-hez és $\mathcal{A}$-hoz azon match és action csúcsokat V'-ből, amiknek befoka 0 lett, mindezt (a 13. modellben) úgy, hogy $\mathcal{M}$-ben és $\mathcal{A}$-ban továbbra is szélesség szerinti csökkenő sorrenden legyenek a csúcsok. Ezt addig folytatjuk, amíg V' az üres halmaz nem lesz. Az algoritmus végén kapunk egy S előütemezést, amiből a 2.6.5. lemma értelmében készíthető egy végrehajtható $\max(A, M)$-periodikus 1-ütemezés, ahol A és M azon órajelek száma, ahová beágyaztunk action vagy match csúcsot.

Legyen j_1 egy utolsó órajelben lévő csúcs. Legyen j_2 az (egyik) utolsó csúcs, ami össze van kötve j_1-el a G gráfban. Definiáljuk ugyanígy j_i-t, addig, amíg nem érkezünk el egy forrás (0 befokú) csúcshoz G-ben. Az utolsó j_i csúcsot jelöljük j_k-val. Jelölje $J_m := \{t | \exists j_i \text{ match csúcs} : S(j_i) = t\}$, azaz azon órajelek halmaza, amiben szerepel egy j_i, ami match csúcs. Definiáljuk J_a halmazt ugyanígy.

Jelölje M^*, A^* azon órajelek halmazát, amikor egy adott órajel a 11. modellben match vagy action csúcsokkal tele van (azaz pontosan $\overline{M}$ darab match csúcs szerepel), a 13. modellben pedig legalább félig tele van (azaz legalább $\frac{\overline{M}}{2}$ darab match csúcs szerepel).

Belátjuk, hogy minden $T(S)$-nél nem nagyobb órajel L_m-ben, vagy L_a-ban, vagy M^*-ban vagy A^*-ban van. Vegyük észre, hogy ha pl. j_i match csúcs, továbbá j_{i+1} és j_i között van néhány órajel, amiben szerepel match csúcs, akkor ezen órajelek a match csúcsra nézve a 11. modellben tele, a 13. modellben legalább félig tele vannak. Ha nem így lenne, akkor j_{i+1}-et korábbi órajelbe is be tudtuk volna ágyazni. (Itt a 13. modell esetén kihasználjuk, hogy $\mathcal{M}$ elemei szélességi értékek szerint csökkenő sorrendbe voltak rendezve, így ha $\mathcal{M}$-ből beágyaztunk néhány csúcsot, de nem az összeset, akkor legalább félig tele lett match szerint az órajel.) Ugyanez elmondható, ha j_i action csúcs. Így j_{i+1} és j_i között minden órajel M^*-ban vagy A^*-ban van. Ugyanezen indoklással az is igaz, hogy j_k előtt minden órajel M^*-ban vagy A^*-ban van. Ezzel mindkét modellben azt kapjuk, hogy

$$\max(A, M) \leq T(S) \leq |J_m| + |J_a| + |M^*| + |A^*|.$$

Világos, hogy $|J_m|, |J_a| \leq OPT_1$, továbbá a 11. modellben $|M^*|, |A^*| \leq OPT_1$, míg a 13. modellben $|M^*|, |A^*| \leq 2 \cdot OPT_1$ Ebből pedig a 11. modell esetén $\max(A, M) \leq 4 \cdot OPT_1$, a 13. modell esetén pedig $\max(A, M) \leq 6 \cdot OPT_1$. □

2.8. Szimulációk

A [2] cikkben található egy szimulációs program és példák, ami segítségével létre lehet hozni a 13. modellben (illetve $IPC = 2$ esetén) dRMT ütemezéseket. A szimuláció a következőképp néz ki vázlatosan: Lefixálnak egy P periódus értéket és keresik a minimális T hosszt. 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 kétszerese.)

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 határoznak meg.

A program segítségével azt vizsgáltam, hogy a kezdeti megoldást beadva vagy nem beadva az IP megoldónak, milyen gyorsan oldja meg három valódi (P4 programból generált) gráfra fix P esetén. Az eredményeket a 2.5. táblázat tartalmazza.

						Futási idők (mp) kezdeti értékkel		Futási idők (mp) kezdeti érték nélkül	
Modell neve	|V|	|E|	P	Kritikus út hossza	Optimális mo. T-re	tetsz. mo.	opt. mo.	tetsz. mo.	opt. mo
egress	104	291	11	197	217	<1	(2100+) up: 218, lb:211	100	1073
ingress	224	930	17	243	245	<1	36	260	260
combined	328	1221	21	243	243	<1	2260	1400	1600

2.5. táblázat.

A tapasztalat az, hogy ha csak egy tetszőleges megoldást akarunk találni, akkor egy majdnem végrehajtható ütemezést beadva az IP megoldó nagyon gyorsan megtalál egy végrehajtható ütemezést, ellentétben, amikor nem adunk be neki semmit. Ha azonban optimális megoldást keresünk, akkor háromból egy esetnél végzett csak gyorsabban a megoldó. (Az első esetnél ennyi idő alatt csak alsó és felső becslést tudott mondani az IP megoldó.)

2.9. Nyitott kérdések

Számos kérdés továbbra is nyitott maradt. Ezeket a kérdéseket ebben a szekcióban foglalom össze.

1. Igaz-e, hogy a 6., 7. és 14. modellben minden dRMT feladatra létezik $K \in \mathbb{N}^+$, hogy $OPT_K = OPT_\infty$?

2. NP-nehéz probléma-e a 14. modell a periódusra nézve, ha igen, akkor milyen közelítő algoritmust tudunk mondani rá?

3. NP-nehéz probléma-e a 6. modell akkor is, ha nem az OPT_1-re vagyunk kíváncsiak, hanem OPT_k-ra?

4. 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?

Irodalomjegyzék

[1] Programming protocol-independent packet processors. `https://p4.org/`.

[2] 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.

[3] M. R. Garey and David S. Johnson. Computers and intractability: A guide to the theory of NP-completeness. 1978.

[4] Lavanya Jose, Lisa Yan, George Varghese, and Nick McKeown. Compiling packet programs to reconfigurable switches. In *USENIX NSDI*, pages 103–115, 2015.

[5] Jordán Tibor. Ütemezés: elmélet és algoritmusok. `https://web.cs.elte.hu/~jordan/utemezes/UtemezesJegyzet2020okt.pdf`, 2020.

[6] 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.