Predstavitev slik z binarnimi odločitvenimi grafi

Robert Meolic, Zmago Brezočnik
Fakulteta za elektrotehniko, računalništvo in informatiko
Univerza v Mariboru
Smetanova 17, 62000 Maribor, Slovenija
meolic@uni-mb.si, brezocnik@uni-mb.si

Representing Images with Binary Decision Diagrams

This paper suggests binary decision diagram (BDD) as a useful data structure for representing images and image sequences. First, we give a short description of BDDs. Next, we show how they can be used to represent images. We perform some tests with bintrees and two common types of binary decision diagrams, ROBDDs and 0-sup-BDDs. Results show that BDDs are very efficient.

1. Uvod

Pri računalniški grafiki potrebujemo predstavitev slike v pomnilniku. Predstavitev mora omogočati učinkovito izvajanje operacij, kot so geometrijsko iskanje, obrezovanje, raztresanje, geometrijske transformacije, itd. Uporabljajo se različne podatkovne strukture: bitno polje, raster, verižna koda, štiriška in dvojiška drevesa [7].

Binarni odločitveni grafi („Binary Decision Diagrams "—BDD) so podatkovna struktura za predstavitev logičnih funkcij. Pri reševanju problemov s področja matematične logike, umetne inteligence, kombinatorike in digitalnega načrtovnja sistemov so v kratkem času nadomestili vse druge podatkovne strukture za predstavitev logičnih funkcij. Obstaja več vrst BDD. V naših testih smo uporabljali *minimalne urejene binarne odločitvene grafe („Reduced Ordered Binary Decision Diagrams"—ROBDD)* [1] in *binarne odločitvene grafe s potlačenimi ničlami („Zero-Suppressed Binary Decison Diagrams"—0-sup-BDD)* [6], ki so se na prej naštetih področjih zelo uveljavili.

V drugem razdelku podamo kratek opis binarnih odločitvenih grafov. V tretjem razdelku opišemo, kako z njimi predstavimo slike. V četrtem razdelku podamo rezultate testov, s katerimi smo primerjali učinkovitost dvojiškega drevesa, ROBDD in 0-sup-BDD pri predstavitvi črnobelih slik in zaporedja slik. Zaključimo s pregledom opravljenega dela in s predlogi za nadaljnje raziskave.

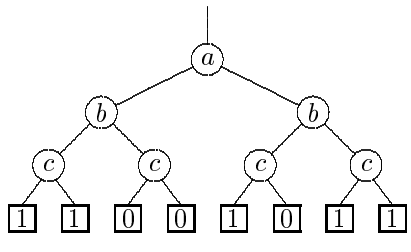
2. Binarni odločitveni graf

BDD je usmerjeni, aciklični graf, ki ima natanko en koren in je podan z množico vozlišč ter množico povezav. Vsebuje dve vrsti vozlišč. Vozlišča v listih grafa so *končna vozlišča*. Označena so z '0' ali '1' in nimajo naslednikov. Vsako nekončno vozlišče je označeno s spremenljivko in ima natanko dva naslednika. Povezavi do naslednikov označimo z 'then' in 'else'. Naslednika, ki sta tudi BDD, imenujemo *'else'* in *'then' naslednik* vozlišča. Spremenljivka v korenu grafa se imenuje *vrhnja spremenljivka*.

BDD lahko zapišemo rekurzivno kot trojko $\boldsymbol{F} = (v, \boldsymbol{E}, \boldsymbol{T})$, kjer je v vrhnja spremenljivka grafa, $\boldsymbol{E}$ in $\boldsymbol{T}$ pa 'else' in 'then' naslednik vozlišča v.

BDD je *prost*, če na vsaki poti od korena do končnega vozlišča vsaka spremenljivka nastopa največ v enem vozlišču. BDD je *poln*, če se na vsaki poti pojavijo vse spremenljivke. BDD je *urejen*, če je prost in če je vrstni red pojavljanja spremenljivk na vseh poteh enak. Pri urejenih BDD („Ordered BDD"—OBDD) je vrstni red spremenljivk določen z relacijo $<$, s katero uredimo spremenljivke od najmanjše (v korenu grafa) do največje.

Primer polnega OBDD [1] ($a < b < c$) je na sliki 1.



Slika 1: Poln OBDD

Vsak BDD predstavlja neko logično funkcijo. Dva BDD, ki predstavljata enako logično funkcijo, sta ekvivalentna. Preslikavo med grafom in logično funkcijo imenujemo razčlenitveno pravilo. Obstaja več različnih razčlenitvenih pravil [5]. BDD temeljijo na Shannonovem razčlenitvenem pravilu.

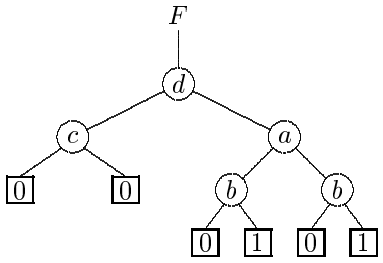
[1] Na slikah je povezava 'else' na levi, povezava 'then' pa na desni strani.

2.1 Shannonovo razčlenitveno pravilo

Imejmo BDD $\boldsymbol{F} = (v, \boldsymbol{E}, \boldsymbol{T})$. Po Shannonovem razčlenitvenem pravilu se logična funkcija, ki jo ta BDD predstavlja, izračuna rekurzivno po formuli: $F = \overline{v} \cdot E + v \cdot T$, kjer je v vrhnja spremenljivka (v korenu grafa), E in T pa logični funkciji, ki ju predstavljata 'else' in 'then' naslednika vozlišča v. Zapis poenostavimo, če uporabimo tromestni operator ITE [3, 2]:

$$F = \overline{v} \cdot E + v \cdot T = \mathsf{ITE}(v, T, E)$$

Kot primer izračunajmo, katero logično funkcijo predstavlja po Shannonovem razčlenitvenem pravilu odločitveni graf na sliki 2 ($d < c < a < b$): $F = \mathsf{ITE}(d, \mathsf{ITE}(a, \mathsf{ITE}(b, 1, 0), \mathsf{ITE}(b, 1, 0)), \mathsf{ITE}(c, 0, 0)) = \mathsf{ITE}(d, \mathsf{ITE}(a, b, b), \mathsf{ITE}(c, 0, 0)) = \mathsf{ITE}(d, b, 0) = b \cdot d$



Slika 2: Urejeni BDD za $F = b \cdot d$

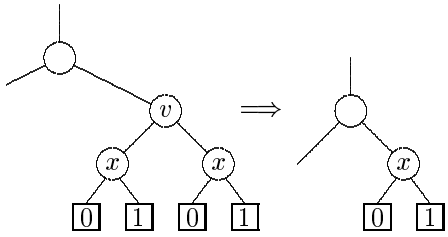
2.2 Minimizacija binarnih odločitvenih grafov

Minimizacija BDD je postopek, pri katerem iz danega grafa odstranimo vsa redundantna vozlišča. Za minimizacijo uporabljamo naslednja pravila:

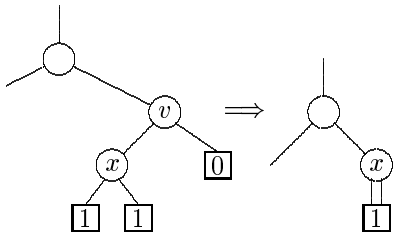
1. vse ekvivalentne podgrafe predstavimo le enkrat,
2. odstranimo vsa vozlišča, katerih 'else' in 'then' naslednika sta enaka (slika 3),
3. odstranimo vsa vozlišča, katerih 'then' naslednik je končno vozlišče '0' (slika 4).

Imejmo poln urejen BDD. Če ga minimiziramo po pravilih 1 in 2, dobimo ROBDD, če pa uporabimo pravili 1 in 3, dobimo 0-sup-BDD. ROBDD in 0-sup-BDD sta kanonični predstavitvi logične funkcije. Tako v ROBDD kot tudi v 0-sup-BDD dodatno zmanjšamo število vozlišč z vpeljavo označenih povezav [2, 3, 5].

Učinkoviti algoritmi za gradnjo BDD in operacije nad njimi delujejo tako, da se redundantna vozlišča nikoli ne tvorijo, zato naknadna minimizacija ni potrebna [2, 3].



Slika 3: Minimizacija pri ROBDD



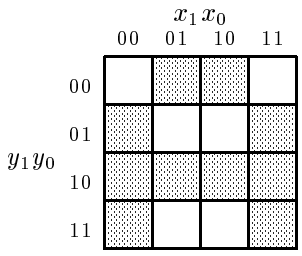
Slika 4: Minimizacija pri 0-sup-BDD

3. Predstavitev slike z dvojiškim drevesom in z BDD

Sliko predstavimo z dvojiškim drevesom tako, da jo delimo na polovico, dokler ne pridemo do delov, ki so v celoti iste barve. Delimo izmenično vodoravno in navpično. Primer dvojiškega drevesa za preprost črnobel vzorec s slike 5 je prikazan na sliki 6. V prvem koraku smo delili navpično.

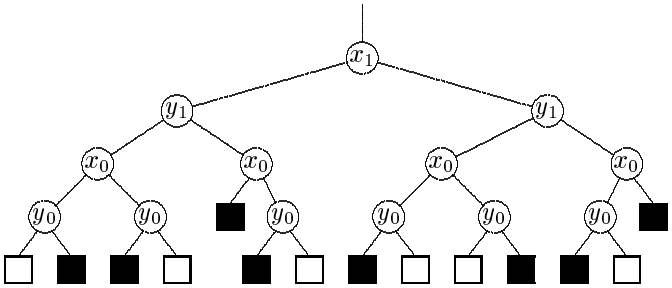
Z BDD predstavimo sliko tako, da jo zapišemo kot logično funkcijo. Za sliko velikosti *širina*×*višina* po širini vpeljemo spremenljivke $x_0, x_1, \ldots, x_{n-1}$, po višini pa spremenljivke $y_0, y_1, \ldots, y_{m-1}$, kjer je $n = \lceil \log_2 \textit{širina} \rceil$ in $m = \lceil \log_2 \textit{višina} \rceil$. Vsaki piki pripada en minterm (slika 5). V logično funkcijo vključimo minterme, ki pripadajo črnim [2] pikam. Za slike, pri katerih obe dimenziji nista potenci števila 2, pikam izven slike priredimo barvo ozadja.

ROBDD in 0-sup-BDD, s katerima predstavimo preprost vzorec s slike 5, sta prikazana na sliki 7.

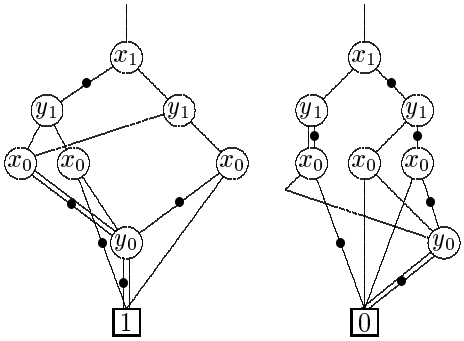


Slika 5: Preprost črnobel vzorec

[2] Dogovorimo se lahko tudi, da namesto črnih predstavimo bele pike.



Slika 6: Dvojiško drevo za vzorec s slike 5



a) ROBDD b) 0-sup-BDD

Slika 7: a) ROBDD in b) 0-sup-BDD za vzorec s slike 5

4. Rezultati testov

Programe za delo z BDD smo napisali v programskem jeziku C, meritve pa smo izvajali na delovni postaji HP 712/80 z naslednjo strojno in programsko opremo: 32 MB fizičnega pomnilnika, največ 75 MB navideznega pomnilnika, operacijski sistem HP-UX 9.05 in prevajalnik gcc 2.6.3.

4.1 Črnobele slike

V tem testu smo izmerili prostorsko učinkovitost predstavitve črnobelih slik (brez vmesnih sivin) z ROBDD, 0-sup-BDD in binarnim drevesom. Za test smo uporabili slike *EVROPA* (764×949), *MEDVED* (560×880), *SKAVT* (1232×1520), *SRCE* (255×255) in *UNIVERZA* (590×280) (slika 9).

Spremenljivke smo vpeljali analogno kot za preprost vzorec s slike 5. Uredili smo jih po naslednjem vrstnem redu:

$$x_{n-1} < y_{m-1} < x_{n-2} < y_{m-2} \ \cdots \ .$$

Dobljeni rezultati (tabela 1) kažejo, da potrebujemo za predstavitev slike z 0-sup-BDD nekoliko manj vozlišč kot za predstavitev z ROBDD. Dvojiško drevo zahteva v vseh primerih mnogo več vozlišč.

Tabela 1: Število vozlišč pri predstavitvi slik

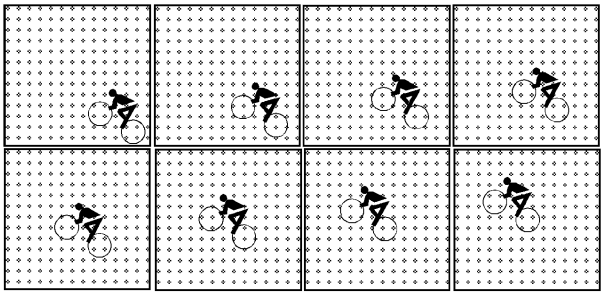
slika	ROBDD	0-sup-BDD	2-drevo
EVROPA	8502	8189	81203
MEDVED	10737	10556	131961
SKAVT	21296	20654	284377
SRCE	784	770	5197
UNIVERZA	8314	8106	101513

Če upoštevamo, da eno vozlišče BDD v naši izvedbi zasede 36 zlogov pomnilnika, vozlišče dvojiškega drevesa pa le 12 zlogov, lahko ugotovimo, da potrebujemo pri predstavitvi slike z BDD še zmeraj približno trikrat manj pomnilnika kot pri predstavitvi z dvojiškim drevesom.

4.2 Zaporedje slik

Za sliki, ki se razlikujeta le v nekaterih manjših delih, se izkaže, da imata pripadajoča BDD mnogo ekvivalentnih podgrafov. Ker se pri BDD ekvivalentni podgrafi ne ponavljajo, lahko z njimi zelo učinkovito predstavimo obe sliki naenkrat. Zaradi te lastnosti so BDD posebej primerni za predstavitev zaporedja slik, ki tvori animacijo [8].

Na sliki 8 vidimo prvih 8 slik preproste animacije, pri kateri objekt drsi čez ozadje. Celotna animacija je razdeljena na 14 slik velikosti 324×313.

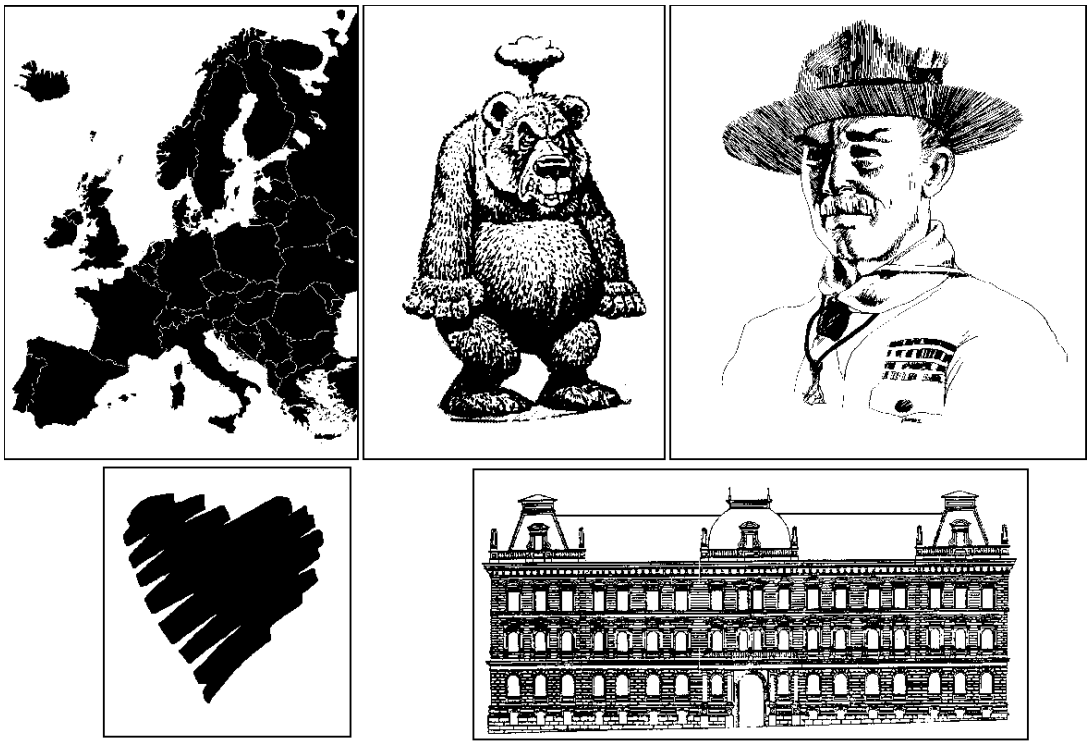


Slika 8: Preprosta animacija

Če zgradimo BDD za vsako sliko posebej, potrebujemo pri ROBDD skupno 10091 vozlišč, pri 0-sup-BDD pa 9157 vozlišč. Če pa posamezne BDD kombiniramo tako, da se ekvivalentni podgrafi predstavijo le enkrat, potrebujemo pri ROBDD le 5063 vozlišč (samo 50 %), pri 0-sup-BDD pa le 4726 vozlišč (samo 52 %). Ti rezultati potrjujejo učinkovitost BDD pri predstavitvi zaporedja slik.

5. Zaključek

Prikazali smo, kako lahko z BDD kompaktno predstavimo slike in zaporedja slik. Še posebej učinkovito



Slika 9: Primeri črnobelih slik

lahko z njim predstavimo ponavljajoče se vzorce v sliki. Dober primer je slika šahovnice velikosti $n \times n$, ki za vsak n zahteva v ROBDD le tri vozlišča, velikost dvojiškega drevesa pa z večanjem n hitro raste.

Nekoliko slabše se odrežejo BDD pri shranjevanju slike na datoteko. V [8] je navedeno, da so datoteke pri BDD za približno 25% daljše kot pri dvojiškem drevesu.

ROBDD in 0-sup-BDD, ki smo ju uporabili v testnih primerih, imata le dve različni končni vozlišči (0 in 1), zato lahko z njima predstavimo le črnobele slike brez vmesnih sivin. Obstajajo pa druge vrste odločitvenih grafov, ki imajo več različnih končnih vozlišč (cela števila). [4]. Z njimi bi bilo mogoče predstaviti črnobele slike z več odtenki in tudi barvne slike.

V prikazanih testih nismo primerjali časovne zahtevnosti BDD in dvojiških dreves. Glede na mnogo manjše število potrebnih vozlišč pri BDD upamo, da se operacije nad slikami, predstavljenimi z BDD, izvajajo vsaj tako hitro, če ne še hitreje, kot pri predstavitvi z dvojiškim drevesom.

Rezultati testov kažejo, da so BDD učinkovitejša podatkovna struktura za predstavitev slike v pomnilniku od dvojiškega drevesa. V prihodnosti pa se bo pokazalo, ali se bodo na področju računalniške grafike tudi res uporabljali.

Literatura

[1] K. S. Brace, R. L. Rudell, R. E. Bryant. Efficient Implementation of a BDD Package. *27. ACM/IEEE Design Automation Conference*, str. 40–45, 1990.

[2] A. Časar, R. Meolic. Representation of Boolean Functions with ROBDDs. *Članek sprejet za objavo v 1993/94 IEEE Student Papers Book.*

[3] A. Časar, R. Meolic, Z. Brezočnik, B. Horvat. Predstavitev logičnih funkcij z minimalnimi urejenimi binarnimi odločitvenimi grafi. *Elektrotehniški vestnik*, 59(5):299–307, december 1992.

[4] E. Clarke, K. L. McMillan, X. Zhao, M. Fujita, J. C.-Y. Yang. Spectral Transforms for Large Boolean Functions with Applications to Technology Mapping. *30th ACM/IEEE Design Automation Conference*, str. 54–60, 1993.

[5] R. Meolic. Uporaba urejenih odločitvenih grafov pri računalniški obdelavi logičnih funkcij. Diplomsko delo, FERI Maribor, junij 1995.

[6] S. Minato. Zero-Suppressed BDDs for Set Manipulation in Combinatorial Problems. *30th ACM/IEEE Design Automation Conference*, str. 272–277, 1993.

[7] H. Samet. *Applications of Spatial Data Structures.* Addison-Wesley Publishing Company, 1989.

[8] M. Starkey, R. Bryant. Using Ordered Binary-Decision Diagrams for Compressing Images and Image Sequences. Technical Report CMU-CS-95-105.