CYBERNETICA
Institute of Information Security

Ründepuud: pooladaptiivne mudel ja ligikaudsed arvutused

Jan Willemson, Aivo Jürgenson

T-4-4 / 2009

Jan Willemson[1], Aivo Jürgenson[2].
[1] AS Cybernetica, Institute of Information Security
[2] Tallinn University of Technology, Institute of Informatics

Cybernetica research reports are available online at
`http://research.cyber.ee/`

Mailing address:
AS Cybernetica
Akadeemia tee 21
12618 Tallinn
Estonia

Ründepuud: pooladaptiivne mudel ja ligikaudsed arvutused

Jan Willemson, Aivo Jürgenson

26. veebruar 2009. a.

Kokkuvõte

Käesolev raport kirjeldab uut ründepuude arvutusmudelit, mis püüab senisest täpsemini modelleerida reaalses elus aset leidvaid olukordi. Mudel võimaldab ründajal valida parima elementaarrünnete järjekorra ning käsitleb selliseid blokeeruvaid elementaarründeid, mille ebaõnnestumisel ebaõnnestub kogu ründepuu.

1 Sissejuhatus

Ründepuu kujutab endast mugavat formalismi mingi (info)vara vastaste võimalike rünnete klassifitseerimiseks ja analüüsiks. Sarnast metoodikat kasutati kriitiliste süsteemide veaohtlikkuse hindamisel juba 1980ndate aastate algul [VGRH81]; infosüsteemide turvaanalüüsi tõi selle esimesena Weiss [Wei91] ning laiema populaarsuse omandas meetod pärast Bruce Schneieri artiklit [Sch99].

Kuigi juba Weiss [Wei91] mõistis, et ründepuu tippudel on praktikas palju parameetreid, keskendusid varased uurimused puudele, millel vaadeldi vaid üht parameetrit (ründe hind, vajalik oskuste tase vms) [Sch99, MO05]. Olulise sammu ründaja otsustusprotsessi täpsema kirjeldamise poole tegid 2006. aastal Buldas jt [BLP+06], tuues sisse mitteparameetrilised ründepuud. Sama töörühm on hiljem seda meetodit praktikas rakendanud [BM07] ning laiendanud juhule, kus parameetrite väärtused pole arvud, vaid vahemikud [JW07]. 2008. aastal tõid Jürgenson ja Willemson välja Buldase jt mudeli kitsaskohad [JW08] ning lõid alternatiivse puuarvutussemantika, mis võimaldab ründaja ootetulu palju täpsemalt hinnata ning on samas kooskõlas Mauw & Oostdijk´i üldise raamistikuga [MO05]. Uue semantika tõsine puudus võrreldes Buldase jt omaga, on arvutuslik ebaefektiivsus. Sellele probleemile pühendame peatüki käesoleva aruande lõpuosas.

Teine ja mõnes mõttes suurem probleem, mis käesoleva uurimuse motiveeris, on ühine praktiliselt kõigile olemasolevatele ründepuude mudelitele, sh kõigile ülalviidatutele. Nimelt lähtuvad kõik senised artiklid millestki, mida võib nimetada *paralleelseks paradigmaks*. Juba alates Weissist ja Schneierist [Wei91, Sch99] on ründepuudele tugineva analüüsi eesmärk olnud selgitada välja kriitiline (st mingis mõttes kõige ohtlikum) alampuu ning üritada siis rünnet võimalikuks tegevaid nõrkusi vähendada. Kõik olemasolevad mudelid eeldavad aga, et ründaja üritab kõiki kriitilisse alampuusse kuuluvaid elementaarründeid realiseerida *korraga* st ajalist järgnevust ignoreerides. Nõnda alahinnatakse sisuliselt ründaja võimalusi ning lõpuks ka ootetulu. Näiteks kui kriitiline rünne eeldab kaupluseakna vaikset purustamist ja sealtkaudu sissehiilimist, siis pärast seda kui akna katkilöömise järel signalisatsioon tööle hakkab (elementaarrünne õnnestub, kuid ründaja on sellega avastatud), pole sissepugemine enam mõistlik ning sellest sammust loobudes kahandab varas oodatavaid trahve, suurendades nii oodatavat tulu.

Käesolevas uurimuses keskendumegi sellistele ründepuude semantikatele, kus ründaja võib mõnede eelmiste elementaarrünnete õnnestumisest või ebaõnnestumisest lähtudes teha otsuseid selle kohta, milliseid elementaarründeid järgmiseks proovida. Ka siin on võimalikud mitmed lähenemised. Kõige üldisemas mudelis võib ründaja igal sammul valida suvalise veel proovimata elementaarründe; niisugust lähenemist nimetame *täisadaptiivseks*. See mudel on hetkel analüüsimiseks veel liiga keeruline, mistõttu käesolevas uurimuses keskendume *pooladaptiivsele* juhule, kus ründaja fikseerib algul elementaarrünnete järjekorra ning võib hiljem teha vaid otsuse mõnda neist vahele jätta. Globaalselt optimaalse ründe leidmiseks tuleb siis läbi vaadata elementaarrünnete hulga kõigi alamhulkade kõikvõimalikud järjestused. Naiivse algoritmi keerukus on seega meeletu, aga ka optimeerimine jääb praegu põhiliselt tulevase uurimistöö hooleks.

Teine probleem, mille elementaarrünnete ajalise järjestuse vaatlemine endaga kaasa toob, on küsimus tippude ja kogu puu blokeerumisest. Praktikas on olemas kahte liiki karistusi — selliseid, mis võimaldavad pärast kohaldamist ründajal edasi tegutseda (trahv, tingimisi karistus), ja sellised, mis ei võimalda (reaalne vangistus, surmanuhtlus). Vastavalt sellele saame ka ründepuu arvutusmudelit reaalsusele lähendada, kui lubame mõnedel elementaarrünnetel puuarvutusprotsessi blokeerida juhul, kui nad ebaõnnestuvad. Paneme tähele, et paralleelses paradigmas blokeerumine erilist rolli ei mängi, sest tipu blokeerumine mõjutab ainult neid arvutuse samme, mis tulevad *pärast* blokeerumist.

Käesolevas uurimuses esitame põhitulemused mitteblokeeruva juhu jaoks (st kus blokeeruvaid tippe ei ole). Mitmed neist tulemustest saab otse üle kanda ka blokeeruvale juhule (st kus osad, kuid mitte tingimata kõik ele-

mentaarründed blokeerivad ebaõnnestumise korral edasise töö). Tuleb aga tõdeda, et blokeeruval juhul jääb hetkel lahtiseks suurem osa küsimusi kui mitteblokeeruval juhul.

2 Ründepuuarvutused

Oma olemuselt on ründepuu tavaline AND-OR-puu, mille lehed $X_1, \ldots, X_n$ väljendavad elementaarründeid ning teised tipud keerulisemaid kombineeritud ründeid [Sch99]. Järgides Buldase jt mudelit [BLP+06], vaatleme järgmisi elementaarrünnete parameetreid:

- Cost_i — ründe X_i maksumus,
- p_i — ründe X_i õnnestumise tõenäosus,
- π_i^+ — ootetrahv juhul kui rünne X_i õnnestub,
- π_i^- — ootetrahv juhul kui rünne X_i ei õnnestu.

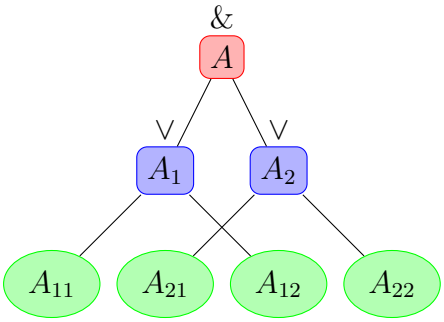
Lisaks on defineeritud globaalne parameeter Gains, millise tulu saab ründaja siis, kui tal õnnestub realiseerida ründepuu juur. Osutub, et senistes seda mudelit järgivateks artiklites [BLP+06, BM07, JW07, JW08] võib nelja ülaltoodud leheparameetri asemel tegelikult vaadelda kahte parameetrit:

- p_i — ründe õnnestumise tõenäosus,
- $\mathsf{Expenses}_i = \mathsf{Cost}_i + p_i \cdot \pi_i^+ + (1 - p_i) \cdot \pi_i^-$ — ründe ootekulu.

Etteruttavalt võib öelda, et sama kehtib ka käesoleva uurimuse kohta, kuid see on omaette mittetriviaalne tulemus.

Üldine skeem, mille alusel ründaja oma ootetulu hindab, on järgmine.

1. Koosta ründepuu elementaarrünnete hulgaga $\mathcal{X} = \{X_1, \ldots, X_n\}$.
2. Vali juurrünnet realiseeriv alamhulk $S \subseteq \mathcal{X}$ ja moodusta alampuu.
3. Vali hulga S permutatsioon α.
4. Hinda leitud alampuust valitud permutatsioonist lähtuvalt ootetulu.
5. Vali üle kõigi S ja α valikute maksimaalne ootetulu.



Joonis 1: Ründepuu $A = (A_{11} \vee A_{12})\&(A_{21} \vee A_{22})$

.

Käesolevas uurimuses keskendume eelneva loetelu 4. punktile, kuid kogu meetodi edu sõltub loomulikult suuresti sellest, kui hästi me suudame optimeerida ka punkte 2 ja 3 kas ligikaudsete arvutuste abil või mingite kriteeriumite alusel lootusetute variantide vaatluse alt välja jätmisega (vt [JW08], Teoreem 1).

Kuna ootetulu hindamisel on oluline ainult vaadeldav alampuu, eeldame järgnevas üldsust kitsendamata, et $S = \mathcal{X}$. Samuti anname kõik algoritmid ja näited kahendpuude jaoks.

Ootetulu hindamise üldine skeem pooladaptiivsel juhul on järgmine:

1. Kui järgmine permutatsiooni α poolt määratud elementaarrünne ei saa juurtippu mõjutada, jäta see vahele ja alusta järgmise ründega uuesti.

2. Kui vaadeldava elementaarründe õnnestumine realiseerib juurründe, lisa selle haru tulemus üldsummasse, muidu võta järgmine rünne.

3. Kui vaadeldava elementaarründe ebaõnnestumine toob kaasa juurründe ebaõnnestumise, lisa selle haru tulemus üldsummasse, muidu võta järgmine rünne.

Paneme tähele, et permutatsiooni valiku tõttu võib juurrünne ebaõnnestuda tänu sellele, et oleme leidnud kriitilise ebaõnnestunud alamhulga, aga ka tänu sellele, et vaadeldav tipp blokeeris arvutused. Järgnevalt formaliseerime selle algoritmi mitteblokeeruva juhu jaoks.

3 Pooladaptiivne mitteblokeeruv juht

Enne arvutusalgoritmi formaalset esitamist teeme läbi väikese näite. Vaatleme puud joonisel 1.

Lehtede järjestus vasakult paremale väljendab siin permutatsiooni α. On olemas järgmised võimalused:

1. A_{11} õnnestub, A_{21} õnnestub. Edasi pole vaja vaadata, sest juurtipp on realiseeritud. Selle võimaluse tõenäosus on $p_{11} \cdot p_{21}$ ning tulemus ründajale $\mathsf{Gains} - \mathsf{Cost}_{11} - \pi_{11}^{+} - \mathsf{Cost}_{21} - \pi_{21}^{+}$. (Siinkohal ja edaspidi eeldame, et kõik õnnestumiste tõenäosused on sõltumatud.)

2. A_{11} õnnestub, A_{21} ebaõnnestub, (A_{12} jääb vahele), A_{22} õnnestub; tõenäosus $p_{11} \cdot (1 - p_{21}) \cdot p_{22}$, tulemus $\mathsf{Gains} - \mathsf{Cost}_{11} - \pi_{11}^{+} - \mathsf{Cost}_{21} - \pi_{21}^{-} - \mathsf{Cost}_{22} - \pi_{22}^{+}$.

3. A_{11} õnnestub, A_{21} ebaõnnestub, (A_{12} jääb vahele), A_{22} ebaõnnestub; tõenäosus $p_{11} \cdot (1 - p_{21})(1 - p_{22})$, tulu $-\mathsf{Cost}_{11} - \pi_{11}^{+} - \mathsf{Cost}_{21} - \pi_{21}^{-} - \mathsf{Cost}_{22} - \pi_{22}^{-}$. Paneme tähele, et juurründe ebaõnnestumise tõttu jääb Gains saamata.

4. A_{11} ebaõnnestub, A_{21} õnnestub, A_{12} õnnestub; tõenäosus $(1 - p_{11}) \cdot p_{21} \cdot p_{12}$, tulu $\mathsf{Gains} - \mathsf{Cost}_{11} - \pi_{11}^{-} - \mathsf{Cost}_{21} - \pi_{21}^{+} - \mathsf{Cost}_{12} - \pi_{12}^{+}$.

5. A_{11} ebaõnnestub, A_{21} õnnestub, A_{12} ebaõnnestub; tõenäosus $(1 - p_{11}) \cdot p_{21} \cdot (1 - p_{12})$, tulu $-\mathsf{Cost}_{11} - \pi_{11}^{-} - \mathsf{Cost}_{21} - \pi_{21}^{+} - \mathsf{Cost}_{12} - \pi_{12}^{-}$.

6. A_{11} ebaõnnestub, A_{21} ebaõnnestub, A_{12} õnnestub, A_{22} õnnestub; tõenäosus $(1 - p_{11})(1 - p_{21}) \cdot p_{12} \cdot p_{22}$, tulu $\mathsf{Gains} - \mathsf{Cost}_{11} - \pi_{11}^{-} - \mathsf{Cost}_{21} - \pi_{21}^{-} - \mathsf{Cost}_{12} - \pi_{12}^{+} - \mathsf{Cost}_{22} - \pi_{22}^{+}$.

7. A_{11} ebaõnnestub, A_{21} ebaõnnestub, A_{12} õnnestub, A_{22} ebaõnnestub; tõenäosus $(1 - p_{11})(1 - p_{21}) \cdot p_{12} \cdot (1 - p_{22})$, tulu $-\mathsf{Cost}_{11} - \pi_{11}^{-} - \mathsf{Cost}_{21} - \pi_{21}^{-} - \mathsf{Cost}_{12} - \pi_{12}^{+} - \mathsf{Cost}_{22} - \pi_{22}^{-}$.

8. A_{11} ebaõnnestub, A_{21} ebaõnnestub, A_{12} ebaõnnestub; tõenäosus $(1 - p_{11})(1 - p_{21})(1 - p_{12})$, tulu $-\mathsf{Cost}_{11} - \pi_{11}^{-} - \mathsf{Cost}_{21} - \pi_{21}^{-} - \mathsf{Cost}_{12} - \pi_{12}^{-}$.

Ründaja ootetulu avaldub siis kujul

$$
\begin{aligned}
p_{11}\cdot p_{21} \quad &\cdot \quad (\mathsf{Gains}-\mathsf{Cost}_{11}-\pi^{+}_{11}-\mathsf{Cost}_{21}-\pi^{+}_{21})\\
+p_{11}\cdot(1-p_{21})\cdot p_{22} \quad &\cdot \quad (\mathsf{Gains}-\mathsf{Cost}_{11}-\pi^{+}_{11}-\mathsf{Cost}_{21}-\pi^{-}_{21}\\
&\qquad -\mathsf{Cost}_{22}-\pi^{+}_{22})\\
+p_{11}\cdot(1-p_{21})(1-p_{22}) \quad &\cdot \quad (-\mathsf{Cost}_{11}-\pi^{+}_{11}-\mathsf{Cost}_{21}-\pi^{-}_{21}\\
&\qquad -\mathsf{Cost}_{22}-\pi^{-}_{22})\\
+(1-p_{11})\cdot p_{21}\cdot p_{12} \quad &\cdot \quad (\mathsf{Gains}-\mathsf{Cost}_{11}-\pi^{-}_{11}-\mathsf{Cost}_{21}-\pi^{+}_{21}\\
&\qquad -\mathsf{Cost}_{12}-\pi^{+}_{12})\\
+(1-p_{11})\cdot p_{21}\cdot(1-p_{12}) \quad &\cdot \quad (\mathsf{Gains}-\mathsf{Cost}_{11}-\pi^{-}_{11}-\mathsf{Cost}_{21}-\pi^{+}_{21}\\
&\qquad -\mathsf{Cost}_{12}-\pi^{+}_{12})\\
+(1-p_{11})(1-p_{21})\cdot p_{12}\cdot p_{22} \quad &\cdot \quad (\mathsf{Gains}-\mathsf{Cost}_{11}-\pi^{-}_{11}-\mathsf{Cost}_{21}-\pi^{-}_{21}\\
&\qquad -\mathsf{Cost}_{12}-\pi^{+}_{12}-\mathsf{Cost}_{22}-\pi^{+}_{22})\\
+(1-p_{11})(1-p_{21})\cdot p_{12}\cdot(1-p_{22}) \quad &\cdot \quad (-\mathsf{Cost}_{11}-\pi^{-}_{11}-\mathsf{Cost}_{21}-\pi^{-}_{21}\\
&\qquad -\mathsf{Cost}_{12}-\pi^{+}_{12}-\mathsf{Cost}_{22}-\pi^{-}_{22})\\
+(1-p_{11})(1-p_{21})(1-p_{12}) \quad &\cdot \quad (-\mathsf{Cost}_{11}-\pi^{-}_{11}-\mathsf{Cost}_{21}-\pi^{-}_{21}\\
&\qquad -\mathsf{Cost}_{12}-\pi^{-}_{12}).
\end{aligned}
$$

Kogu avaldist sobivalt teisendades leiame, et tema väärtus on

$$
\begin{aligned}
&(p_{11}+p_{12}-p_{11}p_{12})(p_{21}+p_{22}-p_{21}p_{22})\cdot\mathsf{Gains}\\
-&\mathsf{Expenses}_{11}-\mathsf{Expenses}_{21}-(1-p_{11})\cdot\mathsf{Expenses}_{12}\\
-&(p_{11}+p_{12}-p_{11}p_{12})(1-p_{21})\cdot\mathsf{Expenses}_{22} \quad .
\end{aligned}
$$

Peatselt näeme, et selline ilus avaldis pole juhuslik ning et mitteblokeeruval juhul võimaldab ta anda väga efektiivse arvutuseeskirja. Blokeeruva juhu jaoks pole autoritel veel õnnestunud analoogilist efektiivset algoritmi konstrueerida, seepärast analüüsime läbi ka ebaefektiivse algoritmi.

Kõik algoritmid peavad suutma arvet pidada selle üle, kas antud leht on antud situatsioonis oluline või mitte. Selleks võtame kasutusele kolmevalentse loogika, kus lisaks tavalistele väärtustele `t (true)` ja `f (false)` on kasutusel ka väärtus `u (undefined)`. Arvutusreeglid on defineeritud tabelis 1.

Järgnev algoritm $\mathsf{compute\_outcome}(A,i,p)$ on rekursiivne. Globaalselt kasutab ta ründepuu põhjal moodustatud Boole´i valemit $\mathcal{F}$ ja muutujate $\mathcal{X}_1,\ldots,\mathcal{X}_n$ indeksite permutatsiooni α. Igal rekursiooni sammul on sisendiks muutujate algne väärtus A antud rekursiooni harus (algselt $[u,u,\ldots,u]$), sügavuse loendur i (algselt 1) ja haru tõenäosus p (algselt 1). Seega on algne väljakutse kujul $\mathsf{compute\_outcome}([u,u,\ldots,u],1,1)$.

&	t	f	u
t	t	f	u
f	f	f	f
u	u	f	u

∨	t	f	u
t	t	t	t
f	t	f	u
u	t	u	u

Tabel 1: Tõeväärtustabelid & ja ∨ tehete jaoks kolmevalentses loogikas.

Algoritm 1 Permutatsiooni α tulususe arvutamine

Sisend : Muutujate väärtustus A, sügavuse loendur i, haru tõenäosus p
Väljund: Permutatsiooni otsitav tulusus *sum*

$sum := 0$;
if $\mathcal{F}(A)$ *väärtustamisel omandab mõni tipp lehest* $X_{\alpha(i)}$ *puu juureni väärtuse* t *või* f **then**
    `compute_outcome`$(A,\ i+1,\ p)$;
    return *sum*;
end
$A[\alpha(i)] := \mathsf{t}$;
if $\mathcal{F}(A) = \mathsf{t}$ **then**
    $sum := sum + p \cdot p_{\alpha(i)} \cdot \Big[\mathsf{Gains} - \sum_{j \in A}(\mathsf{Cost}_j + \pi_i^j)\Big]$;
else
    `compute_outcome`$(A,\ i+1,\ p \cdot p_{\alpha(i)})$;
end
$A[\alpha(i)] := \mathsf{f}$;
if $\mathcal{F}(A) = \mathsf{f}$ **then**
    $sum := sum + p \cdot (1 - p_{\alpha(i)}) \cdot \Big[-\sum_{j \in A}(\mathsf{Cost}_j + \pi_i^j)\Big]$;
else
    `compute_outcome`$\big(A,\ i+1,\ p \cdot \big(1 - p_{\alpha(i)}\big)\big)$;
end
return *sum*;

Siin

$$\mathsf{Cost}_j + \pi_i^j = \begin{cases} 0, & \text{kui } A[j] = u \\ \mathsf{Cost}_j + \pi_j^+, & \text{kui } A[j] = t \\ \mathsf{Cost}_j + \pi_j^-, & \text{kui } A[j] = f \end{cases}$$

ning muutuja *sum* väärtus sisaldab lõpuks otsitavat ootetulu.
Selle algoritmi analüüsil osutuvad kasulikuks järgmised tähelepanekud.

1. Igal rekursiivsel väljakutsel kehtib $A[\alpha(i)] = u$. Tõepoolest, see seos kehtib esimesel väljakutsel ja igal uuel kutsel indeks i suureneb, teisi väärtusi peale u aga on saanud ainult elemendid $A[(\alpha(1)], \ldots, A[(\alpha(i)]$.

2. Igal rekursiivsel väljakutsel kehtib $\mathcal{F}(A) = u$. Jällegi on selle väite kehtivus esimesel väljakutsel ilmne. **if-then** direktiiv 2 kuni 5 real ei muuda A väärtust. Pärast omistamist algoritmi 6 real saab AND-OR-puu monotoonsuse tõttu juur omandada ainult väärtuse t või u . Rekursiooniga minnakse edasi ainult teisel juhul. Analoogiline arutelu kehtib teises harus ridadel 12–17.

3. Kui väljakutsel parameetritega (A, i, p) minnakse rekursiooniga sügavamale, on mõni tipp teel lehest $\mathcal{X}_{\alpha(i)}$ kuni juureni $\mathcal{F}(A)$ väärtustamisel kas `t` või `f` (kus A on see väärtustus, millega uude rekursiooni minnakse). See väide on ilmne **if-then** direktiivi jaoks 2–5 ridadel, aga ka harude jaoks ridadel 6–11 ja 12–17, sest siis vastavalt $A[\alpha(i)] = t$ ja $A[\alpha(i)] = f$.

4. Kui väljakutsel parameetritega (A, i, p) minnakse rekursiooniga sügavamale, on harude tõenäosuste summa p. Ka see väide on tõene, kui järgneb ainult üks haru ridadel 2–5. Kui minnakse harudesse ridadel 6–11 ja 12–17, on nende tõenäosused vastavalt $p \cdot p_{\alpha(i)}$ ja $p \cdot (1 - p_{\alpha(i)})$, nende summa on p (siin loeme ka mitterekursiivsed direktiivid ridadel 8 ja 14 vastava tõenäosusega harudeks).

Esimene vastamist vajav küsimus on loomulikult algoritmi korrektsus.

Teoreem 1. *Algoritm 1 lõpetab oma töö korrektselt.*

Tõestus. Igal rekursiooni sammul suurendatakse parameetrit i ühe võrra. Teoreem saab tõestatud, kui näitame, et i suurim võimalik väärtus on n (see on muuhulgas ka massiivi A suurus). Oletame vastuväiteliselt, et me oleme massiivi A kõik n elementi läbi vaadanud, aga algoritm sunnib meid minema sügavusele $n+1$. Eelpool tehtud tähelepaneku 2 põhjal peab kehtima

$\mathcal{F}(A) = u$. Samas tähelepaneku 3 põhjal peab igal teel lehest $\mathcal{X}_i$ kuni juureni olema mõni tipp, mille väärtus on t või f. Seega ei saa juurtipu väärtus olla u, vastuolu. □

Järgnevalt tõestame, et Algoritmi 1 järgi arvutatud ootetulu avaldub tegelikult väga kompaktsel kujul.

Teoreem 2. *Algoritm 1 järgi arvutatud ootetulu avaldub kujul*

$$p_\alpha \cdot \mathsf{Gains} - \sum_{i=1}^{n} p_{\alpha,i} \cdot \mathsf{Expenses}_i,$$

kus $p_\alpha, p_{\alpha,i} \in [0,1]$ $(i = 1, \ldots, n)$.

Tõestus. Uurime kõigepealt parameetri Gains kordajat p_α. Kuna see avaldub teatud rekursiooniharude tõenäosuste summana, siis kehtib $p_\alpha \geq 0$. Teisalt on eelpool tehtud 4. tähelepaneku põhjal kõigi harude tõenäosuste summa 1, järelikult $p_\alpha \leq 1$.

Liikmete $p_{\alpha,i} \cdot \mathsf{Expenses}_i$ kuju tõestamiseks paneme tähele, et iga kord kui üldsummasse lisatakse liige $p \cdot p_i \cdot (\mathsf{Cost}_i + \pi_i^+)$, liidetakse sinna ka $p \cdot (1 - p_i)(\mathsf{Cost}_i + \pi_i^-)$; mõlemad sündmused juhtuvad kas vastavalt Algoritmi 1 ridadel 8 ja 14 või rekursiivsetel kutsetel vastavalt ridadel 10 ja 16, kasutades ülaltoodud 4. tähelepanekut. Tõestus, et $p_{\alpha,i} \in [0,1]$ on analoogiline parameetri p_α väärtuspiirkonna tõestusega. □

Kuna p_α on kõigi edukate rekursiooniharude tõenäosuste summa, väljendab see parameeter juurründe õnnestumise tõenäosust. See suurus ei sõltu muuhulgas permutatsioonist α. Saadud tulemus väärib eraldi väljatoomist.

Teoreem 3. *Suvalise kahe permutatsiooni* $\alpha_1, \alpha_2 \in S_n$ *korral*

$$p_{\alpha_1} = p_{\alpha_2}.$$

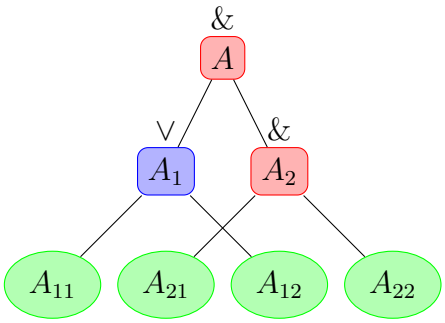
□

Artiklis [JW08] on antud ka efektiivne algoritm p_α arvutamiseks.

Parameetritega $p_{\alpha,i}$ on lugu keerulisem, sisuliselt väljendavad nad tõenäosust, et permutatsiooni α korral arvutustega leheni X_i jõutakse. Need suurused sõltuvad kasutatavast permutatsioonist juba olulisel määral ning nende efektiivne arvutamine on käesoleva uuringuaruande üks põhiküsimusi.

Enne selle küsimuse juurde asumist tõestame veel mõned huvitavad tulemused. Tähistame

$$\mathsf{Outcome}_\alpha = p_\alpha \cdot \mathsf{Gains} - \sum_{i=1}^{n} p_{\alpha,i} \cdot \mathsf{Expenses}_i.$$



A_{11}	A_{21}	A_{12}	A_{22}
+	+		+
+	+		−
+	−		
−	+	+	+
−	+	+	−
−	+	−	
−	−		

Joonis 2: Ründepuu $A = (A_{11} \vee A_{12})\&(A_{21}\&A_{22})$ ning permutatsiooni $(A_{11}, A_{12}, A_{21}, A_{22})$ stsenaariumid. Tabelis olev "+" tähistab lehe tõest, "–" väära ja tühik vahelejäänud väärtustust.

Samuti tuletame meelde, et lehtede hulga $\mathcal{X}$ alamhulga $S \subseteq \mathcal{X}$ jaoks tähendadas artiklis [JW08] toodud semantika korral $\mathsf{Outcome}_S = p_S \cdot \mathsf{Gains} - \sum_{i=1}^{n} \mathsf{Expenses}_i$. Nüüd on lihtne näha, et kehtib järgmine teoreem.

Teoreem 4. $\mathsf{Outcome}_\alpha \geq \mathsf{Outcome}_\mathcal{X}$.

Tõestus. Eelpool nägime, et $p_\mathcal{X} = p_\alpha$ iga $\alpha \in S_n$ korral. Samuti teame Teoreem 2 põhjal, et $\forall\alpha\forall i p_{\alpha i} \leq 1$. Tõestatav võrratus on nüüd ilmne. □

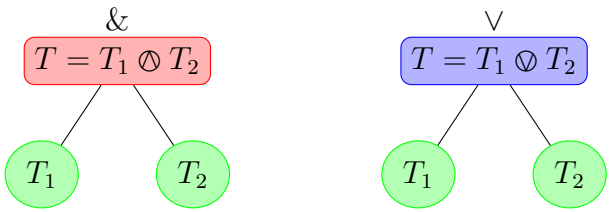
Nagu alguses märgitud, pole eeldus $S = \mathcal{X}$ üldsust kitsendav. Seega näitab teoreem 4 formaalselt, et lehtede ümberjärjestamine võib ründaja jaoks ootetulu artikli [JW08] tulemusega võrreldes ainult parandada (ja üldjuhul parandabki).

Järgmiseks analüüsime Algoritmi 1 ajalist keerukust. Ühest küljest on selles algoritmis igal rekursiivsel kutsel ülimalt kaks rekursiivset haru, seega võib oodata halvima juhu eksponentsiaalset keerukust ründepuu lehtede arvu järgi. Ja tõepoolest, üsna lihtsalt saab konstrueerida puude pere, mis tõesti eksponentsiaalset ajahulka nõuab.

Järgnevas näitame aga veel rohkem, nimelt Algoritmi 1 keskmist eksponentsiaalset keerukust teatud suurel puude ja permutatsioonide klassil.

Selleks, et Algoritmi 1 keerukusest paremini rääkida, toome sisse stsenaariumi mõiste. Iga rekursiooni haru, mis lõppeb Algoritmi 1 real 7 või 11 defineerib massiivi A hetkeseisuga ühe stsenaariumi, mis on vastavalt edukas või ebaedukas. Näiteks joonisel 2 on toodud ründepuu, mille permutatsiooni töötlemisel tekib kokku 7 stsenaariumi, millest kaks on edukad ning viis ebaedukad.

Sobiva puude klassi defineerimiseks läheb meil vaja kaht loomulikku tehet AND-OR-puudel. Olgu T_1 ja T_2 kaks AND-OR-puud, siis $T_1 \owedge T_2$ ja $T_1 \owvee T_2$ tähistavad vastavalt puid:



Need operatsioonid austavad lehtede järjekorda, st määravad ka lehtede permutatsiooni, kus kõigepealt on algses järjekorras puu T_1 lehed ja siis T_2 lehed. Paneme tähele, et tehted $\owedge$ ja $\owvee$ pole puudel ei kommutatiivsed ega assotsiatiivsed.

Järgmiseks laiendame tehted $\owedge$ ja $\owvee$ puude hulkadele. Olgu $\mathcal{T}_1$ ja $\mathcal{T}_2$ kaks AND-OR-puude hulka, siis

$$\mathcal{T}_1 \owedge \mathcal{T}_2 = \{T_1 \owedge T_2 : T_1 \in \mathcal{T}_1, T_2 \in \mathcal{T}_2\}$$

$$\mathcal{T}_1 \owvee \mathcal{T}_2 = \{T_1 \owvee T_2 : T_1 \in \mathcal{T}_1, T_2 \in \mathcal{T}_2\}.$$

Nüüd defineerime puude hulkade pere $\{\mathcal{T}_m\}_{m=0}^{\infty}$ järgmiselt:

$$\mathcal{T}_0 = \{\cdot\},$$

st üks ühe lehega puu ning

$$\mathcal{T}_{m+1} = \mathcal{T}_m \owedge \mathcal{T}_m \cup \mathcal{T}_m \owvee \mathcal{T}_m.$$

Sisuliselt on $\mathcal{T}_m$ kõigi täielike 2^m lehega AND-OR-puude hulk, kus lehtede permutatsioon ei sunni ühtki serva "risti minema". Lihtne on näha, et $\mid \mathcal{T}_m \mid = 2^{2^n-1}$.

Meie eesmärk on hinnata stsenaariumite keskmist arvu, mida Algoritm 1 klassi T_m puude analüüsil kasutab, ning näidata, et see on eksponentsiaalne klassi $\mathcal{T}_m$ puude lehtede arvu 2^m suhtes.

Tähistagu $t(T)$ ja $f(T)$ vastavalt puu T edukate ja ebaedukate stsenaariumite arvu, stsenaariumite arv kokku on siis $s(T) = t(T) + f(T)$. Edaspidistes

arvutustes läheb vaja järgmisi reegleid:

$$
\begin{aligned}
&t(\cdot) = f(\cdot) = 1\\
&t(T_1 \oslash T_2) = t(T_1) \cdot t(T_2)\\
&f(T_1 \oslash T_2) = f(T_1) + t(T_1) \cdot f(T_2)\\
&t(T_1 \otimes T_2) = t(T_1) + f(T_1) \cdot t(T_2)\\
&f(T_1 \otimes T_2) = f(T_1) \cdot f(T_2)
\end{aligned}
$$

Nende reeglite tõestused on sirgjoonelised. Näiteks puu $T_1 \otimes T_2$ õnnestunud stsenaariumid võib jagada kahte lõikumatusse alamhulka: kõigepealt need, kus T_1 õnnestub (ja T_2 pole vaja uurida) ning siis need, kus T_1 ebaõnnestub ja T_2 õnnestub. Paneme tähele, et lehtede järjekord on selles arutluses oluline.

Puude hulga $\mathcal{T}$ jaoks tähistame $s(\mathcal{T}) = \sum_{T \in \mathcal{T}}^{n} s(T)$.

Meie eesmärk on siis näidata, et $\frac{s(\mathcal{T}_m)}{|\mathcal{T}_m|}$ on eksponentsiaalne 2^m suhtes. Avaldame $s(\mathcal{T}_m)$:

$$
\begin{aligned}
s(\mathcal{T}_m) &= \sum_{T \in \mathcal{T}_m} s(\mathcal{T}) = \sum_{T_1, T_2 \in \mathcal{T}_{m-1}} [s(T_1 \oslash T_2) + s(T_1 \otimes T_2)]\\
&= \sum_{T_1, T_2 \in \mathcal{T}_{m-1}} [t(T_1 \oslash T_2) + f(T_1 \oslash T_2) + t(T_1 \otimes T_2) + f(T_1 \otimes T_2)]\\
&= \sum_{T_1, T_2 \in \mathcal{T}_{m-1}} [t(T_1) \cdot t(T_2) + f(T_1) + t(T_1) \cdot f(T_2) + t(T_1) +\\
&\quad + f(T_1) \cdot t(T_2) + f(T_1) \cdot f(T_2)]\\
&= \sum_{T_1, T_2 \in \mathcal{T}_{m-1}} [t(T_1) + f(T_1)] + \sum_{T_1, T_2 \in \mathcal{T}_{m-1}} [t(T_1) + f(T_1)] \cdot [t(T_2) + f(T_2)]\\
&= \sum_{T_1, T_2 \in \mathcal{T}_{m-1}} s(T_1) + \sum s(T_1) \cdot s(T_2)\\
&= \mid \mathcal{T}_{m-1} \mid \cdot s(\mathcal{T}_{m-1})^2 + s(\mathcal{T}_{m-1})^2.
\end{aligned}
$$

See rekursiivne seos võimaldab meil saada ilusa rekursiivse avaldise ka pere $\mathcal{T}_m$ stsenaariumite keskmise arvu kohta:

$$
\begin{aligned}
K_m &= \frac{s(\mathcal{T}_m)}{\mid \mathcal{T}_m \mid} = \frac{s(\mathcal{T}_m)}{2^{2^m - 1}} = \frac{2^{2^{m-1}} \cdot s(\mathcal{T}_{m-1})}{2^{2-1^m}} + \frac{2^{2^{m-1}} \cdot s(\mathcal{T}_{m-1})}{2^{2-1^m}}\\
&= \frac{2^{2^{m-1}} \cdot s(\mathcal{T}_{m-1})}{2^{2^{m-1}} \cdot 2^{2^{m-1}-1} \cdot 2} + \frac{s(\mathcal{T}_{m-1})^2}{(2^{2^{m-1}-1})^2 \cdot 2}\\
&= \frac{1}{2}\left[\frac{s(\mathcal{T}_{m-1})}{\mid \mathcal{T}_{m-1} \mid} + \left(\frac{s(\mathcal{T}_{m-1})}{\mid \mathcal{T}_{m-1} \mid}\right)^2\right] = \frac{1}{2}(K_{m-1} + K_{m-1}^2).
\end{aligned}
$$

Kuna $K_0 = \frac{s(\mathcal{T}_0)}{|\mathcal{T}_0|} = \frac{1+1}{1} = 2$, siis $K_1 = \frac{1}{2}(2 + 2^2) = 3$.

Edasises hinnangus kasutame ülaltuletatud rekursioonist järelduvat võrratust $K_m > \frac{1}{2} \cdot K_{m-1}^2$. Elementaarse induktsiooniga saame nüüd võrdusele $K_1 = 3$ tuginedes võrratuse $m \geq 1$ jaoks:

$$K_m \geq \frac{1}{2^{2^{m-1}-1}} \cdot 3^{2^{m-1}} > 1,5^{2^{m-1}},$$

see aga on eksponentsiaalne 2^m suhtes.

Niisiis näeme, et Algoritm 1 ei käitu halvasti mitte ainult halvimal juhul, vaid teatud mõttes ka keskmiselt.

Õnneks on suuruste $p_{\alpha,i}$ leidmiseks olemas efektiivne algoritm. Pöördudes tagasi Algoritmi 1 ja Teoreemi 2 juurde, näeme, et $p_{\alpha,i}$ on summa selliste rekursiooni harude tõenäosustest, kus leheni X_i jõudes on kõigi tippude väärtus teel X_i-st kuni juureni u. Tähistagu avaldis $(Y_0, Y_1, \ldots, Y_m)$ teed juurtipust Y_0 leheni $Y_m = X_i$.

Seega

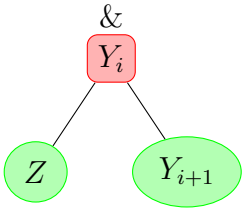
$$\begin{aligned}
p_{\alpha,i} &= \Pr[Y_0 = u \& Y_1 = u \& \ldots \& Y_m = u] \\
&= \Pr[Y_0 = u | Y_1 = u, \ldots, Y_m = u] \cdot \\
&\quad \cdot \Pr[Y_1 = u | Y_2 = u, \ldots, Y_m = u] \cdot \ldots \\
&\quad \cdot \Pr[Y_{m-1} = u | Y_m = u] \cdot \Pr[Y_m = u] \\
&= \Pr[Y_0 = u | Y_1 = u] \cdot \Pr[Y_1 = u | Y_2 = u] \cdot \ldots \\
&\quad \cdot \Pr[Y_{m-1} = u | Y_m = u] \cdot \Pr[Y_m = u] \qquad (1)
\end{aligned}$$

Viimane võrdus kehtib tänu sellele, et meil on tegu puuga.

Seega tuleb meil hinnata tõenäosusi, et teatud tippude väärtus on veel määramata. Selleks lisame puu igale tipule A kolm parameetrit: $A.t$, $A.f$ ja $A.u$, mis väljendavad tõenäosusi, et vastav tipp on õnnestunud väärtustada vastavalt tõeseks, vääraks või on veel väärtustamata. Arvutuste alguses on igas tipus $A.t = 0$, $A.f = 0$ ja $A.u = 1$, samuti kehtib igas tipus igal sammul $A.t + A.f + A.u = 1$.

Vaatleme nüüd avaldise (1) väärtustamist uues tipus $Y_m = X_i$, siis $\Pr[Y_m = u] = 1$. Ülejäänud tegurid korrutises (1) on kujul $\Pr[Y_m = u | Y_{i+1} = u]$.

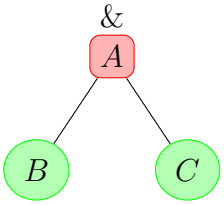
Kui Y_i on AND-tipp, siis puu fragment, mis koosneb tipust Y_i ja tema alluvatest, on järgmine:



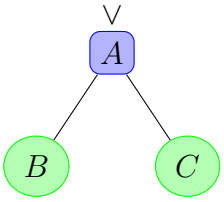
Me peame hindama tõenäosust, et Y_i on määramata eeldusel, et Y_{i+1} on määramata. See on nii siis, kui Z on kas tõene või määramata, seega tõenäosusega $Z.t + Z.u = 1 - Z.f$

Analoogiliselt saame, et kui Y_i on OR-tipp alluvatega Y_{i+1} ja Z, siis $\Pr[Y_i = u|Y = u] = 1 - Z.t$.

Nii saamegi avaldisest (1) leida efektiivselt $p_{\alpha,i}$ väärtuse, seejärel jääb vaid tippude $Y_i (i = 0, ..., m)$ parameetrite väärtused uuendada. See toimub lehest $X_i = Y_m$ juureni: kõigepealt omistame $Y_m.t = p_i$, $Y_m.f = 1 - p_i$, $Y_m.u = 0$ ja siis arvutame teiste tippude parameetrid sõltuvalt sellest, kas nad on AND-tipud või OR-tipud.



$$\begin{aligned} A.t &= B.t \cdot C.t \\ A.f &= B.f + C.f - B.f \cdot C.f \end{aligned} \tag{2}$$



$$\begin{aligned} A.t &= B.t + C.t - B.t \cdot C.t \\ A.f &= B.f \cdot C.f \end{aligned} \tag{3}$$

Parameetreid $A.u$ polegi tegelikult arvutustes kusagil vaja, seega need võib tegelikult ära jätta. Samuti paneme tähele, et arvutuste lõpul väljendab juurtipu parameeter $Y_0.t$ juurtipu õnnestumise tõenäosust, seega suurust p_α.

Paneme väljatöötatud algoritmi ka formaalselt kirja, kusjuures siinkohal piirdume ainult väärtuste $p_{\alpha,i}$ arvutamisega. Samuti paneme tähele, et piirdume vaid binaarse juhuga.

Algoritm 2 Tõenäosuse $p_{\alpha,i}$ arvutamine

Sisend : AND-OR-puu $\mathcal{F}$ lehtede hulgaga $\mathcal{X} = \{X_1, \ldots, X_n\}$, permutatsioon $\alpha \in S_n$

Väljund: Permutatsiooni tõenäosus $p_{\alpha,i}$

forall $Z \in \{X_1, \ldots, X_n\}$ **do** $Z.t := 0$; $Z.f := 0$;
for $i := 1$ **to** n **do**
    Leia tee $(Y_0, Y_1, \ldots, Y_m)$ juurest Y_0 leheni $Y_m = X_{\alpha(i)}$;
    $p_{\alpha(i)} = \prod_{j=1}^{m}(1 - Z_j.a)$;
    kus Z_j on tipu Y_{j-1} teine alluv peale Y_j tippu ning $a = \begin{cases} \mathsf{t} & \text{kui } Y_{j-1} \text{ on OR-tipp} \\ \mathsf{f} & \text{kui } Y_{j-1} \text{ on AND-tipp} \end{cases}$
    $X_{\alpha(i)}.t := p_{\alpha(i)}$;
    $X_{\alpha(i)}.f := 1 - p_{\alpha(i)}$;
    Uuenda tippude $Y_{m-1}, Y_{m-2}, \ldots, Y_0$ parameetrid vastavalt seostele (2) ja (3);
end
return $p_{\alpha,i}$

Kui puu $\mathcal{F}$ on balansseeritud, on $m = \log n$ ning algoritmi ajaliseks keerukuseks tuleb $O(n \log n)$, kui puu on balansseerimata, saame halvimal juhul $m = n$ ning algoritmi keerukuseks $O(n^2)$. Mõlemal juhul on Algoritm 2 tunduvalt efektiivsem kui Algoritm 1.

Nagu juba nägime, ei sõltu suurused p_α tegelikult permutatsioonist α, suurused $p_{\alpha,i}$ aga küll. Veel on lahtine küsimus, mil määral mõjutab neid valem $\mathcal{F}$. Nagu mäletame artiklist [JW08], mängis paralleelses mudelis täppisarvutuste puhul rolli mitte valemi $\mathcal{F}$ konkreetne kuju, vaid ainult tema käitumine Boole´i funktsioonina. Sellest tähelepanekust järeldus ka otseselt [JW08] arvutuste kooskõlalisus Mauw-Oostdijki mudelis [MO05].

Mauw ja Oostdijk defineerisid ründepuudearvutuste sisemise kooskõlalisuse puuteisenduste kaudu, viies aluseks oleva valemi sisuliselt disjunktiivsele normaalkujule ning nõudes, et arvutuste tulemus ei tohi sellest muutuda. Analoogiliselt artikliga [JW08] pole ka käesolevas aruandes toodud arvutusi võimalik esitada propageeruvate puuarvutustena. Lisaks kerkib veel probleem, et valemi viimisel disjunktiivsele normaalkujule tuleb muutujaid üldiselt kopeerida mitmesse eksemplari ning pole selge, mida hakata sel puhul peale lehtede permutatsiooniga α.

Sellegipoolest on võimalik tõestada [JW08] lausega 1 analoogiline tulemus. Tähistagu $T_1 \equiv T_2$ olukorda, kus kaks ründepuud on Boole´i funktsioonidena loogiliselt samaväärsed. Selleks, et algoritmid 1 ja 2 korrektselt

töötaksid, eeldame samuti, et igale muutujatest $X_1, \ldots, X_n$ vastab kummaski puus täpselt üks leht. Antud permutatsiooni $\alpha \in S_n$ puhul saame siis mõlema algoritmi abil arvutada samad ootetulud $\mathsf{Outcome}_\alpha(T_1)$ ja $\mathsf{Outcome}_\alpha(T_2)$.

Teoreem 5. *Kui* $T_1 \equiv T_2$ *ja* $\alpha \in S_n$*, siis* $\mathsf{Outcome}_\alpha(T_1) = \mathsf{Outcome}_\alpha(T_2)$.

Tõestus. Näitame, et $\mathsf{Outcome}_\alpha(T)$ on võimalik välja arvutada, teades ainult T poolt määratud tõeväärtustabelit ning permutatsiooni α. Sisuliselt piisab, kui näitame, et tõeväärtustabeli alusel saab taastada kõik stsenaariumid, mida Algoritm 1 vaatleb. Selleks konstrueerime uue Algoritmi 3, mis käitub Algoritmiga 1 analoogiliselt. Põhiraskus peitub selles, et Boole´i funktsiooni (erinevalt Boole´i valemist) ei saa niisama lihtsalt osalisel sisendil väärtustada. Sellegipoolest võime mõnele sisendile lubada väärtust `u`, kui me oleme kindlad, et funktsiooni väljund ei sõltu sellest, kas vastavaks sisendiks võtta `t` või `f`.

Algoritmis 3 võtame esituse lihtsustamiseks $\alpha = id$. See eeldus poleks üldsust kitsendav juba Algoritm 1 korral, sisuliselt tuleb selle jaoks ainult muutujad ümber nimetada. Algoritmi 3 sisendiks on seega ainult Boole´i funktsioon $\mathcal{F}$, mida hakkame väärtustama ainult täissisenditel. Esmane väljakutse on kujul `output_scenarios`$([u, u, \ldots, u], 1)$.

Algoritmide 1 ja 3 sarnasus on ilmne. Sisulised erinevused on järgmised:

1. Algoritmis 3 peame muutuja X_i ebaolulisuse üle otsustama veendudes, et tema väärtustamisest ei sõltu edasised väärtustused;

2. rekursiooni peatumise üle otsustame selle järgi, et sõltumata järgmiste muutujate väärtustest on $\mathcal{F}$ väärtus alati kas `t` või `f`.

Kuna vajalikud otsused saab teha, kasutades funktsiooni $\mathcal{F}$ ainult Boole´i funktsioonina, emuleerib Algoritm 3 täielikult Algoritm 1 käitumist. □

Siinkohal on sobiv läbi teha näide. Olgu Boole´i funktsioon $\mathcal{F}$ antud tõeväärtustabeliga 2.

Esimesel kahel rekursiivsel kutsel väärtustatakse $A_{11} :=$ `t` ja $A_{21} :=$ `t` , kolmandal kutsel näeme, et $\mathcal{F} =$ `t` sõltumata muutujate A_{12} ja A_{22} väärtustest, seega on esimene stsenaarium käes.

Järgmisel kutsel on meil väärtused $A_{11} =$ `t` ja $A_{21} =$ `f` . Nüüd näeme, et sõltumata sellest, mis on A_{12} väärtus, määrab $\mathcal{F}$ väärtuse A_{22}. Seega jätame A_{12} vahele. Nõnda edasi arutledes leiame stsenaariumid, mis on koos vastava ründepuuga toodud joonisel 3.

Tuleme selle jaotise lõpuks veelkord tagasi alguses formuleeritud üldise skeemi juurde, kus ründaja valib optimaalse ootetuluga ründe üle kõikide

Algoritm 3 Ründepuu permutatsiooni α stsenaariumite arvutamine.

Sisend : Boole funktsiooni $\mathcal{F}$ väärtustus A ja sügavuse loendur i
Väljund: Stsenaariumite list

1 **if**

$$\begin{aligned} \forall a_{i+1}, \ldots, a_n &\in \{\mathsf{t}, \mathsf{f}\} \\ \mathcal{F}([A[1], \ldots A[i-1], t, a_{i+1}, \ldots, a_n]) &= \mathcal{F}([A[1], \ldots A[i-1], f, a_{i+1}, \ldots, a_n]) \end{aligned}$$

then
2 `output_scenarios(`A, $i+1$`);`
3 **return**;
4 **end**
5 $A[i] := \mathsf{t}$;
6 **if**

$$\begin{aligned} \forall a_{i+1}, \ldots, a_n &\in \{\mathsf{t}, \mathsf{f}\} \\ \mathcal{F}([A[1], \ldots A[i-1], t, a_{i+1}, \ldots, a_n]) &= \mathsf{t} \end{aligned}$$

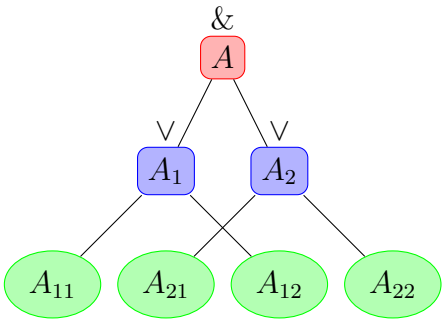
then
7 print A;
8 **else**
9 `output_scenarios(`A, $i+1$`);`
10 **end**
11 $A[i] := \mathsf{f}$;
12 **if**

$$\begin{aligned} \forall a_{i+1}, \ldots, a_n &\in \{\mathsf{t}, \mathsf{f}\} \\ \mathcal{F}([A[1], \ldots A[i-1], t, a_{i+1}, \ldots, a_n]) &= \mathsf{f} \end{aligned}$$

then
13 print A;
14 **else**
15 `output_scenarios(`A, $i+1$`);`
16 **end**

A_{11}	A_{21}	A_{12}	A_{22}	$\mathcal{F}$
1	1	1	1	1
1	1	1	0	1
1	1	0	1	1
1	1	0	0	1
1	0	1	1	1
1	0	1	0	0
1	0	0	1	1
1	0	0	0	0
0	1	1	1	1
0	1	1	0	1
0	1	0	1	0
0	1	0	0	0
0	0	1	1	1
0	0	1	0	0
0	0	1	0	0
0	0	0	0	0

Tabel 2: Funktsiooni $\mathcal{F}(A_{11}, A_{21}, A_{12}, A_{22})$ tõeväärtustabel.



A_{11}	A_{21}	A_{12}	A_{22}
+	+		
+	−		+
+	−		−
−	+	+	
−	+	−	
−	−	+	+
−	−	+	−
−	−	−	

Joonis 3: Ründepuu $A = (A_{11} \vee A_{12})\&(A_{21} \vee A_{22})$ koos permutatsiooni stsenaariumitega.

alamhulkade $S \subseteq \{X_1, \ldots, X_n\}$ ning vastavate permutatsioonide α. Kuigi ootetulu leidmise algoritmi õnnestus meil oluliselt optimeerida, on kogu optimaalse ründe valimise keerukus naiivse algoritmi korral $O(2^n \cdot n! \cdot n^2)$, mis on iga vähegi praktilise puu korral lootusetu. Seega muutub oluliseks küsimus arvutuste optimeerimisest. Siinkohal soovime püstitada kolm uurimisküsimust, mis näitavad võimalikke suundi edasiseks optimeerimiseks.

Küsimus 1. *Kas mingite kergesti kindlaks tehtavate tunnuste järgi on võiimalik mingid alamhulga S ja permutatsiooni α paarid ilma läbi proovimata kõrvale jätta kui kindlasti ebaoptimaalsed?*

Tuleks uurida [JW08] Teoreem 1 laadse tulemuse võimalikkust.

Küsimus 2. *Kas võib väita, et käesoleva jaotise mõttes optimaalse paari (S, α) hulk S on optimaalne ka [JW08] mõttes?*

Kui äratoodud hüpotees kehtiks, võimaldaks see keerukuse saada alla $O(2^n + n! \cdot n^2) = O(n! \cdot n^2)$ peale.

Küsimus 3. *Kas käesoleva jaotise arvutusi saab efektiivselt lahendada?*

Artikli [JW08] arvutuste lähendamisest räägime jaotises 5, tuleb uurida, kas neid lahendeid saab üle kanda ka siia jaotisesse.

4 Pooladaptiivne blokeeruv juht

Nagu eelpool juba mainitud, täiendab blokeeruv juht mitteblokeeruvat sellega, et võimaldab ründepuu lehtedel kanda märgist "blokeeruv", mille semantika on edasiste arvutuste peatamine, kui selle lehe väärtus saab f.

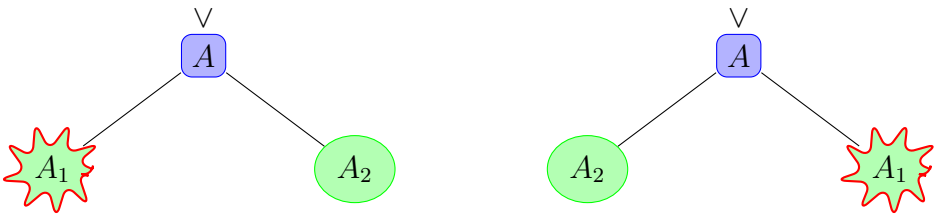
Formaalselt tuleb Algoritm 1 blokeeruvale juhule üle minekuks teha ainult väga väike muudatus, nimelt tuleb 13. rida ümber kirjutada kui

---

13 **if** $\mathcal{F}(A) = \mathsf{f}$ **or** $X_{\alpha(i)}$ *on blokeeruv* **then**

Algoritmi seisukohast lõikab tipu blokeerimine ära edasise rekursioonimineku false-harus. See pealtnäha väike muudatus toob endaga kaasa suure erinevuse algoritmiväljundi käitumises. Teoreemide 1 ja 2 analoogid kehtivad endiselt ja ma ei hakka siinkohal tõestusi ümber kirjutama.

Küll aga ei kehti enam Teoreem 3, st juurtipu õnnestumise tõenäosus sõltub nüüd kasutatavast permutatsioonist. Seda võib näha juba väga lihtsas kahe lehega puus, kus tipp A_1 on blokeeruv, joonisel 4.



Joonis 4: Vasakpoolse puu juure õnnestumise tõenäosus on p_1, parempoolse puu juure õnnestumise tõenäosus on $p_2 + (1 - p_2)p_1 = p_1 + p_2 - p_1p_2$.

Algoritmi tasemelt vaadeldes tekib probleem sellest, et seoses osade rekursiooni harude äralõikamisega ei saavuta me enam kõiki juurtipu realiseerivaid stsenaariume.

Kuna Teoreem 3 enam blokeeruval juhul ei kehti, satub löögi alla ka Teoreem 4, mis seda kasutab; kontranäidet peaks olema lihtne konstrueerida.

Küll aga kehtib blokeeruval juhul endiselt Teoreemi 5 analoog, tõestuses tuleb ainult Algoritmis 3 täiendada 12. rida kontrolliga, kas leht X_i oli blokeeruv. Niisiis on ka blokeeruval juhul võimalik ründega ootetulu leida, lähtudes vaid ründepuu aluseks olevast Boole´i funktsioonist, lehtede permutatsioonist ning infost selle kohta, millised lehed on blokeeruvad. Selle pealtnäha ilusal tulemusel on ka ebameeldivaid järeldusi, nagu järgnevas ilmneb.

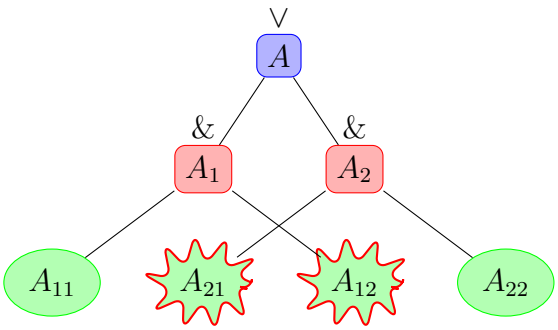
Kuigi Algoritm 1 tasemel oli blokeerumisega arvestamiseks vajalik muudatus väga väike, toob see endaga kaasa väljundi hoopis teistsuguse käitumise. Autoritel pole seni õnnestunud tuletada Algoritmi 2 analoogi suuruste $p_{\alpha,i}$ efektiivseks arvutamiseks ega isegi artiklis [JW08] toodud algoritmi analoogi juurtipu õnnestumise tõenäosuse p_α arvutamiseks. Käesolevas dokumenteerime mõned ebaõnnestunud katsetused.

Artikli [JW08] algoritm p_α arvutamiseks on väga lihtne: võtame puu igas lehes $X_i.t = p_i$, $X_i.f = 1 - p_i$ ja kasutame väärtuste propageerimiseks juurtipuni valemeid (2) ja (3). Loomulik idee selle lähenemise üldistamiseks on tuua puu tippudele juurde parameeter $X.b$, mis näitab blokeerumise tõenäosust. Siis saab lehtedes võtta

$$\begin{cases} X_i.t = p_i \\ X_i.f = 1 - p_i \\ X_i.b = 0 \end{cases} \quad \text{kui } X_i \text{ ei ole blokeeruv}$$

ja

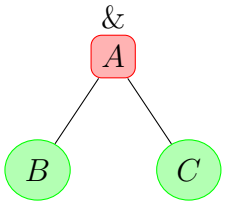
$$\begin{cases} X_i.t = p_i \\ X_i.f = 0 \\ X_i.b = 1 - p_i \end{cases} \quad \text{kui } X_i \text{ on blokeeruv.}$$



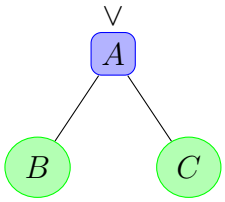
Joonis 5: Blokeeruva puu näide

Siin interpreteerime parameetrit $X_i.f$ kui tipu X_i ebaõnnestumist ilma blokeerumiseta.

Jällegi loomulikud üldistused valemitele (2) ja (3) on järgmised:



$$\begin{cases} A.t = B.t \cdot C.t \\ A.f = B.f + B.t \cdot C.f \\ A.b = B.b + B.t \cdot C.b \end{cases} \tag{4}$$



$$\begin{cases} A.t = B.t + B.f \cdot C.t \\ A.f = B.f \cdot C.f \\ A.b = B.b + B.f \cdot C.b \end{cases} \tag{5}$$

Paneme tähele, et mitteblokeeruval juhul kehtib $B.t + B.f = 1$, seega saaksime ülaltoodud AND-tipu $A.f$–reegli korral $A.f = B.f + B.t \cdot C.f = B.f + (1 - B.f) \cdot C.f = B.f + C.f - B.f \cdot C.f$, st täpselt valemites (2) ja (3) antu; analoogiliselt ka OR-tipu $A.t$-reegli jaoks.

Paraku on juba väikeste näidete pealt võimalik näha, et see algoritm ei tööta. Näiteks joonisel 5 toodud puu korral (kus tipud A_{21} ja A_{12} on blokeeruvad), on Algoritm 1 abil leitud juurtipu õnnestumise tõenäosus $p_{11}p_{21}p_{12} + (1 - p_{11})p_{21}p_{22}$, aga ülalkirjeldatud algoritm annab $p_{11}p_{12} + (1 - p_{11})p_{21}p_{22}$.

Erinevus ei tundu suur, kuid on siiski fundamentaalne. Suurim vahe valemite (2,3) ning valemite (4,5) seisneb selles, et arvutused (2,3) järgi on kommutatiivsed, (4,5) järgi aga mitte. Puu lehtede järjekord on permutatsiooniga

α ette antud, vahetippude järjekord aga mitte. seega tuleb mittekommutatiivsete propageeruvate puuarvutuste korral kuidagi vahetippude järjekorra üle otsustada. Esiteks pole selge, kuidas seda teha ja teiseks on olemas lihtsad näited, kus ükski vahetippude järjekord ei anna õiget tulemust (nt seesama viimane näide).

Igal juhul tundub, et kui efektiivne propageeruv algoritm p_α leidmiseks eksisteerib, peavad tippude arvutusreeglid olema kommutatiivsed, sest Teoreemi 5 analoogi põhjal blokeeruva juhu jaoks ei tohi arvutuste tulemus sõltuda konkreetsest puust, vaid ainult Boole´i funktsioonist. Samas pole kommuteeruvaid arvutusi lihtne ette kujutada, sest mingi lehe blokeerumine toob arvutustesse sisse fundamentaalse ebasümmeetria. Ainus võimalus seda ebasümmeetriat kommuteeruvate arvutuste juures arvestada, on muuta lehtede parameetreid. Lehtede parameetriteks saab võtta

$$\begin{cases} X_i.t = p_{\alpha,i} - p_i \\ X_i.f = p_{\alpha,i} - (1 - p_i) & \text{kui } X_i \text{ ei ole blokeeruv} \\ X_i.b = 0 \end{cases}$$

ja

$$\begin{cases} X_i.t = p_{\alpha,i} - p_i \\ X_i.f = 0 & \text{kui } X_i \text{ on blokeeruv.} \\ X_i.b = p_{\alpha,i} \cdot (1 - p_i) \end{cases}$$

See oleks täiesti võimalik, kui meil õnnestuks leida Algoritmi 2 efektiivne analoog, aga ka see pole veel õnnestunud. Proovisime nii defineeritud leheväärtuste korral vahetippudes arvutamiseks käsitleda väärtuseid teatava Boole´i algebra elementidena, aga see ei viinud soovitud tulemusteni. Lisaks tähendaks üleminek reaalarvudelt Boole´i algebra elementidele algoritmi mälukeerukuse tõusu algebra elementide esituse arvelt.

5 Ründepuude täppisarvutuste lähendamisest

Kuigi võrreldes artikliga [BLP$^+$06] annavad artiklis [JW08] toodud arvutused tunduvalt parema (ning isegi mingis mõttes optimaalse) tulemuse, on nad oluliselt aeglasemad. Seetõttu on väga huvitav küsimus nende arvutuste efektiivsest lähendamisest. Järgnevalt kirjeldame mõnesid ideid selles valdkonnas.

Juba artiklis [JW08] tegime tähelepaneku, et [BLP$^+$06] semantika üks põhipuudusi on asjaolu, et ta valib igas OR-tipus vaid ühe alluva. Niisiis võiks

tulemuse paranemist loota juba siis, kui meil õnnestuks OR-tippudes teha kiireid ja mingis mõttes mõistlikke otsuseid, mis põhimõtteliselt lubaks ka mitut alluvat. Kõige lihtsam oleks OR-tippude alluvaid valida juhuslikult. Seda strateegiat on küll raske mõistlikuks nimetada, kuid teda on lihtne programmeerida ja võrdlustestina on ta huvitav, näidates, kui palju "mõistlikumad" õieti need "mõistlikud" valikud on.

Paremat tulemust tõotab strateegia, kus me vaatleme iga OR-tippu koos tema vahetute alluvatega omaette väikese ründepuuna ja rakendame sellele [JW08] algoritmi (kasutades seejuures endiselt parameetri Gains globaalset väärtust, nagu artiklis [BLP$^+$06]). Kui puus on OR-tippudel maksimaalselt k alluvat ja puus on n lehte, on selle algoritmi keerukus $O(n \cdot 2^k)$, mis pole väga hull, kui k on mõistlikult väike.

Järgmine idee on teha lehtede kohta "kasulikkuse statistikat". Selleks eraldame kõigepealt lehtede hulgast need, mis kindlasti peavad igas juurrünnet realiseerivas komplektis olema (need on parajasti need, millede jaoks teel neist kuni juureni on ainult AND-tipud).

Iga ülejäänud lehe X_i jaoks koostame mingi fikseeritud arvu m juhuslikke ründekomplekte, kus tippu X_i ei ole. (Selleks võime tipu X_i puust välja jätta ning igas OR-tipus alluvate valimisel kulli ja kirja visata.). Siis arvutame nende ründekomplektide keskmise ootetulu üle m katse, paneme X_i kõigisse komplektidesse juurde ning arvutame uuesti keskmise ootetulu. Seejärel reastame lehed X_i kõigisse komplektidesse juurde nng arvutame uuesti keskmise ootetulu. Seejärel reastame lehed X_i selle järgi, millise lisamine keskmist ootetulu kõige rohkem kasvatas, ja lisame neid lehti parimast alates ründekomplekti seni, kuni saame juurrünnet realiseeriva komplekti. Selle algoritmi keerukus on $O(n^2 \cdot m)$. Parameetri m optimaalne väärtus selgub praktilisel testimisel, võib proovida näiteks väärtusi $m = 100$, $m = n$, $m = n^2$.

Kolmas lahendus, mis näib hetkel lootustandvaim, on geneetiline programmeerimine. Paneme tähele, et ründepuud võib vaadelda kui arvutusmasinat, mille programmiks on ründekomplekt. Juurrünnet realiseerivaid komplekte võime vaadelda kui süntaktiliselt korrektseid programme ning arvutatavat ootetulu kui *fitness*-funktsiooni.

Esimeseks generatsiooniks võime valida m juurrünnet realiseerivat komplekti (genereerides neid näiteks juhuslikult, nii nagu eelpool kirjeldatud). Järgmise generatsiooni moodustamiseks jagame kõik komplektid juhuslikult kaheks: $S_i = S_{i,1} \cup S_{i,2}$, kus $(i = 1, \ldots, m)$. Uueks generatsiooniks saavad komplektid

$$S = \left\{ S_{i,1} \cup S_{j,2} : i = 1, \ldots, m, j = 1, \ldots, m \right\},$$

millede seast valime kõigepealt süntaktiliselt korrektsed ja siis nende seast

omakorda m parima fitnessiga eksemplari. Kuna ilmselt $S_1, \ldots, S_m \in S$, siis on see alati võimalik (vaadeldes S-i vajadusel multihulgana).

Kuna võimalikke ründekomplekte on lõplik arv, siis see protsess kindlasti koondub, misjärel saab valida suurima ootehulga komplekti. Praktikas pole koondumist muidugi lihtne kindlaks määrata ja seetõttu tuleb peatuda mingi lihtsalt kindlaks tehtava tunnuse alusel — näiteks kui parimate komplektide hulk on juba viimased k generatsiooni sama olnud. Loomulikult pole meil siis garantiid, et me tõesti koondumiseni jõudsime, samuti ka mitte selle kohta, et me isegi antud instantsi koondumisel globaalse optimumi leidsime. Tulemuse parandamiseks võime kogu protsessi l korda käivitada. Tulemuse headus sõltub parameetrite k, l ja m valikust ja isegi siis on algoritmi keerukust raske hinnata. Kõigele sellele vaatamata usume, et geneetiline programmeerimine annab ründekomplekti headuse mõttes vaadeldud lähenemistest kõige parema tulemuse. Tõe kriteerium on siinkohal praktika.

Kõik kolm ülalvaadeldud meetodit leiavad mingi ründekomplekti ja annavad seega ründaja ootetulule alumise hinnangu. Nagu märgitud juba artiklis [JW08], oleks ülemised hinnangud tegelikult palju huvitavamad.

Veel üks lahendama küsimus on heade lähendite leidmine pooladaptiivse juhu jaoks. Ilmselt tuleb ülaltoodud algoritmide poolt leitud lahendid kuidagi permuteerida, aga selle tagajärjed võivad olla ootamatud; näiteks on siis väga raske öelda midagi geneetilise algoritmi koonduvuse kohta.

Viited

[BLP+06] Ahto Buldas, Peeter Laud, Jaan Priisalu, Märt Saarepera, and Jan Willemson. Rational Choice of Security Measures via Multi-Parameter Attack Trees. In *Critical Information Infrastructures Security. First International Workshop, CRITIS 2006*, volume 4347 of *LNCS*, pages 235–248. Springer, 2006.

[BM07] Ahto Buldas and Triinu Mägi. Practical security analysis of e-voting systems. In A. Miyaji, H. Kikuchi, and K. Rannenberg, editors, *Advances in Information and Computer Security, Second International Workshop on Security, IWSEC*, volume 4752 of *LNCS*, pages 320–335. Springer, 2007.

[JW07] Aivo Jürgenson and Jan Willemson. Processing multi-parameter attacktrees with estimated parameter values. In A. Miyaji, H. Kikuchi, and K. Rannenberg, editors, *Advances in Information and Computer Security, Second International Workshop on Security, IWSEC*, volume 4752 of *LNCS*, pages 308–319. Springer, 2007.

[JW08] Aivo Jürgenson and Jan Willemson. Computing exact outcomes of multi-parameter attack trees. In Robert Meersman and Zahir Tari, editors, *On the Move to Meaningful Internet Systems: OTM 2008*, volume 5332 of *LNCS*, pages 1036–1051. Springer, 2008.

[MO05] Sjouke Mauw and Martijn Oostdijk. Foundations of attack trees. In Dongho Won and Seungjoo Kim, editors, *International Conference on Information Security and Cryptology – ICISC 2005*, volume 3935 of *LNCS*, pages 186–198. Springer, 2005.

[Sch99] Bruce Schneier. Attack trees: Modeling security threats. *Dr. Dobb's Journal*, 24(12):21–29, December 1999.

[VGRH81] W.E. Vesely, F.F. Goldberg, N.H. Roberts, and D.F. Haasl. *Fault Tree Handbook*. US Government Printing Office, January 1981. Systems and Reliability Research, Office of Nuclear Regulatory Research, U.S. Nuclear Regulatory Commission.

[Wei91] J. D. Weiss. A system security engineering process. In *Proceedings of the 14th National Computer Security Conference*, pages 572–581, 1991.