Vizualizacija binarnih odločitvenih grafov

Robert Meolic

Fakulteta za elektrotehniko, računalništvo in informatiko
Koroška cesta 46, 2000 Maribor
E-pošta: robert.meolic@um.si

Binary Decision Diagram Visualisation

Visualisation of data structures and algorithms is not only a beneficial technology in teaching but also very handy and useful in researching new and difficult topics. BDD Scout is a software for visualising binary decision diagrams, a data structure which appeared in the late '80s and quickly became state-of-the-art representation of Boolean functions and combination sets. Binary decision diagrams are complex enough that studying their properties without the help of computer is hard and tiresome. The paper gives a short introduction to this interesting topic in computer science, a short description of the BDD Scout software as well as some screenshots, examples, and hints for its usage.

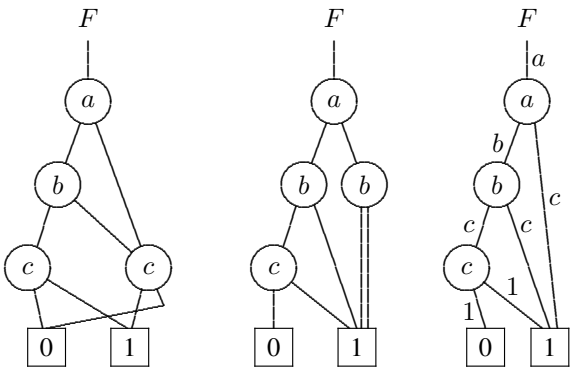
1 Uvod

Pri razvoju novih in pri optimizaciji obstoječih podatkovnih struktur ter računalniških algoritmov se pogosto srečamo s problemom vizualizacije. Pojem vizualizacija v tem primeru označuje interaktivno simulacijo podprto z grafiko (idealno tudi z animacijo), ki vsaj za majhne primere nazorno prikaže lastnosti preučevane rešitve. Najpogostejše so vizualizacije algoritmov za sortiranje, algoritmov nad grafi (barvanje, iskanje cikla, vpeto drevo, trgovski potnik, itd.), ter simulacije osnovnih podatkovnih struktur (sklad, povezan seznam, drevo, itd.).

Večina aplikacij za vizualizacijo algoritmov ima v prvi vrsti izobraževalni namen in služi programerjem kot pripomoček, da se lažje naučijo njim novih pojmov in pristopov. Še posebej zanimive so vizualizacije kompleksnejših ter še ne povsem raziskanih problemov, ki vsebujejo tudi raziskovalne prvine. Hitra in nazorna predstavitev rezultatov omogoča raziskovalcem, da opazijo še nepoznane oz. nedokazane lastnosti. Možnost prilagajanja obstoječih algoritmov in njihovih parametrov pa omogoča razvoj povsem novih različic.

Binarni odločitveni graf (BDD, Binary Decision Diagram) je podatkovna struktura za predstavitev množic, posredno pa tudi za vrsto drugih stvari, ki temeljijo na množicah. Zadnjih 25 let so BDD-ji state-of-the-art način predstavitve Boolovih funkcij (Boolova funkcija je množica mintermov) in osnova popularne metode za verifikacije sistemov imenovane preverjanje modelov (model checking). Prelomni članek R. E. Bryanta iz leta 1986 [1] je po podatkih Web of Science tretji najbolj citiran članek na področju Computer Science za leto 1986. Raziskave BDD-jev so do leta 2009 potekale predvsem na področju sinteze digitalnih vezij in formalne verifikacije sistemov. Leta 2009 je D. E. Knuth objavil samostojno poglavje o BDD-jih (134 strani) v svoji enciklopediji *The Art of Computer Programming* [2] in pokazal številne možnosti uporabe BDD-jev na področju kombinatoričnega preiskovanja, gradnje slovarjev ter teorije grafov, torej vsepovsod, kjer sta potrebna predstavitev in preiskovanje velikih množic. BDD-je je označil kot *"one of the only really fundamental data structures that came out in the last twenty-five years"*. Prvič se je z njimi srečal šele leta 1995, saj so bili članki objavljeni na konferencah in v revijah posvečenih strojni opremi, ki jih ni spremljal.

BDD-ji niso dokončno raziskani. Ker izvedba vključuje NP-polne probleme (npr. optimalni vrstni red spremenljivk v grafu) so aktualne hevristične metode in različne vrste evolucijskih algoritmov. Obetavno področje novejših raziskav je iskanje in preučevanje številnih podoblik te podatkovne strukture. Iz osnovne ideje binarnega odločitvenega grafa se je namreč razvilo veliko podvrst oz. tipov BDD-jev, ki se razlikujejo v podrobnostih same strukture in se zelo različno obnašajo v praksi [4]. Novi tipi BDD-jev so običajno vpeljani zato, ker omogočajo učinkovitejšo rešitev za določen problem kot že obstoječi tipi BDD-jev. Dokazana so le nekatera razmerja med različnimi tipi BDD-jev, ni pa niti znano, koliko različnih tipov BDD-jev sploh obstaja. Na sliki 1 so prikazani trije izmed njih.



Slika 1: Različni tipi BDD-jev za predstavitev Boolove funkcije $F(a,b,c) = a \cdot \overline{c} + b \cdot \overline{c} + \overline{a} \cdot \overline{b} \cdot c$. Od leve na desno so prikazani OBDD, ZBDD in TZBDD.

2 Binarni odločitveni graf

BDD je binarni in aciklični graf. V nadaljevanju povzamemo nekaj osnovne teorije, da omogočimo vpogled v naše delo tudi bralcem, ki tega podoročja računalništva ne poznajo. Za podrobnejše razumevanje je potrebno poseči po člankih in knjigah s tega področja [2, 4].

BDD ima en koren in dve končni vozlišči označeni z 0 in 1. Vsako notranje vozlišče v je označeno z oznako $var(v)$ in ima natanko dva naslednika, ki ju tukaj imenujemo naslednik $low(v)$ (do njega vodi leva povezava) in naslednik $high(v)$ (do njega vodi desna povezava). Vsak BDD ima tudi vrhnjo povezavo, ki vodi do korena. BDD je urejen, če se vsaka spremenljivka na vsaki poti pojavi največ enkrat in se spremenljivke na vseh poteh od korena do končnega vozlišča pojavljajo v istem vrstnem redu. Še ena skupna značilnost vseh tipov BDD-jev je ta, da graf ne sme vsebovati izomorfnih podgrafov.

Različi tipi BDD-jev k temu naboru osnovih pravil dodajo še dodatna. Velika skupina, ki jo označujemo kot MTBDD-ji (multi-terminal BDD-ji), namesto dveh končnih vozlišč 0 in 1 uporablja večjo množico končnih vozlišč označenih z naravnimi števili od 0 do 2^n. A tudi če se omejimo le na dve končni vozlišči, obstaja precej različnih tipov. Med seboj se razlikujejo po pravilu dekompozicije in po pravilu minimizacije, poleg tega pa tudi glede na dodatne oznake v grafu.

Pravilo dekompozicije v grafu določa, kako danemu BDD-ju priredimo Boolovo funkcijo (v splošnem gre za prireditev množice elementov). Za BDD, ki predstavlja Boolovo funkcijo, oznake v vozliščih predstavljajo konstante in spremenljivke. Boolove funkcije so prirejene povezavam med vozlišči ter vrhnji povezavi. Ta prireditev je vedno enolična in je podana lokalno tako, da za povezavo do vozlišča v določi Boolovo funkcijo glede na spremenljivko v vozlišču in glede na Boolovi funkciji, ki ju predstavljata povezavi do naslednikov. Do danes se vse raziskave osredotočajo le na dve pravili dekompozicije in sicer takšno, ki temelji na Shannonovi razširitvi Boolove funkcije $f = \overline{v} \cdot f|_{v=0} + v \cdot f|_{v=1}$, ter takšno, ki temelji na Reed-Mullerjevi razširitvi (imenovani tudi Daviova razširitev) $f = f|_{v=0} \oplus v \cdot (f|_{v=0} \oplus f|_{v=1})$.

Pravilo minimizacije določa, kaj pomeni odsotnost določenega vozlišča. Ponavadi jo sicer podamo obratno in sicer tako, da specificiramo vozlišča, ki jih lahko v grafu odstranimo pa bo BDD še vedno predstavljal isto Boolovo funkcijo. Na prvi pogled izgleda, da odsotnost določene spremenljivko lahko pomeni le to, da Boolova funkcija ni odvisna od te spremenljivke, a izkaže se da je to le ena od možnosti in da so uporabne tudi druge.

Boolovo funkcijo, ki jo predstavlja podan BDD določimo tako, da rekurzivno uporabimo pravilo dekompozicije, pri čemer pa moramo upoštevati tudi pravilo minimizacije in pomen morebitnih dodatnih oznak.

Tip BDD-jev, ki je bil prvi predstavljen in je še vedno najpogosteje uporabljen, danes označujemo kot OBDD. Za OBDD je pravilo dekompozije podano kot:

$$f(v) = \overline{var(v)} \cdot f(low(v)) + var(v) \cdot f(high(v))$$

Pravilo minimizacije pri OBDD je tisto najbolj običajno, torej da manjkajoča spremenljivka ne vpliva na Boolovo funkcijo. Med vsemi tipi BDD-jev se prav OBDD-ji najpogosteje uporabljajo za sintezo in analizo digitalnih vezij ter v programski opremi, ki izvaja verifikacijo sistemov z metodo preverjanja modelov. Za primer izračunajmo, katero Boolovo funkcijo predstavlja OBDD s slike 1 (notranja vozlišča od zgoraj navzdol in od leve na desno označimo z A, B, $C1$ in $C2$):

$$\begin{aligned} F = f(A) &= \overline{a} \cdot f(B) + a \cdot f(C2) = \\ &= \overline{a} \cdot (\overline{b} \cdot f(C1) + b \cdot f(C2)) + a \cdot (\overline{c} \cdot 1 + c \cdot 0) = \\ &= \overline{a} \cdot (\overline{b} \cdot (\overline{c} \cdot 0 + c \cdot 1) + b \cdot (\overline{c} \cdot 1 + c \cdot 0) + a \cdot \overline{c} = \\ &= \overline{a} \cdot (\overline{b} \cdot c + b \cdot \overline{c}) + a \cdot \overline{c} = a \cdot \overline{c} + b \cdot \overline{c} + \overline{a} \cdot \overline{b} \cdot c \end{aligned}$$

Drugi tip BDD-jev, ki tudi ima praktične aplikacije, je ZBDD. Za ZBDD je pravilo dekompozicije enako kot pri OBDD, pravilo minimizacije pa pravi, da manjkajoče spremenljivke v Boolovi funkciji nastopajo kot negativni literali. ZBDD-ji se dobro izkažejo pri kombinatoričnih problemih in tam, kjer imamo množice z zelo veliko domeno a majhnim številom elementov. Tukaj je izračun Boolove funkcije za ZBDD s slike 1 (notranja vozlišča od zgoraj navzdol in od leve na desno označimo z A, $B1$, $B2$ in C, negativne literale, ki se pojavijo zaradi minimizacije, pa označimo s krepko pisavo):

$$\begin{aligned} F = f(A) &= \overline{a} \cdot f(B1) + a \cdot f(B2) = \\ &= \overline{a} \cdot (\overline{b} \cdot f(C) + b \cdot \overline{\mathbf{c}} \cdot 1) + a \cdot (\overline{b} \cdot \overline{\mathbf{c}} \cdot 1 + b \cdot \overline{\mathbf{c}} \cdot 1) = \\ &= \overline{a} \cdot (\overline{b} \cdot (\overline{c} \cdot 0 + c \cdot 1) + b \cdot \overline{c}) + a \cdot \overline{b} \cdot \overline{c} + a \cdot b \cdot \overline{c} = \\ &= a \cdot \overline{c} + b \cdot \overline{c} + \overline{a} \cdot \overline{b} \cdot c \end{aligned}$$

Tretji tip BDD-jev, ki ga predstavljamo, je TZBDD. Gre za čisto nov tip BDD-jev, ki še ni bil opisan v nobeni publikaciji (avtor TZBDD-jev je avtor tega članka). Pri TZBDD je pravilo dekompozicije enako kot pri OBDD in ZBDD, pravilo minimizacije pa pravi, da za manjkajoče spremenljivke dodamo negativne literale, vendar le za tiste, ki so enake ali pa po urejenosti bolj spodaj v grafu kot spremenljivka, ki je podana kot oznaka na povezavi. Za primer izračunajmo, katero Boolovo funkcijo predstavlja TZBDD s slike 1 (notranja vozlišča od zgoraj navzdol označimo z A, B in C, negativne literale, ki se pojavijo zaradi minimizacije, pa označimo s krepko pisavo):

$$\begin{aligned} F = f(A) &= \overline{a} \cdot f(B) + a \cdot \overline{\mathbf{c}} \cdot 1 = \\ &= \overline{a} \cdot (\overline{b} \cdot f(C) + b \cdot \overline{\mathbf{c}} \cdot 1) + a \cdot \overline{c} = \\ &= \overline{a} \cdot (\overline{b} \cdot (\overline{c} \cdot 0 + c \cdot 1) + b \cdot \overline{c}) + a \cdot \overline{c} = \\ &= a \cdot \overline{c} + b \cdot \overline{c} + \overline{a} \cdot \overline{b} \cdot c \end{aligned}$$

Za konec te kratke predstavitve moramo še posebej poudariti, da so vsi trije BDD-ji na sliki 1 minimalni v smislu, da jim ni mogoče odstraniti nobenega vozlišča več tako, da bi ob danih pravilih dekompozicije in minimizacije graf predstavljal isto funkcijo. Za vse tri opisane tipe BDD-jev (pa tudi za številne druge) velja, da za vsako Boolovo funkcijo obstaja natanko eden in točno določen graf, ki predstavlja Boolovo funkcijo ter je minimalen glede na dani pravili dekompozicije in minimizacije. Torej so opisani tipi BDD-jev (in vsi drugi z enako lastnostjo) kanonična predstavitev Boolovih funkcij, kar zagotavlja zelo učinkovite operacije nad njimi, npr. preverjanje ekvivalence med dvema Boolovima funkcijama v konstantnem času.

3 BDD Scout

BDD Scout (`http://biddy.meolic.com/`) je prost program za vizualizacijo različnih tipov BDD-jev. Je del projekta Biddy. Na voljo je za operacijska sistema MS Windows in GNU/Linux. Uporabniški vmesnik programa je prikazan in na kratko razložen na sliki 2. Programska rešitev je sicer sestavljena iz treh delov:

- matematična knjižnica *Biddy* skrbi za tvorjenje in obdelavo BDD-jev v pomnilniku, vsebuje tudi razpoznavalnik formul in generator datotek, ki so potrebne za izris grafa,
- Tcl/Tk skripta *bddview* tvori in upravlja uporabniški vmesnik, skrbi tudi za izris BDD-ja, ki je podan v tekstovni datoteki z ustreznim formatom,
- glavni program imenovan *BDD Scout* je modularno zgrajen, osnovni modul povezuje *Biddy* in *bddview* v celoto, dodatni moduli dodajajo funkcionalnosti, v verziji 1.7 so na voljo moduli *BRA* (algoritmi za preurejanje vrstnega reda spremenljivk), *BDD Traces* (primerjalni testi) in *IFIP* (primerjalni testi).

Za vizualizacijo BDD-ja iz pomnilnika je dodatno potrebno orodje *graphviz/dot*. Za izvoz grafa v obliki slike PNG in datoteke PDF pa je potrebno orodje *ghostscript*.

Skripta *bddview* je neodvisna od knjižnice *Biddy* in sama ne omogoča tvorjenja ter obdelave BDD-jev. Nekaj preprostih operacij, ki so na voljo (npr. štetje vozlišč, izračun globine grafa), je izvedenih neposredno v jeziku Tcl brez uporabe bistveno učinkovitejše kode iz knjižnice *Biddy*. Ker lahko *BDD Scout* v obstoječo okolje dodaja svoje menuje, gumbe in druge elemente, je sicer mogoče uporabniku ponuditi poljubne funkcionalnosti, tudi takšne, ki potrebujejo zunanje knjižnice.

Format *bddview* je naslednji:

```
label <n> <name> <x> <y>
node <n> <name> <x> <y>
terminal <n> <name> <x> <y>
connect <n1> <n2> <type> [tag]
```

Posamezni elementi imajo naslednji pomen:

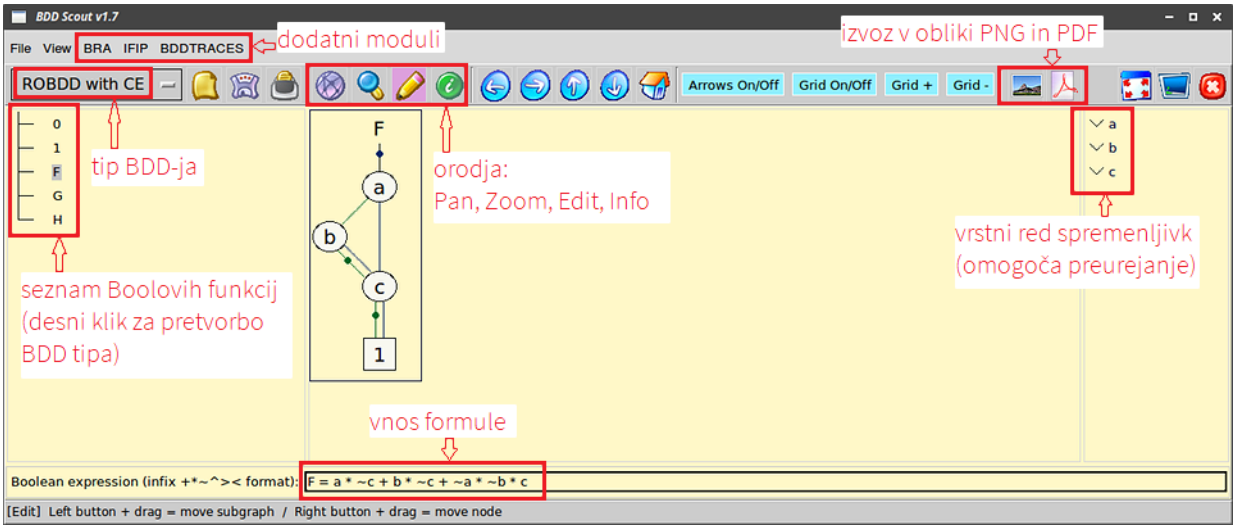
- `<n>`, `<n1>` in `<n2>` so številske oznake,
- `<name>` je niz znakov,
- `<x>` is `<y>` sta številski koordinati,
- `<type>` je ena od naslednjih oznak:

  `s`: enojna povezava,
  `si`: enojna povezava z oznako,
  `r`: povezava do naslednika $high$,
  `ri`: povezava do naslednika $high$ z oznako,
  `l`: povezava do naslednika low,
  `li`: povezava do naslednika low z oznako,
  `d`: dvojna povezava (leva ima oznako),
  `di`: dvojna povezava (desna ima oznako),
  `e`: dvojna povezava (brez oznak),
  `ei`: dvojna povezava (obe imata oznako),

- `[tag]` je niz znakov (neobvezno).

Primer datoteke v formatu `bddview` in rezultat programa BDD Scout sta prikazana na slikah 3 oz. 4.

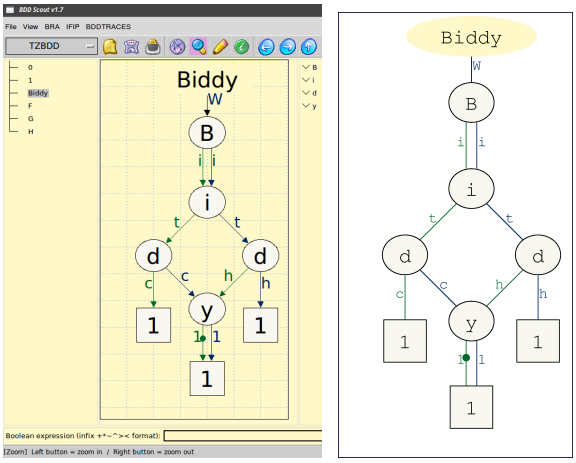
Funkcionalnosti programa BDD Scout v1.7 zadostujejo za izobraževalne namene, omogočajo pa tudi nekaj raziskovalnega dela. Boolovo funkcijo lahko podamo interaktivno v obliki formule (npr. F = a * ∼c + b * ∼c + ∼a * ∼b * c) ali pa preberemo eno oz. več formul iz datoteke (v tem primeru je trenutno podprt le prefiksni način zapisa). Sočasno lahko obstaja poljubno število Boolovih funkcij, a pripadajoči BDD je mogoče prikazati le za vsako posebej. Razporejanje vozlišč je samodejno, lahko pa uporabnik posamezna vozlišča ali pa celotne podgrafe poljubno premika. Če vključi mrežo v ozadju, se vozlišča samodejno pripnejo na vozlišča mreže in dobimo lepši graf. Gostoto mreže lahko prilagajamo. Vsak prikazan BDD lahko pretvorimo v kateregakoli od preostalih tipov BDD-jev. Spremenimo lahko vrstni red spremenljivk (prikazan graf se samodejno posodobi). Vgrajene so tudi druge funkcionalnosti, nekatere od njih so razvidne iz slik 2, 4 in 5, za podrobnejši opis žal ni več prostora.



Slika 2: Uporabniški vmesnik programa BDD Scout v1.7. Na posnetek zaslona so dopisani komentarji.

```
label 0 "Biddy" 100.0 10
node 1 "B" 100 60
node 2 "i" 100 125
node 3 "d" 50 175
terminal 4 1 50 240
node 5 "y" 100 225
terminal 6 1 100 290
node 7 "d" 150 175
terminal 8 1 150 240
connect 0 1 s W
connect 1 2 e i i
connect 2 3 l t
connect 2 7 r t
connect 3 4 l c
connect 3 5 r c
connect 5 6 d 1 1
connect 7 5 l h
connect 7 8 r h
```

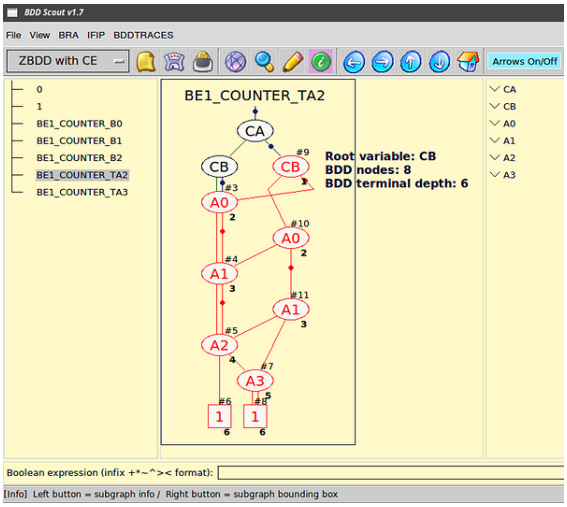
Slika 3: Datoteka v formatu `bddview`, ki jo tvori funkcija iz knjižnice *Biddy* s pomočjo zunanjega orodja *graphviz/dot*.



Slika 4: BDD, ki je prikazan po včitanju datoteke s slike 3 (levo) in izvoz grafa v obliki slike PNG (desno). Uporabnik je vključil prikazovanje puščic in mrežo v ozadju grafa na zaslonu.

4 Zaključek

Predstavljen delček teorije zadostuje, da lahko bralec za graf, ki uporablja enega od opisanih tipov BDD-jev, sam izračuna pripadajočo Boolovo funkcijo. A podan je le eden od možnih načinov. Prav tako v članku ni niti besede o tem, kako za dano Boolovo funkcijo tvorimo ustrezni BDD, niti kako operacije nad Boolovimi funkcijami učinkovito izvajamo neposredno nad BDD-ji. Za vse te in druge podrobnosti v zvezi z BDD-ji bo moral radovedni bralec poseči po dostopni literaturi. Prav gotovo pa mu pri razumevanju in preučevanju te zanimive podatkovne strukture lahko zelo pomaga program BDD Scout z nazornim prikazom BDD-ja za poljubno Boolovo funkcijo. Omogočena je tudi študija vpliva tipa BDD-ja (v trenutni verziji so sicer podprti le trije) in študija vpliva vrstnega reda spremenljivk na velikost grafa. V prihodnje načrtujemo, da bo podprtih še mnogo več tipov BDD-jev, ter da bo vpliv vrstnega reda spremenljivk še bolj nazorno prikazan s pomočjo animacije in podrobnih statistik.



Slika 5: BDD Scout zna prikazati, katera vozlišča spadajo v izbran podgraf. Na voljo je tudi nekaj osnovne statistike.

BDD Scout nikakor ni dokončan projekt. Da bi postal razširjen in uspešen, je potrebno izboljšati njegovo učinkovitost in robustnost, priložnost pa vidimo tudi v tem, da ga naredimo čimbolj prenosljivega (podpora za različne prevajalnike in razvojna okolja), da ponujamo uporabniku prijazne namestitvene datoteke (namestitveni paket za MS Windows in Mac OS X, paketi deb in rpm za GNU/Linux) ter da je izvorna koda prosta ter lepo urejena (uporabljamo doxygen, skladišče imamo na GitHub in na GNU Savannah). Predvsem pri praktičnem delu nam lahko veliko pomagajo študenti.

Za konec pa velja omeniti še to, da programska oprema v modernem svetu ni več omejena na tradicionalne programe, ki jih poganjamo na namiznem računalniku, pravzaprav je takšnih vedno manj. V ospredje prihajajo spletne aplikacije (izvajajo se v spletnem brskalniku) in mobilne aplikacije oz. app-i (izvajajo se na pametnih telefonih in tablicah). Vizualizacija algoritmov in podatkovnih struktur na novih platformah je trenutno zelo aktualna tema (glej npr. `http://algo-visualizer.jasonpark.me/` in `https://visualgo.net/` ter [3]). Vizualizacijo BDD-jev na spletnih in mobilnih platformah je zato še eden od pomembnih ciljev.

Literatura

[1] R. E. Bryant: "Graph-Based Algorithms for Boolean Function Manipulation", IEEE Transactions on Computers, C-35(8), str. 677–691, 1986.

[2] D. E. Knuth: "The Art of Computer Programming, Volume 4, Fascicle 1: Bitwise Tricks & Techniques; Binary Decision Diagrams", Addison-Wesley Professional, 2009.

[3] R. Meolic, T. Dogša: "A C++ App for Demonstration of Sorting Algorithms on Mobile Platforms", International Journal of Interactive Mobile Technologies, 8(1), str. 40–45, 2014.

[4] S. Minato: "Techniques of BDD/ZDD: Brief History and Recent Activity", IEICE Trans. on Information and Systems, E96D(7), str. 1419–1429, 2013.