VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

# Bayesovské evoluční algoritmy

## s aplikacemi v úlohách dekompozice a alokace

Habilitační práce

2002

Ing. Josef Schwarz, CSc.

# 1 Předmět habilitační práce

Předložená habilitační práce se zabývá návrhem, analýzou a využitím Bayesovských evolučních algoritmů pro řešení složitých, vesměs NP-úplných optimalizačních kombinatorických problémů zejména z oblasti dekompozice a alokace grafových struktur, které lze v kontextu fyzického návrhu číslicových obvodů interpretovat jako členění a rozmísťování číslicových obvodů. Jde o pokročilé evoluční algoritmy založené na pravděpodobnostních modelech, které odstraňují nevýhody klasických evolučních algoritmů spojené s volbou genetických operátorů a nastavováním četných řídicích parametrů. Souhrnným označením pro tuto třídu algoritmů je zkratka EDA (Estimation Distribution Algorithm) - volně přeloženo algoritmy založené na odhadu pravděpodobnostního rozložení slibných řešení. [Larra99], [Muel98], [Pel99c]. V roce 1999 byl poprvé publikován sofistikovaný EDA algoritmus BOA (Bayesian optimization algorithm) [Pel99a], [Pel99b], který využívá pravděpodobnostní model reprezentovaný Bayesovskou sítí. Z uvedených publikací vyplývá, že tento algoritmus byl testován jen na umělých úlohách s binárním zakódováním a tzv. klamných úlohách.

V této práci je rozpracována a zobecněna idea Bayesovského evolučního optimalizačního algoritmu ve čtyřech oblastech:

- Kombinatorické optimalizační úlohy
- Využití dodatečných znalostí o řešeném problému
- Multikriteriální optimalizace
- Obecnější využití binárních rozhodovacích grafů (BDD).

Výběr aplikací, převážně z oblasti fyzického návrhu číslicových obvodů, vyplývá z odborného a pedagogického zaměření autora, který se dlouhodobě zabývá návrhem číslicových obvodů, mikropočítačových podsystémů a návrhem evolučních algoritmů.

Důraz kladený na řešení úloh dekompozice a alokace vyplývá jednak z obtížnosti řešení těchto úloh pro většinu používaných heuristik (jde vesměs o NP-úplné úlohy vhodné pro testování), ale také proto, že obě úlohy jsou dostatečně obecné a pokrývají širší škálu úloh včetně syntézy a testování číslicových obvodů.

Součástí habilitační práce jsou také experimentální výsledky získané vybranými heuristickými metodami a klasickými genetickými algoritmy, které jsou použity pro komparaci s EDA algoritmy.

# 2 Obsah a rozsah habilitační práce

Obsahem habilitační práce jsou výsledky výzkumné práce autora v oblasti využití pokročilých evolučních algoritmů EDA (Estimation Distribution Algorithm) a částečně heuristických metod a klasických genetických algoritmů při řešení zejména dekompozičních a alokačních problémů. Habilitační práce je podána formou souboru devíti článků, publikovaných v mezinárodních časopisech a mezinárodních konferencích s náročným výběrovým řízením. Text práce je strukturován do deseti kapitol. Třetí kapitola je úvodem do problematiky kombinatorických optimalizačních úloh a jejich řešení pomocí heuristických metod, klasických a pokročilých evolučních algoritmů. Ve čtvrté kapitole je průvodní komentář k devíti vybraným článkům A-1 až A-9, jejichž xerokopie jsou v příloze A. V páté kapitole je uveden přehled výzkumných úkolů na kterých se autor práce podílel. Šestá kapitola je věnována souhrnnému hodnocení dosažených výsledků, v sedmé kapitole je seznam vybraných prací, na které se autor odkazuje zejména v úvodní třetí kapitole. V osmé kapitole je seznam prací tvořících předloženou habilitační práci. V deváté kapitole je seznam dalších prací autora souvisejících s řešenou problematikou. Desátou kapitolu tvoří příloha A obsahující xerokopie předložených článků A-1 až A-9.

Pro snadnější orientaci je uveden přehled kapitol, které korespondují s články v příloze A.

| | | | |
|---|---|---|---|
| 4.1.1 | A_1 | 4.2.4 | A_6 |
| 4.1.2 | A_2 | 4.2.5 | A_7 |
| 4.2.1 | A_3 | 4.2.6 | A_8 |
| 4.2.2 | A_4 | 4.2.7 | A_9 |
| 4.2.3 | A_5 | | |

# 3 Metody řešení kombinatorických úloh

Lze definovat dvě základní úlohy kombinatorické optimalizace [Kvas00]:

A. Permutační úloha, kde optimalizační úloha hledání globálního extrému funkce $f$ je definovaná nad symetrickou grupou $G_n$ složenou ze všech permutací $n$ parametrů

$$f: G_n \rightarrow R \tag{1}$$

při čemž permutace je definovaná $n$-ticí celých různých čísel

$$P=(p_1, p_2, \ldots, p_n) \tag{2}$$

Optimalizační/minimalizační problém lze formalizovat výrazem

$$P_{opt} = \arg\min_{P \in G_n} f(P) \tag{3}$$

Prostor řešení s $n!$ permutacemi je velmi velký, takže pro velká $n$ může čas CPU narůstat faktoriálně ( $n!$). Typickou permutační úlohou je úloha obchodního cestujícího (TSP).

B. Úloha s exponenciální mohutností prohledávaného prostoru. Nechť $R_m=\{1, 2,\ldots,r\}$ je množina obsahující prvních $r$ přirozených čísel, karteziánský součin $R_m^n$ obsahuje $n$-tice

$\pi=(\pi_1, \pi_2,\ldots, \pi_n) \in R_m^n = R_m \, x \, R_m x \ldots x \, R_m$, při čemž parametr $\pi_i$ nabývá celočíselných hodnot z uzavřeného intervalu $[1, r]$. Funkce

$$f: R_m^n \rightarrow R^+ \tag{4}$$

zobrazuje n-násobný karteziánský součin $R_m^n$ na reálná čísla. Optimalizační problém má tvar

$$\pi_{opt} = \arg\min_{\pi \in R_m^n} f(\pi) \tag{5}$$

Protože kardinalita množiny $R_m^n$ je $r^n$, může čas CPU pro velká $n$ narůstat exponenciálně. Typickou úlohou je problém rozkladu čísla NPP (Number Partitioning Problem).

Pro úlohy malého rozsahu lze použít enumerativní techniku nebo metodu větví a mezí. V některých případech lze použitím restrikce původního zadání řešit problém v polynomiálním čase. Další možností je použití aproximativního algoritmu, který produkuje řešení s předepsanou chybou. Časová složitost může být redukována v případě TSP úlohy na $O(n^3)$ nebo $O(n)$, podle velikosti stanovené chyby. Často používané jsou heuristické metody, které poskytují přibližná řešení. Stále více se prosazují stochastické evoluční techniky různé složitosti. Typickým představitelem této třídy algoritmů jsou klasické genetické algoritmy a pokročilé evoluční algoritmy využívající pravděpodobnostní modely různé složitosti.

## 3.1 Klasické optimalizační metody vs. evoluční algoritmy

V současné době jsme svědky stále širšího využívání evolučních algoritmů pro řešení teoretických úloh a rozmanitých úloh praxe. Lze dokonce hovořit o stále větší popularitě této třídy algoritmů. Je oceňována jejich robustnost - schopnost nalézt globální optimum i pro multimodální multikriteriální úlohy s ostrými zlomy povrchu účelové funkce (function/fitness landscape). Je však známo, že konkrétní verzi evolučního algoritmu lze použít efektivně jen pro jistý okruh problémů (no free lunch theorem). Není tedy účelné používat evoluční algoritmy pro jakýkoliv typ řešeného problému. Existuje řada klasických propracovaných metod, které jsou schopny uspokojivě řešit řadu optimalizačních úloh s relativně nízkou výpočetní náročností.

První významnou skupinu klasických metod tvoří algoritmy používané pro řešení klasických grafových úloh: hledání souvislých komponent, minimální cesty, minimálních stromových struktur, maximálního toku, optimálního párování hran, obarvení grafu.

Další významnou skupinu tvoří metody operačního výzkumu. Ke známým úlohách patří lineární a celočíselné programování, nelineární (kvadratické) programování a dynamické programování. Pro řešení lineárních úloh se často používá simplexová metoda, pro úlohy nelineárního programování s hladkou funkcí (smooth function) např. gradientní metody GPM (Gradient-projection Method) a GRG (Generalized Reduced Gradient). Pro NP-úplné problémy lze použít metodu větví a hranic (Branch and Bound) a simulované žíhání.

Při rozhodování o volbě metody je tedy vhodné klasifikovat předložený problém, zda jej nelze převést na některou úlohu z teorie grafů nebo operačního výzkumu. Např. je známo, že dekompozici grafu bez omezujících podmínek na velikost vzniklých subgrafů lze převést na úlohu hledání maximálního toku v síti. Úlohu rozmísťování obvodových uzlů do pravidelných pozic lze definovat jako kvadratický přiřazovací problém a řešit jej pomocí metody větví a mezí anebo po zjednodušení jako kvadratickou úlohu kvadratického programování [Lenga96].

Tento trend je respektován v profesionálních optimalizačních systémech, které agregují klasické metody, různé heuristické přístupy a evoluční algoritmy (např. systém Premium Solver V3.5, [Solv35]).

## 3.2 Heuristické metody

Heuristické metody patří k nejrozšířenějším metodám. Jsou vesměs založeny na iterativních technikách (deterministických nebo náhodných) postupného zlepšování řešení. Je možné nalézt globální optimum, ale v procesu hledání nelze specifikovat ani vzdálenost aktuálně dosaženého řešení od globálního optima, ani s jakou pravděpodobností může být globálního optima dosaženo. Nejznámějším přístupem např. v alokačních úlohách je uplatnění sekvencí párových nebo vícenásobných záměn objektů pro vylepšení konfigurace těchto objektů. Výběr dvojic je buď náhodný nebo systematický a akceptuje se každá rekonfigurace vedoucí ke zlepšení účelové funkce.

## 3.3 Klasické genetické algoritmy (SGA)

Klasické genetické algoritmy SGA (Simple Genetic Algorithms), jako samostatná podtřída evolučních algoritmů, patří mezi stochastické optimalizační algoritmy založené na analogii s evolučními procesy probíhajícími v biologických systémech [Holl75], [Gold86], [Back96], [Kvas00], [Foge00], [Mari01].

Biologická evoluce (podle Darwinovy evoluční teorie a Mendlovy teorie dědičnosti) je progresivní změna genetické informace jedinců populace během mnoha generací v rámci evoluční epochy. Lze ji charakterizovat třemi základními fenomény:

- Přirozeným výběrem, kdy schopnější jedinci (s větší silou, výhodností) mají více potomků, než jedinci slabší, méně adaptovaní na prostředí,
- Procesem reprodukce, kdy nový jedinec vzniká rekombinací/křížením genetické informace rodičů, přičemž může docházet k poruchám vlivem mutace,
- Náhodným genetickým driftem, kterým může být exitus schopného jedince před realizací reprodukce, případně jeho náhodná mutace. Uplatňuje se u malých populací.

Tato teorie evoluce mnohdy nazývaná syntetickou teorií evoluce nebo také neodarwinismem, je založena na genech jako nositelích dědičnosti. Individuum je charakterizováno řetězcem genů – chromozómem (charakterizujícím jeho genetickou výbavu - genotyp). Fenotyp individua je dán interakcí jeho genotypu s prostředím. Výhodnost jedince (fitness) je dána mírou jeho schopností přežít v daném prostředí. Jednotkou vývoje je populace jedinců, která se skládá z genofondu daného genotypem všech jedinců.

V současné době existuje celá paleta evolučních algoritmů založených na různé interpretaci evoluční teorie [Back96]. K nejznámějším patří Evoluční strategie (ES), Evoluční programování (EP) a Genetické programování (GP).

Objevují se také nové expanze původního Darwinovského paradigmatu, obohacující jedince o adaptabilní kognitivní orgán (Baldwinův efekt), popřípadě o sociální zkušenost reprezentovanou mémy, které zavedl R. Dawkinson ve své knize „The Selfish Gene" [Kva00a].

Evoluční algoritmy (EA) jsou typické svojí robustností, schopností řešit obtížné optimalizační a rozhodovací úlohy, které lze charakterizovat vlastnostmi jako je multimodálnost, multikriteriálnost a různými typy omezujících podmínek. Jejich nasazení je efektivní v úlohách, které lze definovat následovně:

- Prostor řešení je příliš rozsáhlý a chybí expertní znalost, která by umožnila zúžit prostor slibných řešení,
- Nelze provést matematickou analýzu problému,
- Tradiční metody selhávají,
- Jde o úlohy s mnohačetnými extrémy, kriterii a omezujícími podmínkami.

EA se používají pro numerickou i kombinatorickou optimalizaci, při návrhu obvodů, plánování výroby, strojovém učení, v tvorbě ekonomických, sociálních a ekologických modelů atd.

Je však třeba poukázat na jisté nevýhody evolučních algoritmů:

- Kvalitu řešení lze ohodnotit pouze relativně. Nelze otestovat, zda se jedná o globální optimum,

- Pro mnohé úlohy je typická velká časová náročnost,
- Pro příliš rozsáhlé úlohy poskytuje řešení příliš vzdálená od optima,
- Ukončení optimalizace je explicitní na základě časového limitu nebo stagnace kritériální funkce.

Pro návrh evolučních algoritmů existuje řada komerčních produktů, které umožňují rychlý návrh aplikací. Klíčovým krokem je volba zakódování problému, která by měla respektovat použité genetické operátory a náročnost výpočtu účelové funkce.

### 3.3.1 Formalizace činnosti SGA

Návrh genetického algoritmu pro řešení optimalizačního problému je určen vždy konkrétní specifikací evolučních principů a jejich algoritmizací. Nejdříve definujme základní pojmy, které budeme dále používat. Chromozóm (jedinec/individuum) kódující řešení je reprezentován binárním vektorem (řetězcem) konstantní délky $n$ :

$X = (X_0, X_1,..,X_{n-1})$, kde $X_i$ je $i$ - tou proměnnou řetězce

$x = (x_0, x_1,..,x_{n-1})$ je řetězec konkrétních instancí proměnných $X_i = x_i$, $x_i \in \{0,1\}$

$D = (X^1, X^2,.., X^N)$, $X^j \in D$ je množina $N$ řetězců, která specifikuje populaci $D$

$D \subseteq \{0,1\}^n$.

Nechť $f$ je účelová/cenová funkce definovaná nad množinou binárních vektorů délky $n$

$$f: \{0,1\}^n \rightarrow R \tag{6}$$

která ohodnotí každý binární vektor $x$ reálným číslem. Cílem je nalézt globální extrém funkce $f$. V případě minimalizační úlohy jde o nalezení vektoru

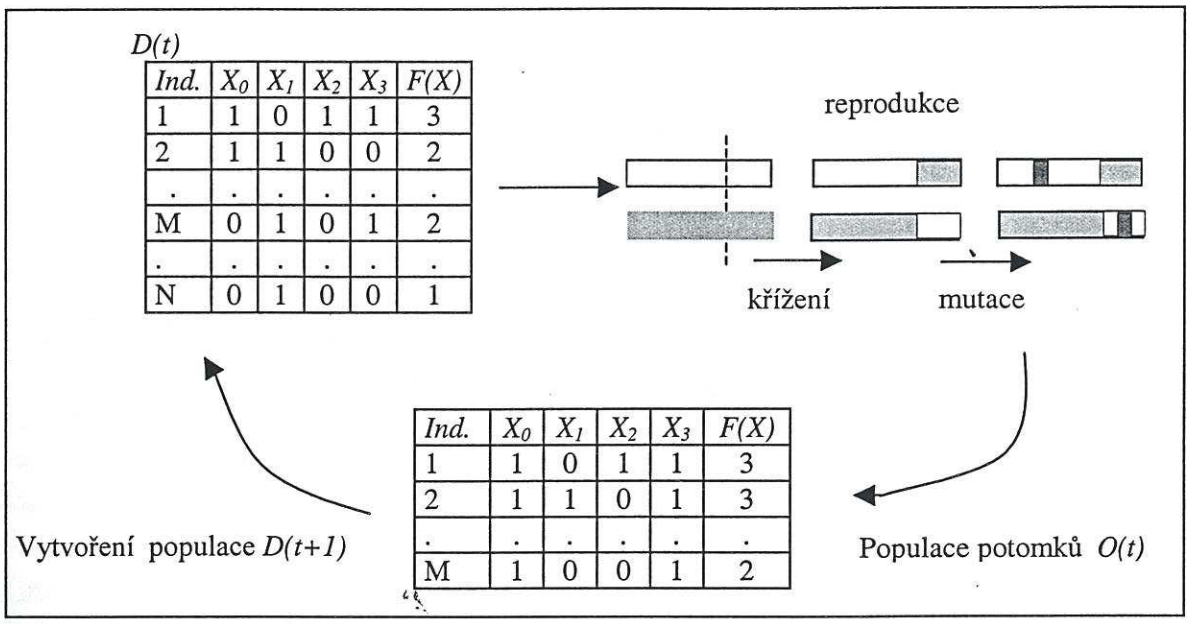
$$x_{opt} = \arg\min_{x \in \{0,1\}^n} f(x) \tag{7}$$

Funkce $f$ se zpravidla transformuje na funkci výhodnosti $F$ (fitness funkci) tak, aby původní optimalizační úloha byla převedena vždy na maximalizační úlohu a bylo dosaženo vhodného měřítka fitness funkce. Užití této transformace usnadňuje také změnu selekčního tlaku, který výrazně ovlivňuje konvergenci evolučního procesu.

Činnost klasického GA algoritmu lze popsat následujícím pseudokódem:

```
(1) Nastav $t=0$, náhodně generuj počáteční populaci $D(0)$ s mohutností $N$,
(2) Proveď ohodnocení jedinců populace $D(t)$ funkcí výhodnosti $F(X)$,
(3) Generuj populaci potomků $O(t)$ s mohutností $M \leq N$ použitím operátorů křížení a
    mutace,
(4) Vytvoř novou populaci $D(t+1)$ nahrazením části populace $D(t)$ jedinci z $O(t)$,
(5) Nastav $t \leftarrow t+1$,
(6) Pokud není splněna podmínka pro ukončení algoritmu, jdi na (2).
```

Schématické znázornění genetického algoritmu je na obr.1. V prvním kroku je generována počáteční populace vektorů řešení/jedinců. Každý jedinec je ohodnocen pomocí fitness funkce $F(X)$. V procesu křížení (uplatňovaném s četností křížení $p_c$) vznikají rekombinací vybraných dvojic individuí/rodičů dva potomci/řetězce, které jsou s četností $p_m$ náhodně mutovány a zařazovány do populace potomků. V procesu tvorby nové rodičovské populace část jedinců z populace potomků vytěsňuje jedince předchozí rodičovské populace. Celý proces se opakuje v rámci evoluční epochy, která je specifikována maximálním počtem generací nebo na základě stupně saturace aktuální populace identickými jedinci. Před aktivací GA je nutné specifikovat typy genetických operátorů a hodnoty parametrů - četnost selekce, křížení a mutace.



| *Ind.* | $X_0$ | $X_1$ | $X_2$ | $X_3$ | *F(X)* |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 1 | 3 |
| 2 | 1 | 1 | 0 | 0 | 2 |
| . | . | . | . | . | . |
| M | 0 | 1 | 0 | 1 | 2 |
| . | . | . | . | . | . |
| N | 0 | 1 | 0 | 0 | 1 |

| *Ind.* | $X_0$ | $X_1$ | $X_2$ | $X_3$ | *F(X)* |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 1 | 3 |
| 2 | 1 | 1 | 0 | 1 | 3 |
| . | . | . | . | . | . |
| M | 1 | 0 | 0 | 1 | 2 |

Obr. 1 Schéma klasického genetického algoritmu

## 3.3.2 Teorie schémat

Evoluční proces reprodukce populace řešení, specifikovaný operátory selekce, křížení a mutace vedoucí k nalezení optima nebo suboptima, byl formalizován J. Hollandem [Holl75], s použitím teorie schémat, která byla později rozpracována dalšími autory [Gold86]. Z teorie i praxe genetických algoritmů je znám stále aktuální problém vhodného zakódování řešení a volby vhodného sortimentu genetických operátorů, které výrazně ovlivňují konvergenci řešení. Většina modelů GA se opírá o zmíněnou teorii schémat – schéma je podobnostní šablona definovaná nad abecedou *{0,1*}*. Volný symbol "*" znamená, že se na příslušné pozici binárního řetězce, odpovídající danému schématu může vyskytovat libovolná hodnota 0 nebo 1. Říkáme, že řetězec $r \in \{0,1\}^n$, je příkladem/vzorem schématu $t \in \{0,1,*\}^n$, pokud v každé pozici schématu s jiným znakem než "*" je hodnota znaku v řetězci stejná jako ve schématu. Počet pevně stanovených pozic ve schématu *H* se nazývá řádem schématu *o(H)*, vzdálenost mezi první a poslední pevnou pozicí se nazývá délkou schématu $\delta(H)$. Schéma (0**1) má tedy řád 2 a délku 3. Místo sledování šíření jednotlivých jedinců je účelnější sledovat šíření schémat, která reprezentují příslušnou podmnožinu jedinců.

Nechť v čase *t* obsahuje populace *D(t)* počet *m* chromozomů vyhovujících podobnostnímu schématu *H*, tedy

$$m = m(H,t)$$

Výskyt schématu v populaci po realizaci reprodukčních operátorů je dán vztahem [Gold86]

$$m(H,t+1) \geq m(H,t) \cdot \frac{f(H)}{\bar{f}} \left[ 1 - p_c \cdot \frac{\delta(H)}{n-1} - o(H) \cdot p_m \right] \qquad (8)$$

Vztah (8) je důležitou formulí pro vysvětlení činnosti SGA. Slovně se vyjadřuje jako teorém o schématech: *Počet krátkých nadprůměrných schémat nízkého řádu v jednotlivých generacích exponenciálně roste.* Z teorému bezprostředně vyplývá hypotéza o stavebních blocích: *Genetický algoritmus upřednostňuje a přeskupuje nadprůměrná schémata nízkého řádu nazývaná stavební bloky (částečná řešení daného problému).* Delší schémata vyšších řádů jsou operátory poškozována, a to úměrně své definující délce a svému řádu. Tento fenomén rozbíjení stavebních bloků je hlavní příčinou pomalé konvergence klasických genetických algoritmů hlavně v případě nelineárních multimodálních účelových funkcí a klamných problémů (deceptive problems). Při řešení klamných úloh vykazují SGA anomální chování. Stavební bloky obsahují komponenty, které mohou směrovat řešení do lokálního optima. Existují totiž suboptimální schémata, která mají vyšší ohodnocení než optimální schémata stejného řádu a délky.

Dodejme, že vztah (8) je často citován, ale jde jen o jednu z možných verzí popisu šíření schémat v populaci, která předpokládá proporcionální výběr rodičovských jedinců pro rekombinaci a dostatečně velkou populaci jedinců. V uvedeném vztahu dominuje fenomén rozbíjení schémat, obecně však platí, že křížením rodičů schémata nejen zanikají, ale také vznikají. Jednou z možností jak snížit pravděpodobnost rozbíjení částečných řešení, je použití genetického operátoru inverze, který zrcadlově přeskupuje proměnné řetězce v náhodně vybraném segmentu řetězce. Naše výsledky experimentů s tímto operátorem, realizované v [D14] byly nejednoznačné a značně závisely na řešeném problému. V současné době je klasická teorie schémat podrobována kritické analýze a hledají se nové obecnější modely činnosti SGA.

Účinnost evolučního procesu závisí na detekci, spojování, filtraci a expanzi stavebních bloků, které jsou částí optimálního řešení. Předpokládá to identifikovat závislosti a nezávislosti mezi proměnnými (linkage learning). Klasické genetické algoritmy zachycují tyto vztahy implicitně selekcí slibných rodičovských jedinců a jejich rekombinací. V genetickém algoritmu není uchovávána žádná explicitní informace, která skupina proměnných přispívá, a jak významně, k účelové funkci slibných řešení. Proces rekombinace realizovaný křížením je náhodný a může porušit mnohé z vazeb mezi proměnnými. Hlavní problém je v tom, že rekombinační proces je odstíněn od znalosti řešeného problému a jeho potenciální dekompozice.

Dílčí překonání těchto problémů umožnilo zavedení messy-genetických algoritmů (messy-GA) [Gold99] s proměnnou délkou chromozómů a explicitní indexaci genů. Pro klasické i messy-GA však zůstává nevyřešen problém nastavení četností aktivace jednotlivých genetických operátorů – předpokládá znalost experta pro daný problém. Expertní znalost je často využívána při návrhu fuzzy regulátoru pro nastavování výše uvedených parametrů na základě statistik odvozených z aktuální populace. Výsledky získané genetickými algoritmy s tímto typem řízení parametrů však nejsou jednoznačné.

Jak už bylo zmíněno v první kapitole, principiálním řešením uvedených problémů při konstrukci a ladění SGA je nová koncepce pokročilých evolučních algoritmů označovaných jako EDA (Estimation Distribution Algorithm) algoritmy nebo jako PMBGA algoritmy (Probabilistic Model Building Genetic Algorithms) [Pel99c], [Larra02].

## 3.4 EDA algoritmy

Jejich činnost je založená na teorii pravděpodobnosti - používají statistickou informaci obsaženou v populaci nadějných řešení/jedinců k detekci (ne)závislostí mezi proměnnými řešení. Novým znakem tohoto přístupu je globální pohled na celou populaci při tvorbě pravděpodobnostního modelu pro daný problém. Noví jedinci jsou generováni podle odhadnutého pravděpodobnostního rozdělení. Na rozdíl od klasických genetických algoritmů jsou EDA algoritmy schopny explicitně vyjádřit jednoduché i vícenásobné závislosti mezi proměnnými a tedy detekovat stavební bloky optimálního řešení, což umožňuje efektivně optimalizovat složité nelineární a klamné úlohy.

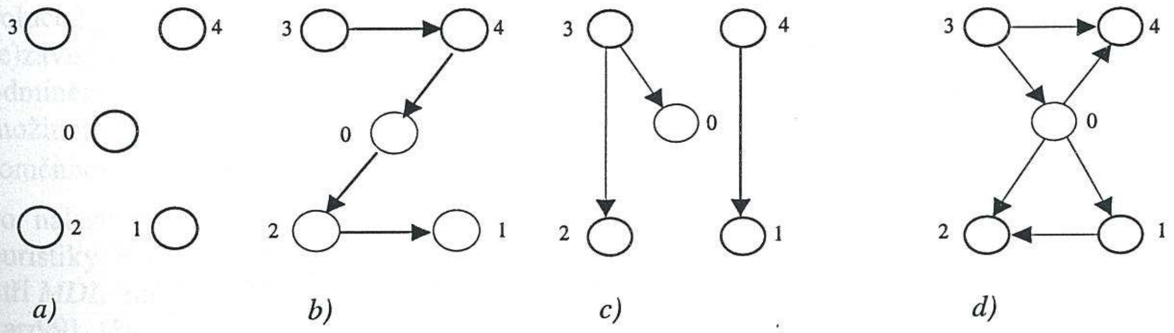
Činnost EDA algoritmu lze popsat následujícím pseudokódem:

```
(1) Nastav $t=0$, náhodně generuj počáteční populaci $D(0)$ s mohutností $N$,
(2) Vyber množinu $D^s(t)$ slibných jedinců z populace $D(t)$ s mohutností $M \leq N$,
(3) Proveď odhad pravděpodobnostního rozložení vybraných jedinců $p(X \mid D^s(t))$,
(4) Generuj množinu potomků $O(t)$ s mohutností $M$ (vzorkováním získaného rozložení),
(5) Vytvoř novou populaci $D(t+1)$ nahrazením části populace $D(t)$ jedinci z $O(t)$,
(6) Nastav $t \leftarrow t+1$,
(7) Pokud není splněna podmínka pro ukončení algoritmu, jdi na (2).
```

Fundamentálním problémem je odhad pravděpodobnostního rozložení $p(X|D^s(t))$. Z teorie je znám vztah pro sdružené rozložení pravděpodobnosti náhodného vektoru pomocí lokálních pravděpodobnostních rozložení s využitím podmíněných pravděpodobností

$$p(X) = \prod_{i=0}^{n-1} p\left(X_i \mid X_0, X_1, \ldots, X_{i-1}\right) \tag{9}$$

Složitost pravděpodobnostního modelu závisí na předpokládané interakci mezi proměnnými. Tyto závislosti/nezávislosti jsou modelovány grafovými strukturami. Uzly grafu odpovídají proměnným vektoru $X$, orientované hrany modelují závislost/kauzalitu mezi proměnou a rodičovskými proměnnými. Na obr. 2 jsou grafové modely používané pro EDA algoritmy různé složitosti s příslušným typem faktorizace.



*a)* $p(X) = p(X_3)\, p(X_0)\, p(X_2)\, p(X_4)\, p(X_1)$,

*b)* $p(X) = p(X_3)\, p(X_4 | X_3)\, p(X_0 | X_4)\, p(X_2 | X_0)\, p(X_1 | X_2)$

*c)* $p(X) = p(X_3)\, p(X_0 | X_3)\, p(X_2 | X_3)\, p(X_4)\, p(X_1 | X_4)$

*d)* $p(X) = p(X_3)\, p(X_0 | X_3)\, p(X_4 | X_3, X_0)\, p(X_1 | X_0)\, p(X_2 | X_1, X_0)$

Obr. 2 Grafické modely a sdružené rozložení pravděpodobnosti pro algoritmy:
a) UMDA, b) MIMIC, c) BMDA, d) BOA, EBNA

První evoluční algoritmy založené na jednoduchých pravděpodobnostních modelech znázorněných na obr. 2a, 2b, 2c vznikly asi před pěti lety. Nejjednodušší model předpokládá, že všechny proměnné jsou vzájemně nezávislé a sdružené pravděpodobnostní rozložení je dáno součinem marginálních pravděpodobností. S tímto modelem pracuje algoritmus PBIL (Population Based Incremental Learning) [Balu94], UMDA (Univariate Marginal Distribution Algorithm) [Mueh96]. Algoritmus MIMIC (Mutual Information Maximizing for Input Clustering) [Etxe99], [Larr99], [Balu97] a BMDA (Bivariate Marginal Distribution Algorithm) [Pel98a] pracují s grafem závislosti (dependency graphs) a umožňují modelovat podvojné závislosti. FDA algoritmus (Factorized Distribution Algorithm) využívá grafový pravděpodobnostní model specifikovaný expertem [Mueh98] - je hlavně určen pro aditivně rozložitelné problémy.

Nejobecnějším pravděpodobnostním modelem je Bayesovská síť *(B, $B_p$)*, kde *B* je struktura sítě (obr. 2d) a $B_p$ parametry sítě specifikující sdružené pravděpodobnostní rozložení. Bayesovská síť je obecně používaná pro modelování multinomických dat s diskrétními i spojitými proměnnými [Brad00], [Gelm98], [Gott98], [Gott99]. Umožňuje reprezentovat vícenásobné pravděpodobnostní interakce mezi proměnnými. Nechť pro každou proměnnou $X_i$ je $\Pi_{X_i} \subseteq \{X_0, X_1, .., X_{i-1}\}$ podmnožina (rodičovských) proměnných, které jí ovlivňují a přitom proměnné $X_0, X_1, .., X_{i-1}$ jsou podmíněně nezávislé, pak sdružené pravděpodobnostní rozložení náhodného vektoru *X* lze vyjádřit vztahem

$$p(X) = \prod_{i=0}^{n-1} p\left(X_i \mid \Pi_{X_i}\right) \tag{10}$$

Vztah (10) lze interpretovat jako zakódování struktury Bayesovské sítě *B* reprezentující podmíněné nezávislosti mezi proměnnými. Bayesovská síť na obr. 2d representuje např. nezávislost proměnné $X_4$ na proměnných $X_1$, $X_2$ (11) nebo nezávislost proměnné $X_1$ na proměnných $X_2, X_3, X_4$ (12).

$$p(X_4 \mid X_0, X_1, X_2, X_3) = p(X_4 \mid X_3, X_0) \tag{11}$$

$$p(X_1 \mid X_0, X_2, X_3, X_4) = p(X_1 \mid X_0) \tag{12}$$

Ilustrace činnosti Bayesovského evolučního optimalizačního algoritmu je na obr. 3. Klíčovou operací je konstrukce/indukce grafu Bayesovské sítě z populace slibných jedinců/řešení $D^s(t)$ a výpočet parametrů $B_p$. Jde o proces učení Bayesovské sítě, který se opakuje každou generaci evolučního procesu. V první fázi učení jde o určení struktury Bayesovské sítě, která reflektuje (ne)závislosti uzlů/proměnných. Ve druhé fázi navazuje proces učení parametrů - hodnot podmíněných pravděpodobností s ohledem na strukturu sítě *B*. Parametry jsou reprezentovány množinou tabulek $CPT_0$-$CPT_{n-1}$, specifikujících podmíněné pravděpodobnosti pro každou proměnnou $X_i = x_i$ (uzel sítě) pro všechny instance $\pi_{X_i}$ jejich rodičovských proměnných.

Pro nalezení struktury *B* lze použít stochastické metody, genetické algoritmy a rychlé heuristiky. K ohodnocení kvality *B* lze použít metriky s různou složitostí. K nejpoužívanějším patří *MDL* metrika *(Minimal Length Description)* a *BD* metrika *(Bayes-Dirichletova metrika)* [Larr99], [Heck95], [Frid02]. V EBNA algoritmu *(Estimation of Bayes Network Algorithm)* [Etxe99] je použita *BIC* metrika *(Bayesian Information Criterion)*.

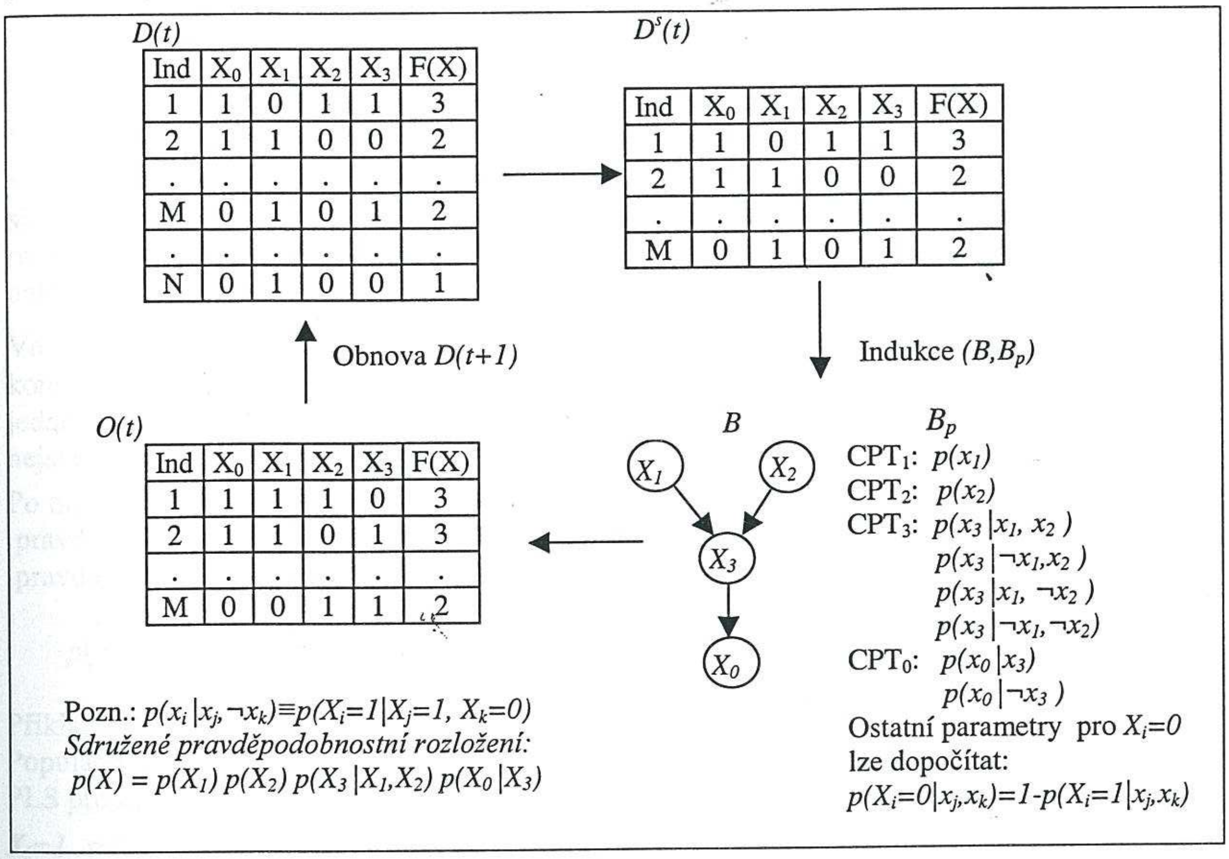
Použití *MDL* metriky vychází z filozofického přístupu nazývaného Occamova břitva/pravidlo (Occam razor), které preferuje ze soupeřících teorií nejjednodušší z nich. V kontextu teorie kódování jde o nalezení optimálního modelu (kodéru) pro kompresi dat *D* $(D \equiv D^s)$. Optimální model minimalizuje celkovou délku kódu složeného z kódu dat a kódu modelu. Celkovou délku kódu (v bitech) lze vyjádřit celkovou sumou

$$L(B, B_p, D) = L(B) + L(B_p) + L(D/B), \tag{13}$$

přičemž *L(B)* je délka zakódování struktury grafu, $L(B_p)$ je délka zakódování CPT tabulek a *L(D/B)* je zakódování dat pro danou strukturu grafu *B*. Výpočet délky kódu prvních dvou položek je jednoduché. Zakódování dat *D* pro známou síť *B* vychází z Huffmanova kódování.

$$L(D|B) = M\sum_{i=0}^{n-1} H(X_i \mid \Pi_{X_i}), \tag{14}$$

kde $H\left(X_i \mid \Pi_{X_i}\right)$ je podmíněná entropie, $M$ je počet jedinců populace. Výhodou této metriky je, že není třeba explicitně stanovit horní mez složitosti modelu. Nevýhodou je, že nelze použít apriorní informace o řešeném problému.



$D(t)$

| Ind | $X_0$ | $X_1$ | $X_2$ | $X_3$ | F(X) |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 1 | 3 |
| 2 | 1 | 1 | 0 | 0 | 2 |
| . | . | . | . | . | . |
| M | 0 | 1 | 0 | 1 | 2 |
| . | . | . | . | . | . |
| N | 0 | 1 | 0 | 0 | 1 |

$D^s(t)$

| Ind | $X_0$ | $X_1$ | $X_2$ | $X_3$ | F(X) |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 1 | 3 |
| 2 | 1 | 1 | 0 | 0 | 2 |
| . | . | . | . | . | . |
| M | 0 | 1 | 0 | 1 | 2 |

$O(t)$

| Ind | $X_0$ | $X_1$ | $X_2$ | $X_3$ | F(X) |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 0 | 3 |
| 2 | 1 | 1 | 0 | 1 | 3 |
| . | . | . | . | . | . |
| M | 0 | 0 | 1 | 1 | 2 |

$B_p$

CPT$_1$: $p(x_1)$
CPT$_2$: $p(x_2)$
CPT$_3$: $p(x_3 | x_1, x_2)$
$p(x_3 | \neg x_1, x_2)$
$p(x_3 | x_1, \neg x_2)$
$p(x_3 | \neg x_1, \neg x_2)$
CPT$_0$: $p(x_0 | x_3)$
$p(x_0 | \neg x_3)$
Ostatní parametry pro $X_i=0$ lze dopočítat:
$p(X_i=0|x_j,x_k)=1-p(X_i=1|x_j,x_k)$

Pozn.: $p(x_i | x_j, \neg x_k) \equiv p(X_i=1 | X_j=1, X_k=0)$
*Sdružené pravděpodobnostní rozložení:*
$p(X) = p(X_1)\, p(X_2)\, p(X_3 | X_1, X_2)\, p(X_0 | X_3)$

Obr. 3 Ilustrace Bayesovského evolučního algoritmu

Při použití Bayes-Dirichletovy *(BD)* metriky (odvozené z metod Bayesovské statistiky) lze zahrnout do procesu konstrukce $B$ sítě apriorní informace o pravděpodobnostním rozložení struktury a parametrů sítě. Aposteriorní pravděpodobnost Bayesovské sítě $B$ pro data $D$ (zde interpretujeme $D \equiv D^s$) lze stanovit použitím Bayesova teorému

$$p(B \mid D) = \frac{p(D,B)}{p(D)} = \frac{p(D \mid B)p(B)}{p(D)}, \tag{15}$$

kde $p(B|D)$ je aposteriorní pravděpodobnost sítě $B$ podmíněná daty $D$,
$p(B)$ je apriorní pravděpodobnost sítě $B$,
$p(D|B)$ je pravděpodobnost dat $D$ pro síť $B$ (marginální věrohodnostní funkce),
$p(D)$ představuje normativní koeficient.

Čím větší je hodnota $p(B|D)$, tím je síť $B$ věrohodnějším modelem dat $D$. Při srovnávání různých sítí pro stejnou datovou strukturu (populaci) lze vypustit jmenovatele $p(D)$ v rovnici (15)

$$p(B|D) \propto p(D|B)\, p(B) \tag{16}$$

V případě, že všechny sítě jsou stejně pravděpodobné (není k dispozici znalost experta) platí

$$p(B|D) \propto p(D|B) \tag{17}$$

Nalezení sítě *B* s maximální aposteriorní pravděpodobností (věrohodností) je pak ekvivalentní nalezení sítě *B*, která maximalizuje pravděpodobnost dat *D*. Obecně, výpočet pravděpodobnosti *p(D,B)* je velmi nesnadný. D. Heckerman [Heck95] odvodil metriku, která vychází kromě jiného z předpokladu, že rozložení lokálních podmíněných pravděpodobností má charakter Dirichletova rozdělení. *BD* metrika je pak dána vztahem

$$p(D,B) = p(B)\prod_{i=0}^{n-1} \prod_{\pi_{X_i}} ML(\, p(X_i \,|\, \pi_{X_i})) \tag{18}$$

kde *ML* je marginální věrohodnostní funkce ($\pi_{X_i}$ je množina instancí $\Pi_{X_i}$). Podrobnější vztah je uveden v článku A_4 a A_3. Nevýhodou klasického přístupu konstrukce Bayesovské sítě je nutnost specifikace maximální složitosti sítě – většinou dané maximálním počtem rodičovských proměnných *k* (jinak řečeno *k* určuje maximální počet vstupních hran každého uzlu sítě *B*). Možným řešením tohoto problému je využít *p(B)* pro penalizaci složitosti sítě.

Ve všech navržených verzích BOA algoritmů je pro nalezení sítě *B* použit jednoduchý konstruktivní algoritmus, který využívá elementární grafovou operaci – opakované přidávání jedné hrany v aktuální síti *B* s maximální hodnotou *BD* metriky. Hrany, které vytvářejí cyklus nejsou akceptovány.

Po nalezení Bayesovské sítě *B* lze z populace $D^s$ stanovit množinu parametrů $B_p$. Podmíněné pravděpodobnosti pro jednotlivé proměnné lze stanovit podle definice podmíněné pravděpodobnosti

$$p(X_i \,|\, \Pi_{X_i}) = \frac{p(X_i, \Pi_{X_i})}{p(\Pi_{X_i})} \tag{19}$$

Příklad stanovení položky CPT je uveden v článku A-7.

Populace potomků *O(t)* se získává PLS vzorkováním (Probabilistic Logic Sampling). PLS produkuje hodnotu proměnné kvazináhodně:

$X_i=1$ pokud $r \leq p(X_i=1 \,|\, \Pi_{X_i}=\pi_{X_i})$ jinak $X_i=0$ (*r* je náhodné číslo s rovnoměrným rozložením v intervalu [0,1]). Proces vzorkování začíná uzly s marginálním rozdělením (v obr. 3 jsou to uzly $X_1$, $X_2$) a pokračuje uzly, pro které jsou již definovány rodičovské proměnné. Z takto získané populace potomků *O(t)* a předchozí rodičovské populace *P(t)* je vytvořena populace *P(t+1)* s přihlédnutím k fitness funkci jedinců z obou populací. Celková časová složitost konstrukce sítě *B* a nové populace jedinců je dána vztahem $O(n^2N+n^3)$.

Konvergence BOA algoritmu a obecně EDA algoritmů závisí na globálním selekčním tlaku a vyváženosti pravděpodobnostního modelu. Model by měl být dostatečně přesný a přitom dostatečně obecný, aby v procesu inference byla také generována nová slibná řešení mimo rámec aktuální populace.

Ověřování účinnosti jednotlivých variant EDA algoritmů je tedy stále aktuální. Je málo experimentálních studií, které porovnávají EDA algoritmy s klasickými SGA algoritmy a tradičními metodami. Letos se k nim přiřadila kniha: Pedro Larrañaga a Jose Lozano: „Estimation of Distribution Algorithms“, [Larra02]. Část testů porovnává účinnost jednotlivých EDA algoritmů mezi sebou, ostatní testy porovnávají vybrané EDA algoritmy vůči klasickým SGA algoritmům a heuristickým metodám. Za zmínku stojí dvě úlohy:

1) Úloha nalezení podmnožiny výběrových charakteristik/rysů pro klasifikační systém, kde algoritmus EBNA (patřící do třídy Bayesovských algoritmů) [Larra02] produkoval nejlepší průměrné řešení ve srovnání s klasickými SGA algoritmy

s jednobodovým a uniformním křížením a dvěma sekvenčními heuristikami. EBNA algoritmus byl rychlejší než SGA algoritmy a pomalejší než heuristické metody,

2) TSP problém, na kterém byla testována účinnost EBNA se dvěma klasickými GA algoritmy se specifickými operátory křížení. Jeden z SGA algoritmů produkoval lepší řešení než EBNA. Pokud byl EBNA rozšířen o lokálně-optimalizační proceduru, výsledky byly shodné.

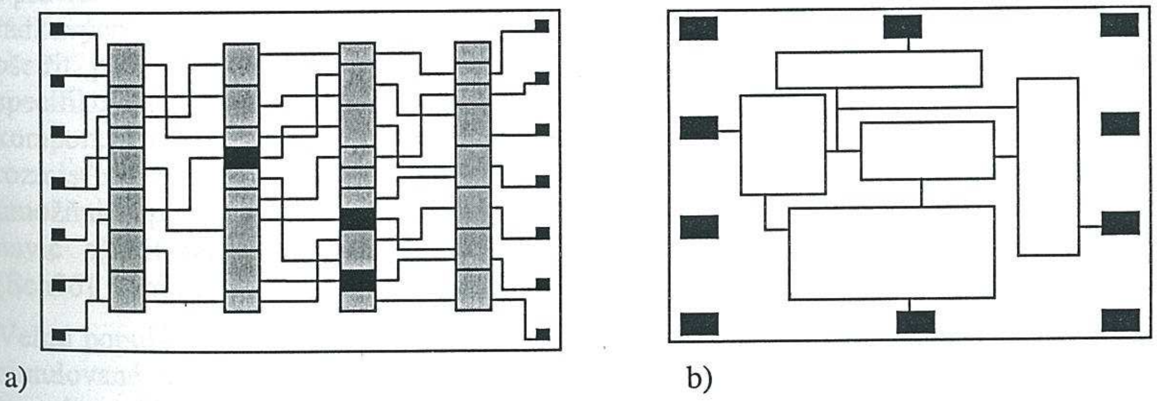
## 4 Úlohy dekompozice a alokace

S úlohou dekompozice se lze setkat při řešení rozmanitých úloh inženýrské praxe. Jde o známou koncepci "divide et impera", kdy složitá rozsáhlá úloha je rozdělena na několik podúloh s menší velikostí a složitostí. Tyto úlohy nejsou nezávislé a je snahou nalézt dekompozici s minimem vzájemných interakcí. Jako příklad lze uvést dekompozici rozsáhlých telekomunikačních sítí, programových produktů, databázových systémů a rozsáhlých číslicových integrovaných obvodů.

Úloha alokace je zmiňována často v kontextu alokace zdrojů (finančních), plánování výroby apod., které jsou řešeny často metodami operačního výzkumu. V této práci je analyzována alokační úloha, která je charakterizována hledáním vhodných pozic pro objekty s definovanou interakcí ve dvou nebo trojrozměrném prostoru v kontextu fyzického návrhu číslicových obvodů.

Návrh číslicových integrovaných obvodů lze charakterizovat pěti základními etapami: specifikací, logických návrhem (zahrnuje návrh architektury, funkčních boků a obvodovou realizaci), fyzickým návrhem, výrobou a testováním. Fyzický návrh se standardně člení na 3 základní etapy: členění/dekompozici, rozmisťování/alokaci a propojování. Charakter těchto etap závisí na použitém návrhovém stylu: desky s plošnými spoji (PCB), VLSI čipy, multičipové moduly (MCM) a hybridní integrované obvody (v dnešní době se již příliš nepoužívají).

Na obr. 4 jsou schématicky zobrazeny dva nejčastější návrhové styly (layouts) VLSI čipu - standardní buňky (standard cells) a funkční bloky (macrocells) s globálním propojení.

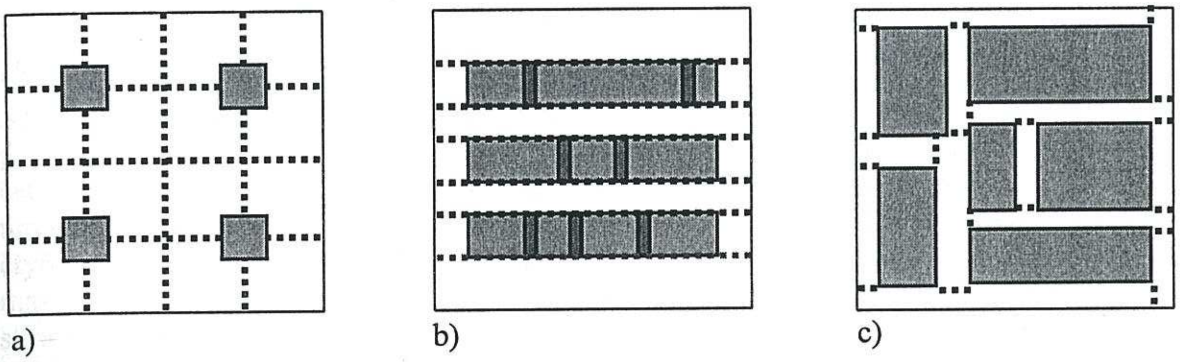


■ externí vývod, ▨ standardní buňka, ■ průchozí buňka ■ externí vývod

Obr. 4 Návrhové styly VLSI čipu včetně topologie globálního propojení
a) standardní buňky, b) funkční bloky.

## 4.1 Úloha alokace/rozmístění obvodových uzlů

Úloha rozmísťování je determinována použitým návrhovým stylem; základní typy jsou schématicky zobrazeny na obr. 5. V případě platformy standardních buněk, hradlových polí a PCB jde o pravidelné pozice (šedě označené pravoúhelníky), do kterých jsou obvodové komponenty (buňky) umisťovány. Kriteriem optimality v případě PCB a hradlových polí je realizace propojení komponent zabezpečující plnou funkčnost obvodu.



Obr. 5 Návrhové styly pro alokaci a) hradlové pole a PCB, b) standardní buňky, c) funkční bloky.

V případě standardních buněk jde o minimalizaci plochy čipu, které se dosahuje minimalizací sumární délky propojení. Rozmísťování funkčních bloků je složitější, protože bloky jsou různých rozměrů a tvarů [Drec98]. Pokud navíc není tvar a rozměr fixní (floorplanning problem), jde o velmi obtížně řešitelný problém. Kriterium optimality zahrnuje plochu čipu, poměrový koeficient tvaru (aspect ratio) a kritické zpoždění signálů na spojích.

Existuje řada metod, jak dosáhnout stanovených cílů. Univerzální grafově orientovanou metodou je metoda minimálních řezů (min-cut algorithm) [Breu77]. Použitím sekvence horizontálních a vertikálních řezů jsou obvodové komponenty členěny tak, aby počet spojů v řezu byl minimální. V současné době se používají sofistikovanější varianty s lineární složitostí [Fidu82], [Alpe96]. V případě hradlových polí, PCB desek a standardních buněk s pravidelnými pozicemi obvodových komponent, je možné použít kanonickou množinu řádkových a sloupcových řezů (v případě standardních buněk a sloupcových řezů je nutné ošetřit případné překrytí buněk). V případě funkčních bloků není poloha řezu předem specifikována – je nutné splnit omezující podmínku přípustnosti řezu tak, aby podmnožiny komponent vymezené řezem bylo možné umístit do vymezené oblasti čipu. Obtížnost tohoto rozmisťování shora dolů vedla ke vzniku duální metody rozmisťování zdola nahoru, která umožňuje uplatnit rotace agregovaných funkčních bloků a vzájemné propojení, musí však navíc ošetřit možné překročení poměrového koeficientu tvaru (aspect ratio) [Esbe92], [Sch96]. V rámci diplomové práce D-20 jsme experimentálně ověřili účinnost tohoto přístupu.

Velmi populární alokační metodou (používanou např. v návrhovém systému FPGA Xilinx) je simulované žíhání D-8, které je založeno na analogii s procesem postupného ochlazováním taveniny. Jde o stochastickou optimalizační metodu, kdy náhodně generovaná rekonfigurace obvodových komponent je akceptována kvazi-náhodně na základě kvality nové konfigurace. Tento přístup zabraňuje uvíznutí v lokálním optimu účelové funkce. Další oblíbenou a názornou metodou je metoda založená na analogii s mechanickým systémem těles vzájemně propojených pružinami (forced-directed placement). Tato metoda je dvoufázová - v první fázi se nalezne relativní pozice komponent na volné ploše a v druhé fázi je nutné ošetřit jejich vzájemné překrytí. V případě regulární struktury pozic je nutná normalizace jejich relativních pozic. Tato metoda byla používaná hlavně pro rozmísťování diskrétních součástek na PCB deskách nebo pro nalezení počátečního rozmístění funkčních bloků VLSI čipu.

Z pedagogického hlediska jde o velmi názornou metodu, která byla podrobně zkoumána v rámci diplomové práce D-4.

Moderní metodiky rozmísťování jsou vesměs založeny na sofistikovaných iterativních heuristických metodách někdy v kombinaci s genetickými algoritmy [Drec98], [Mazu99], [Alpe96]. Pro rozmísťování standardních buněk jsou používány algoritmy s permutačním zakódováním, podobně jako při řešení úloh obchodního cestujícího (TSP). V případě alokace funkčních bloků je používáno sofistikované zakódování podle složitosti zvolených cílů. V návrhovém systému SAGA [Mazu99] je použito binárního grafu určujícího absolutní pozici pevných bloků.

Kritériem je minimalizace čipu s garantovaným propojením. Systém SAGA je kombinací genetického algoritmu a simulovaného žíhání. SA algoritmus řídí velikost populace GA a četnost mutace během evoluční epochy. Komplexnější návrhový systém EXPLORER umožňuje fyzický návrh čipu s flexibilním tvarem funkčních bloků (floorplaning) [Drec98] se čtyřmi optimalizačními kritérii zahrnujícími plochu čipu, jeho poměrový koeficient tvaru, maximální zpoždění propojovacích cest a nahuštění spojů. Řešení je zakódováno pomocí stromové struktury řezů (slicing tree) specifikujících rozvržení funkčních bloků, jejich implementaci, orientaci a kritické zpoždění sítí (spojů).

### 4.1.1 Heuristická metoda limitovaných řezů

V článku A-1 je popsána originální heuristická metoda rozmísťování obvodových uzlů do pravidelných pozic, která je zaměřená zejména na platformu desek s plošnými spoji (PCB) a hradlová pole dle obr. 5a. Pro tento styl lze rozmisťovací problém formalizovat následovně: Je zadán hypergraf $H=(V,E)$ reprezentující obvod s $n=|V|$ uzly a $m=|E|$ hranami odpovídající signálovým sítím a poziční regulární graf $G=(P,D)$ reprezentující $p= |P|$ pozic a $d = |D|$ hran s jednotkovou délkou spojující sousední pozice. Cílem je nalezení jednoznačného zobrazení $f_p : V \rightarrow P$ které minimalizuje účelovou funkci, jež je koncipována tak, že umožňuje nejen minimalizaci celkové délky spojů, ale současně rovnoměrné rozložení spojů. Metoda využívá koncept dynamicky generované posloupnosti kanonických řezů, přičemž hodnota kanonických řezů (počet spojů v řezu) je limitována zdola. Minimalizační proces se zastaví, pokud horizontální řezy dosáhnou hranice $L_h$ a vertikální řezy hranice $L_v$. Nastavení těchto parametrů výrazně ovlivňuje optimalizační proces. Při příliš nízké hodnotě se algoritmus mění na standardní algoritmus minimálních řezů, který minimalizuje pouze sumární délku spojů. Při nastavení obou parametrů na hodnoty vyšší než je střední hodnota horizontálních respektive vertikálních řezů algoritmus produkuje rozmístění sice s rovnoměrným rozložením spojů, ale sumární délka spojů narůstá.

Dosažené experimentální výsledky potvrzují teoretické předpoklady, pokud jsou hodnoty limitů na úrovni 80-ti až 90-ti procent střední hodnoty řezů počátečního rozmístění. Problém volby hodnot limitů je řešen v diplomové práci D-10. Úloha experta je zde nahrazena posloupností několika optimalizačních běhů (s testovací rutinou) pro postupně snižované hodnoty limitů. Předností nové heuristické metody je malá časová složitost, nelze však zaručit dosažení globálního optima.

### 4.1.2 Paralelní klasický genetický algoritmus

Článek A-2, navazující na dřívější práci autora B-5 a B-6, je experimentální studií využití klasického genetického algoritmu pro řešení rozmisťovacího problému pro platformu PCB a hradlových polí. Pro zlepšení konvergence evolučního procesu byl algoritmus obohacen o dva fenomény – fenomén hybridizace a paralelizace.

Účelová funkce je standardní - minimální sumární délky spojů. Délka spojů se aproximuje minimálním poloobvodem *MSP* (Minimal Semi-Perimeter - polovina obvodu minimálního obdélníka obepínajícího obvodové uzly spoje). Tato aproximace se hojně používá, protože je dostatečně přesná a výpočetní složitost je podstatně nižší, než v případě použiti minimální kostry *MST* (Minimal Spanning Tree).

Paralelizace byla realizována na platformě transputeru T800 a T9000 s kruhovou topologií procesorů (ring topology), umožňující současný běh až osmi procesů – izolovaných genetických algoritmů se vzájemnou kooperací pouze na bázi migračního operátoru. Hybridizace genetického algoritmu byla realizována přidáním rychlé heuristiky založené na párových záměnách pozic obvodových uzlů, která je aktivována s nastavitelnou četností a intenzitou.

V článku jsou prezentovány experimenty testující citlivost řešení na řídicích parametrech hybridního algoritmu zahrnujících četnost migrace, četnost mutace, četnost a intenzitu přídavné heuristiky a vliv počtu procesů. Se zvyšujícím se počtem procesů/subpopulací roste rychlost konvergence algoritmu, ale také režijní čas komunikace mezi procesy. Byl potvrzen předpoklad o nutnosti udržování dostatečné úrovně izolace vývoje jednotlivých subpopulací- migrace byla aktivována v optimálním případě jedenkrát za epochu trvající 50 generací.

Prokázalo se také, že použití rychlé heuristiky výrazně přispělo k nalezení kvalitnějšího řešení. Nutno však dodat, že použití přídavné heuristiky může způsobit uváznutí optimalizačního procesu v lokálním optimu. Vliv mutace byl v souladu s očekáváním, i když byly nalezeny dva oddělené intervaly optimálních hodnot četnosti mutace. Pro dynamické nastavování četnosti mutace byl také testován fuzzy řídící modul navržený systémem FIDE [Fide92], D-17.

Nutnost nastavování velkého počtu parametrů a výběru vhodných typů genetických operátorů je velkou nevýhodou tohoto typu klasických genetických algoritmů. Jak bylo řečeno v kap. 3.3.1 principiálním řešením těchto problémů je použití EDA algoritmů. V následujících kapitolách jsou prezentovány Bayesovské evoluční algoritmy navržené pro úlohy dekompozice.

## 4.2 Bayesovské algoritmy v úloze dekompozice hypergrafů

Dekompozice systémů, např. telekomunikačních systémů, databázových systémů a VLSI obvodů, patří k častým úlohám členění rozsáhlých systémů na menší celky. Jde o systémy charakterizované množinou entit a vícečlennými relacemi mezi nimi. Takové systémy lze reprezentovat hypergrafy a problém členění lze převést na členění hypergrafů.

Základní úlohu členění hypergrafu lze formalizovat následovně: Je dán hypergraf $H=(V,E)$ s $n=|V|$ uzly a $m=|E|$ hyperhranami, dále jen hranami. Cílem je nalezení členění množiny uzlů $V$ do disjunktních podmnožin (modulů) $V1$, $V2$ $(V=V1 \cup V2)$, které minimalizuje počet externích hran (20) při splnění omezující podmínky na mohutnost obou podmnožin (21). Množina externích hran je označena $E_{cut}(V1,V2)$ a jejich počet $C_1$.

Účelová/cenová funkce (metrika) $C_1$ je definována vztahem:

$$C_1(V_1,V_2)= |E_{cut}(V_1,V_2)| = |\{e\in E \,/\, e\cap V_1 \neq \emptyset,\ e\cap V_2 \neq \emptyset\}| \qquad (20)$$

s omezující podmínkou

$$(1\text{-}\alpha)\, n\,/\,2 \leq |V_1|,\quad |V_2| \leq (1+\alpha)n\,/2,\qquad \alpha \in (0,\, 1) \qquad (21)$$

Balanční koeficient $\alpha$ určuje charakter dekompozice. V případě, že koeficient má nulovou hodnotu, jde o bisekci/půlení (bisection) produkující subgrafy/moduly se stejným počtem uzlů, viz obr. 6a, nebo členění s nenulovým koeficientem, viz obr. 6b. Je zřejmé, že nenulový balanční koeficient poskytuje členění s menší hodnotou řezu.

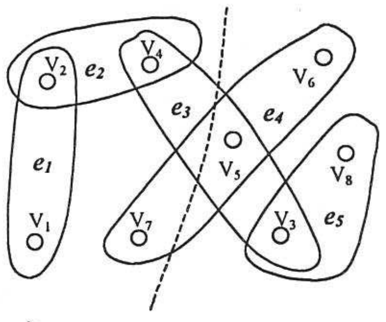
Jinou formou vyjádření rozdílné velikosti podgrafů je prostý rozdíl jejich velikostí $C_2 = || V_1 | - | V_2 ||$. Pro realizaci dekompozice hypergrafu na $k$ částí je možné použít dva přístupy. Rekurzivní přístup je založen na opakované dekompozici vznikajících modulů/subgrafů na dvě části. Nerekurzivní přístup předpokládá konstrukci počátečního rozdělení hypergrafu na $k$ částí, které je iterativně je vylepšováno.

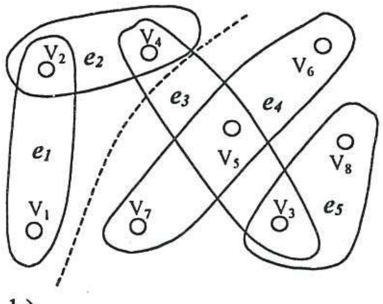
Rekurzivní algoritmy se používají častěji. Účelová funkce je definována vztahem

$$C_1 (V_1, V_2, \dots, V_k) = | E_{cut} (V_1, V_2, \dots, V_k)| = |\{e \in E \,/\, \exists\, i, j,\ i \neq j\ e \cap V_i \neq \emptyset,\ e \cap V_j \neq \emptyset\}| \quad (22)$$

s omezující podmínkou

$$(1-\alpha)\, n/k \leq | V_i | \leq (1+\alpha)\, n/k, \quad \alpha \in (0, 1), \quad i, j = 1, \dots, k \qquad (23)$$





Obr.6 Příklad dekompozice hypergrafu a) půlení (s nulovým balančním koeficientem) $\alpha=0$, $C_1=2$, b) členění na dvě nestejné části, $\alpha=1/4$, $C_1=1$

.

Z hlediska rychlosti dekompozice je nejvýhodnější použít bisekci. Tento přístup však neumožňuje detekovat přirozené shluky v hypergrafu. Řešením je zavedení složené účelové funkce

$$R_c = C_1 / F_R (V_1, V_2, \dots, V_k), \qquad (24)$$

kde $C_1$ je počet externích hran, jmenovatel je funkcí velikostí modulů $V_1$ až $V_k$. Jednou z možností je použít součinovou funkci $F_R (V_1, V_2, \dots, V_k) = | V_1 | * |V_2| *, \dots, * | V_k|$. Další varianty $R_c$ jsou uvedeny v článku [Alpe95].

Úlohu dekompozice hypergrafu lze také ilustrovat v kontextu dekompozice obvodu např. takto: je dán obvod $O = (B, S)$, kde $B = \{b_1, b_2, \dots, b_n\}$ je množina obvodových uzlů, $S = \{s_1, s_2, \dots, s_m\}$ je množina signálových sítí. Cílem je rozdělit obvod $O$ do $k$ modulů $\{M_1, M_2, \dots, M_k\}$ podle zvoleného kritéria. Kromě kritéria $C_1$ reprezentující počet externích signálových sítí lze použít další dvě kritéria – celkový počet vývodů modulů $P_c$ a celkový počet spojek $W_e$, které bez redundance propojí obvodové uzly jednotlivých sítí. Počet vývodů $P_c$ lze vyjádřit vztahem:

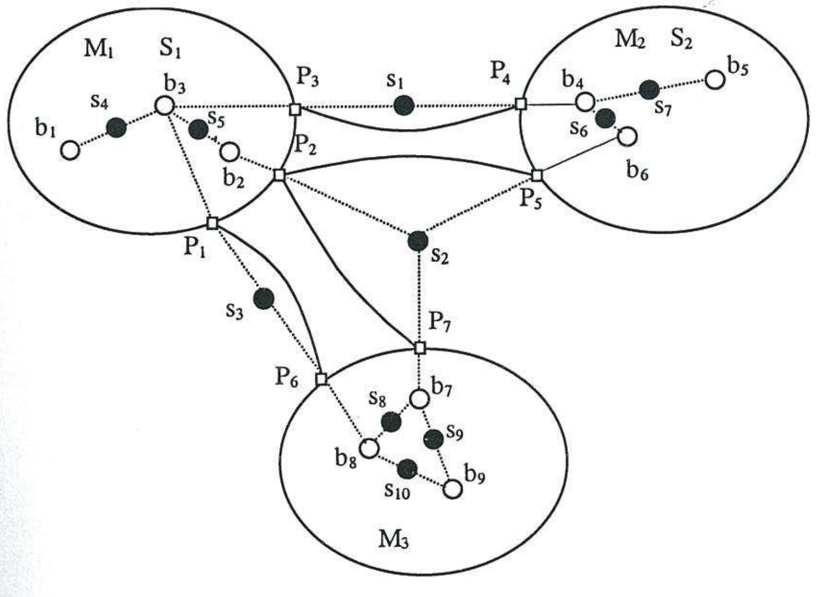
$$P_c = \sum_{s \in S} |\{ i | \ 0 < |s \cap M_i| < | s | \}|, \quad i = 1, \dots, k \qquad (25)$$

$$W_e = \sum_{s \in S} ( |\{ i | \ 0 < |s \cap M_i| < | s | \}| - 1), \quad i = 1, \dots, k \qquad (26)$$

Mezi uvedenými účelovými funkcemi platí vztah:

$$P_c = W_e + C_I \tag{27}$$

Na obr. 7 je příklad dekompozice obvodu s devíti obvodovými uzly a deseti sítěmi na tři části. K externím sítím patří podmnožina sítí *{$s_1$, $s_2$, $s_3$}*, počet externích spojů $C_I = 3$, počet vývodů $P_c = 7$, počet spojek $W_e = 4$.



Obr. 7 Dekompozice obvodu do tří modulů se zobrazením vývodů a spojek externích sítí

V případě, že obvod obsahuje pouze dvoučlenné spoje platí:

$$C_I = W_e \text{ a následně } P_c = 2\,W_e \tag{28}$$

Existují i další metriky (např. koeficient hustoty shluků, koeficient absorpce sítí ve shlucích apod.), které jsou používány k vyhledávání shluků v hypergrafu/obvodu v případě, kdy *k* je velké vzhledem k velikosti hypergafu/obvodu. Dekompozice hypergrafu (obvodů) se specifikovaným koeficientem balance je NP-úplným problémem. Proto jsou pro řešení této úlohy velmi často používané četné heuristiky založené na F-M algoritmu (Fiduccia-Matheyses) [Fidu82], který má časovou složitost $O(n \log n)$.

## 4.2.1 Srovnání evolučních algoritmů SGA, BMDA a BOA

V článku A-3 jsou prezentovány výsledky získané třemi variantami klasického genetického algoritmu a dvěma verzemi EDA algoritmů na úloze členění hypergrafů na dvě a čtyři části. (bisekce a kvadrisekce). První dvě varianty klasického genetického algoritmu SGA a SGAN vycházejí z verze publikované v článku B-5. Třetí varianta SGAH obsahuje navíc rychlou heuristickou proceduru publikovanou v článku A-2. Byla použita účelová funkce $W_e$, která je v článku reprezentována symbolem $L$ (tedy $L = W_e$).

Byly navrženy a implementovány dva algoritmy EDA: a) BMDA (Bivariate marginal distribution algorithm) na základě publikovaného článku v [Pel98a], b) BOA (Bayesian optimization algorithm) na základě základní programové verze publikované v [Pel99a], [Pel99c]. BMDA algoritmus pracuje s jednodušším pravděpodobnostním modelem, reprezentovaný lesem, který umožňuje detekovat pouze podvojné závislosti mezi proměnnými. Původní BMDA algoritmus [Pel98a] byl modifikován tak, aby mohl pracovat nad konečnou abecedou, což umožnilo realizovat nerekurzivní členění hypergrafu na více částí (*k*-way partitioning). Byla použita modifikace původní metriky, využívající Pearsonovu (chí-kvadrát) statistiku a pro srovnání modifikovaná K2 metrika tak, aby umožňovaly detekci závislosti mezi dvojicemi proměnných i v případě použití celočíselné abecedy.

Původní algoritmus BOA [Pel99a] umožňoval řešit jednoduché umělé problémy nad binární abecedou pro omezený počet proměnných. Nově navržená koncepce má stavebnicovou strukturu a umožňuje explicitní specifikaci řešeného problému na úrovni fenotypu a globálních dat. Pro ohodnocení kvality konstruované Bayesovské sítě byla použita metrika K2, kterou lze odvodit jednoduchou úpravou z metriky *BD* (uvedeno např. v A-4).

Byly realizovány tři typy experimentů pro zjištění celkového počtu evaluací (a tedy rychlosti konvergence) pro nalezení předem známého globálního optima:

1. Srovnání BMDA a BOA algoritmu s různým stupněm složitosti pravděpodobnostního modelu BOA (pro $k=1$ až $k=5$; v tomto kontextu má *k* význam maximálního počtu vstupních hran uzlu Bayesovské sítě). Minimální velikost populace a minimální počet evaluací vyžadoval algoritmus BOA pro složitost modelu $k=3$. Potvrdil se předpoklad, že složitost modelu má být v relaci se složitostí řešeného problému.
2. Srovnání algoritmů BMDA, BOA *(k=3)*, SGA a SGAH. Minimální počet evaluací potřeboval algoritmus BOA.
3. Srovnání BMDA, SGAN a SGAH pro nerekurzivní dekompozice na dva, čtyři a šest modulů. Minimální počet evaluací vyžadoval SGAH algoritmus, druhé nejlepší výsledky produkoval BMDA algoritmus.

Z experimentů je zřejmé, že v úloze bisekce hypergrafů poskytuje algoritmus BOA nejlepší výsledky s nejnižším počtem evaluací (časovou složitostí). Nerekurzivní dekompozice hypergrafu na *k* modulů je obtížnou úlohou pro všechny testované algoritmy, nejmenší časovou složitost dosahuje SGAH algoritmus následován BMDA algoritmem.

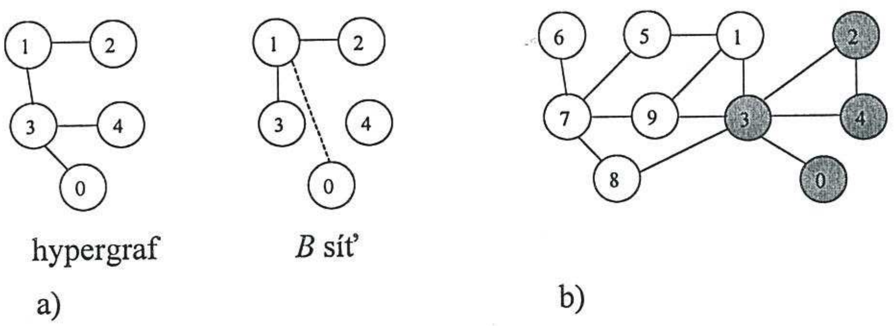
V publikaci [Mazu99] je provedeno srovnání účinnosti F-M algoritmu s klasickým genetickým algoritmem a hybridním genetickým algoritmem na známých testovacích úlohách dekompozice číslicových obvodů. Pro většinu úloh produkují oba genetické algoritmy kvalitnější řešení než F-M algoritmus, nutno však dodat, že s větší výpočetní dobou. V případě členění s definovaným balančním koeficientem je rozdíl výsledků ještě výraznější v neprospěch F-M algoritmu.

### 4.2.2 Algoritmus KBOA využívající apriorní informaci

V článku A-4 je popsána originální aplikace využívající při konstrukci sítě *B* dodatečných znalostí o řešeném problému, tzv. apriorní informaci, viz vztah (15). Větší apriorní pravděpodobnost má síť, která reflektuje znalost experta. Podle Heckermana [Heck95] lze apriorní pravděpodobnost sítě vyjádřit vztahem:

$$p(B) = c\kappa^{\delta}, \tag{29}$$

kde $c$ je normalizační konstanta, $\kappa \in (0,1]$ je penalizační koeficient a $\delta$ je symetrická diference mezi sítí *B* a apriorní sítí specifikované expertem. V našem případě se vychází z požadavku podobnosti Bayesovské sítě *B* a struktury hypergrafu *(V, E)*, jehož dekompozice je cílem optimalizačního procesu. Jestliže existuje hrana mezi uzly hypergrafu a tedy existuje závislost mezi oběma uzly, lze předpokládat, že by měla existovat také hrana mezi příslušnými uzly Bayesovské sítě. Při porovnávání dvou variant Bayesovské sítě *B* lze tedy využít informaci o počtu nepárových hran vůči členěnému hypergrafu. Preferuje se síť $B$ s minimálním počtem nepárových hran. Při konstrukci sítě $B$ postupným přidáváním jedné hrany platí volba $\delta = 0$, pokud se přidaná hrana (plná čára) vyskytuje i v hypergrafu. V opačném případě (čárkovaná hrana) je použito nastavení $\delta = 1$ (viz obr. 8a).



Obr. 8 Uplatnění apriorní informace a) pro síť $B$, b) pro počáteční populaci $D(0)$

Byl realizován také další experiment s využitím apriorní informace o struktuře hypergrafu. V hypergrafu byly vyhledány shluky uzlů (vyznačené šedým odstínem v obr. 8b) zvolené velikosti, které byly injektovány do počáteční populace řešení. Tímto postupem se významně ovlivnila počáteční struktura sítě $B$. Optimální velikost shluků činila 5-10% z celkového počtu uzlů hypergrafu. Použití obou typů apriorních informací vedlo ke zrychlení konvergence řešení.

### 4.2.3 BOA algoritmus pro rozmisťování

Techniku dekompozice hypergrafu lze také použít pro řešení rozmisťovacího problému. V článku A-5 je popsán rozmisťovací algoritmus pro PCB platformu (popřípadě pro standardní buňky), založený na hierarchickém rekurzivním algoritmu bisekce hypergrafu. Ve srovnání s klasickým GA pracující s permutačním zakódováním řešení poskytuje rozmísťovací algoritmus BOA kvalitnější řešení.

Je zajímavé, že v případě dvou grafů (IC67, IC151) algoritmus FORCE pracující se silovým modelem [Breu82] produkuje kvalitnější řešení než BOA. Příčinou je pravděpodobně použitý typ dekompozice s nulovým balančním koeficientem – bisekce. Řešením je použití členění s nenulovou hodnotou balančního koeficientu. BOA algoritmus tohoto typu je popsán v článcích A-8, A-9 (viz kap.4.2.6).

### 4.2.4 Multikriteriální BOA algoritmus pro dekompozici hypergrafů

V předchozích uvedených článcích bylo vždy použito jediné kriterium optimality dekompozice hypergrafů (minimalizace spojů v řezu - hodnoty řezu). V článku B-12 byl poprvé publikován BOA algoritmus pro multikriteriální optimalizaci, který byl testován na známé úloze batohu (knapsack problem).

Výsledky získané v článku B-12 byl pro autora inspirací pro návrh bi-kriteriálního BOA algoritmu pro členění hypergrafů, který je popsán v článku A-6. Jedním kriteriem je minimální hodnota řezu, druhým je rozdíl velikostí (balance) obou částí hypergrafu. Pro detekci a ohodnocení nedominantních a dominantních řešení byl použita metoda publikovaná v práci [Zitz99].

Článek A-6 je obsáhlou studií, která porovnává čtyři navržené varianty BOA algoritmu:

- Algoritmus Pareto- BOA s vektorovou účelovou funkcí,
- Algoritmus WSO1 BOA se skalarizovanou účelovou funkcí 1.typu,
- Algoritmus WSO2 BOA se skalarizovanou účelovou funkcí 2.typu,
- Algoritmus SOP BOA s jednoduchou účelovou funkcí, reprezentující první kritérium. Druhé kritérium je transformováno do omezující podmínky.

Z výsledků řešení je zřejmé, že WSO BOA (weighted sum optimization BOA) algoritmy mají nízkou výpočetní složitost, ale jsou velmi citlivé na použité hodnoty váhových koeficientů a typ členěných grafů. Algoritmus SOP (simple optimization process) produkuje řešení na jednom okraji Paretovské hranice, což odráží vliv omezující podmínky. Testy prokázaly, že navržený multikriteriální algoritmus Pareto-BOA produkuje kvalitní řešení rovnoměrně rozložená v Paretovské hranici. Na základě znalostí či rozhodnutí experta lze po skončení optimalizace vybrat řešení s nízkou hodnotou řezu a větší hodnotou balance nebo s větší hodnotou řezu a malou hodnotou balance.

### 4.2.5 Teorie a praxe multikritériálních BOA algoritmů

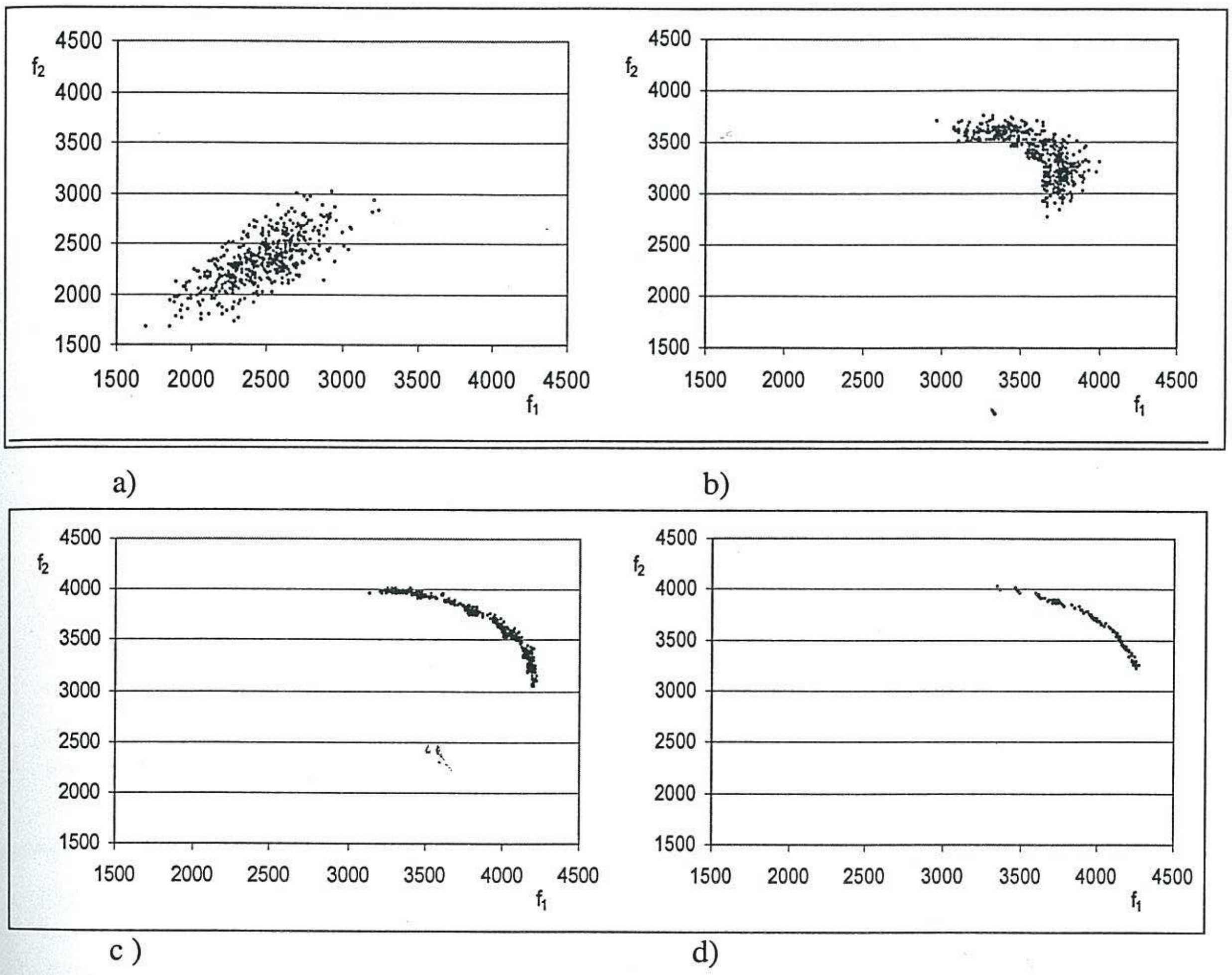
Zkušenosti publikované v předchozích dvou článcích A-6 a B-12 byly pro autora inspirací pro systematické zpracování problematiky multikriteriální optimalizace kombinatorických problémů v článku A-7. Jsou zde popsány často používané metody multikriteriální optimalizace, které nejsou založeny na koncepci Pareto-optimality a je specifikována propracovanější verze algoritmu Pareto-BOA. Jsou uvedeny základní pravděpodobnostní modely používané v algoritmech EDA a prezentována Bayesovská statistika pracující nad fragmentem populace řešení a fragmentem Bayesovské sítě pro stanovení hodnot podmíněné pravděpodobnosti (položky CPT tabulky) pro jeden uzel Bayesovské sítě.

Algoritmus Pareto-BOA je testován na třech bi-kriteriálních kombinatorických úlohách:

1. Teoretické úloze Onemax/Xor se známým globálním extrémem, kdy jedním kritériem je počet jedniček v binárním řetězci a druhým počet dvojic sousedních znaků/bitů s různou hodnotou. Pareto-BOA algoritmus nalezl všechna existující řešení Paretovské hranice (fronty), komparační algoritmus [Horn94] nenalezl řešení na jednom jejím okraji. Výsledky jsou doplněny grafem závislosti pokrytí předem známé Pareto hranice na velikosti populace.
2. Úloze členění reálných číslicových obvodů. Jde o pozměněnou variantou úlohy publikované v článku A-6. Výsledné Pareto hranice byly konstruovány agregací lokálních hranic získaných v opakovaných nezávislých optimalizačních bězích. Z výsledků je evidentní, že Pareto-BOA algoritmus umožňuje generovat bohatší sortiment řešení než WSO algoritmy se skalarizovanou účelovou funkcí.
3. Úloze batohu, která navazuje na výsledky publikované v B-12. Byl analyzován vliv velikosti populace a nalezena velikost populace při které Pareto-BOA algoritmus produkuje Paretovskou hranici plně pokrývající Paretovskou hranici produkovanou algoritmy [Horn94] a [Zitz99] a navíc poskytuje další řešení na jejích okrajích.

Na obr. 9 je ukázka formování Paretovské hranice bi-kriteriální úlohy batohu publikované v B-12 v různých fázích evolučního procesu.

Jde o maximalizační bi-kriteriální problém, zvýšení počtu kriterií lze však snadno implementovat.



Obr. 9 Ukázka rozložení řešení úlohy batohu po příslušném počtu generací a) g=1, b) g=25, c) g=50, d)g=100.

Otevřeným problémem zůstává časová složitost rozsáhlejších úloh. Řešením je paralelizace nejen konstrukce Bayesovské sítě, ale i paralelizace detekce řešení Paretovské hranice.

### 4.2.6 MBOA dekompoziční algoritmus s poměrovým řezem

Jistou nevýhodou optimalizačních Paretovských metod je jejich algoritmická a výpočetní složitost. Je vždy na zvážení experta, zda zvolí tuto metodu nebo zjednodušené metodiky (skalarizaci účelové funkce, prioritní sekvence účelových funkcí apod.).Ve vybraných aplikacích je možné a účelné agregovat dvě, popřípadě více kriterií do jedné účelové funkce. To je i případ stále aktuální úlohy hierarchické dekompozice rozsáhlých obvodových struktur respektující přirozené shluky obvodových uzlů. Lze vyzvednout dva stejně významné požadavky - nalezení subgrafů/modulů s minimální interakcí (hodnotou řezů) a minimálním balančním koeficientem reflektující rozdílnou velikost modulů. Řešením je zavedení složené účelové funkce $R_c$ (ratio-cut partitioning), definované podílem hodnoty minimálních řezů a součinu velikostí modulů, viz kap. 4.2.
Většinou používané heuristické metody založené na lokálních rekonfiguracích [Weic91] nezaručují nalezení optimálního řešení. Příznivější výsledky produkují klasické genetické algoritmy, které však, jak už bylo řečeno výše, vyžadují specifikaci genetických operátorů a

kvalitní nastavení řídicích parametrů. Tento stav věcí byl pro autora inspirací pro návrh specializovaného MBOA algoritmu. V článku **A-8** jsou presentovány tři varianty algoritmu:

1. Rekurzivní algoritmus RRC, který realizuje hierarchickou dekompozici grafů s využitím opakovaného členění na dvě části minimalizující účelovou funkci $R_c$,
2. Nerekurzivní algoritmus MRC, který realizuje (paralelní) dekompozici grafu na zvolený počet modulů, minimalizující hodnotu $R_c$ v jenom optimalizačním běhu,
3. Rekurzivní algoritmus RB, který se od RRC algoritmu liší tím, že používá pro dekompozici bisekci, tedy půlení grafů. Tato omezující podmínka je implementovaná na úrovni genotypu populace řešení.

Byly použity čtyři skupiny testovacích grafů. Regulární grafy různé velikosti a složitosti: graf typu housenka, náhodně generovaný graf s předem definovaným stupněm uzlů grafu a dva reálné grafy reprezentující číslicové obvody (náhodná logika). V případě regulárních grafů a grafu typu housenka, často používaných jako obtížné testovací úlohy, bylo globální optimum známo. Tabelované výsledky dokládají, že oba algoritmy RRC a MRC produkují řešení srovnatelné kvality a oba nalezly globální optimum pro regulární grafy a graf typu housenka. Nevýhodou MRC algoritmu je jeho velká časová složitost, vůči RRC algoritmu až desetinásobná. Algoritmus RB produkoval podstatně horší výsledky vůči RRC i MRC, protože použití bisekce omezuje detekci přirozených shluků grafových struktur. Je však vhodné poznamenat, že v případě grafů s malý podílem významných shluků mohou všechny zmíněné algoritmy poskytnout srovnatelná řešení.

RRC a RB algoritmy pracují s binárním zakódováním řešení, MRC algoritmus pracuje s řetězci nad celými čísly. To vyžadovalo výraznou modifikaci původního BOA algoritmu pracujícího nad binárními řetězci na pokročilejší verzi MBOA, která také akceptuje řetězce s celočíselnou abecedou.

Pro konstrukci grafové struktury pravděpodobnostního modelu byl použit koncept binárních rozhodovacích grafů (BDD), které reprezentují (ne)závislosti proměnných a podmíněné pravděpodobnosti úsporněji, viz obr. 10. CPT tabulka pro *i*-tou proměnnou Bayesovské sítě obsahuje v případě *k* rodičovských proměnných $2^k$ parametrů pro stanovení podmíněné pravděpodobnosti $p(X_i \mid \Pi_{X_i})$ přičemž $k = \mid \Pi_{X_i} \mid$. BDD kóduje pravděpodobnostní model úsporněji a přitom detailněji – umožňuje reprezentovat nezávislost proměnných pro konkrétní kontext/instance. Na obr.10 je hvězdičkou vyznačena nezávislost proměnné $X_1$ na proměnné $X_2$ pro konkrétní instanci proměnné $X_0=1$.

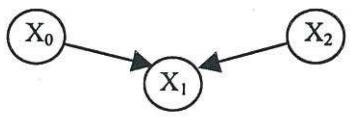
Pro každou proměnnou je konstruován jeden BDD graf, jehož listy specifikují podmíněné pravděpodobnosti této proměnné. *BD* metrika pro BDD graf je publikovaná v článku [Pel00a].

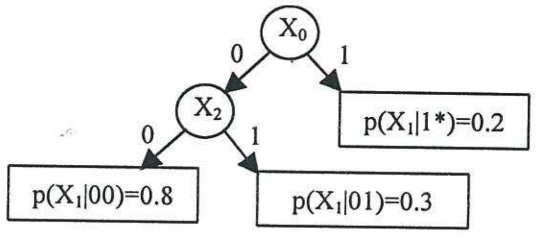
V algoritmu MBOA je konstrukce BDD stromu řízena inkrementální metrikou

$$\frac{\Gamma\left(\sum_{r\in\{0,1\}}\sum_{s\in\{0,1\}}(m_{r,s}+1)\right)\cdot\prod_{r\in\{0,1\}}\prod_{s\in\{0,1\}}\Gamma(m_{r,s}+1)}{\left(\prod_{r\in\{0,1\}}\Gamma\left(\sum_{s\in\{0,1\}}(m_{r,s}+1)\right)\right)\cdot\left(\prod_{s\in\{0,1\}}\Gamma\left(\sum_{r\in\{0,1\}}(m_{r,s}+1)\right)\right)} \qquad (30)$$

Položky $m_{r,s}$ vyjadřují počet výskytů jedinců v příslušné subpopulaci pro které platí $X_j=r$ ($X_j$ je proměnná/uzel rozkladu BDD grafu), $X_i=s$ ($X_i$ je cílová proměnná rozkladu BDD). Pro omezení složitosti BDD grafů je použita penalizační funkce, jejíž argumentem je počet listů BDD grafu. Uvedený vztah (30) je současně korekturou vztahu uvedeného v článcích

A-8 a A-9, kde došlo nedopatřením k částečné záměně operátorů násobení za operátory sumace. V prováděných experimentech byla použita bezchybná metrika, takže výsledky jsou platné.





CPT$_1$

| $X_0$ $X_2$ | $p(X_1=1/ X_0,X_2)$ |
|---|---|
| 0 0 | 0.8 |
| 0 1 | 0.3 |
| 1 0 | 0.2 |
| 1 1 | 0.2 |

CPT$_1$

| $X_0$ $X_2$ | $p(X_1=1/ X_0,X_2)$ |
|---|---|
| 0 0 | 0.8 |
| 0 1 | 0.3 |
| 1 * | 0.2 |

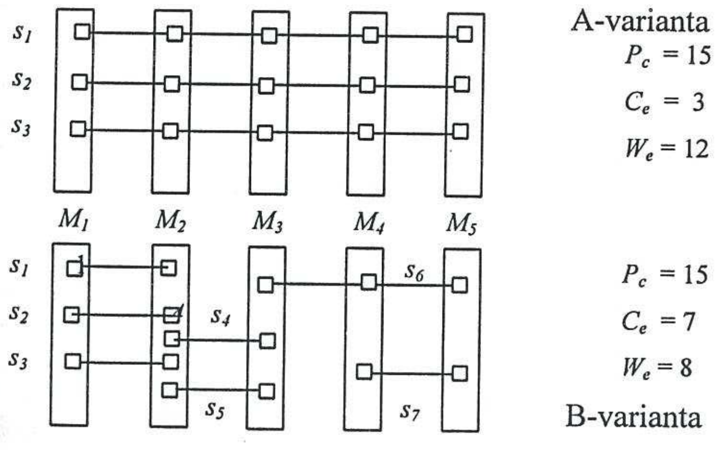
a) b)

Obr. 10 Zakódování pravděpodobnostního rozložení $p(X)=p(X_1 \mid X_0,X_2)\, p(X_0)\, p(X_2)$ a příslušné CPT tabulky: a) Bayesovská síť, b) BDD graf.

V MBOA algoritmu je použit zobecněný BDD model umožňující pracovat s hybridním genotypem řetězců zahrnující binární, celočíselné i spojité parametry – bližší popis je v publikaci B-16 a B-14.

### 4.2.7 MBOA dekompoziční algoritmus s volitelnou metrikou

V článku A-9 je detailněji propracována problematika volby typu kritérií optimality dekompozice hypergrafu, což je významné hlavně v kontextu členění číslicových obvodů. Aktuálnost použití kritérií $C_e$ (zde $C_e$=$C_1$), $W_e$, $P_c$ závisí na kontextu řešeného problému - zda jde o dekompozici obvodu pro vícevrstvé propojování, minimalizaci průměrného zpoždění sítě nebo minimalizaci počtu sítí mezi subgrafy/moduly pro účely snadné testovatelnosti. Na obr.11 je ukázka dvou případů dekompozice obvodu se stejnou hodnotou metriky $P_c$. A- varianta má pouze 3 externí sítě, ale délka propojení je rovna hodnotě $W_e$ = 12 . B-varianta má 7 externích sítí, ale délka propojení je dána hodnotou $W_e$=8. B-varianta je tedy výhodnější v kontextu délky spojů nebo průměrného zpoždění sítě.



Obr. 11 Ukázky odlišné dekompozice obvodu se stejnou hodnotou metriky $P_c$

Otevřenou otázkou zůstává jak ovlivní výběr dekompozičního kritéria hodnoty zbývajících dvou metrik. Byly testovány dva rekurzivní algoritmy – ARC (shodný s RRC v článku A-8) a AMB (shodný s RB v článku A-8) na dvou vybraných obvodových strukturách ze souboru používaných testovacích úloh střední složitosti [Benc98].

Byly realizovány 3 typy experimentů:

1. Dekompozice obvodové struktury byla realizována podle všech tří minimalizačních kritérií a pro každé z nich byly vyhodnoceny dosažené hodnoty metrik $C_e$, $W_e$, $P_c$. Z výsledků např. vyplývá, že v případě obvodu IC67 pro dosažení co nejmenší hodnoty $W_e$ je možné použít minimalizační kritérium $W_e$ nebo $P_c$, kritérium $C_e$ dává horší výsledky. Pro dosažení co nejmenší hodnoty $P_c$ platí tentýž závěr. V případě dekompozice obvodu IC116 je použití jednotlivých kritérií přibližně stejně výhodné. Souvisí to se strukturou obvodu - složitost/délka jednotlivých sítí se příliš neliší. V případě, že obvod obsahuje pouze dvoubodové sítě, je výsledek dekompozice stejný pro každé uvedené kritérium. Pro zobecnění získaných výsledků je nutný šířší sortiment testovacích úloh se specifikovaným rozložením shluků.
2. Srovnání algoritmů ARC a AMB pro stejné minimalizační kritérium. Podle předpokladů algoritmus AMB používající striktně bisekci produkuje horší výsledky.
3. Srovnání střední doby zpoždění sítě/signálu pro oba použité algoritmy. Z výsledku je evidentní, že nejkratší zpoždění je dosaženo při použití minimalizačního kritéria $W_e$ a algoritmu ARC.

Lze konstatovat, že minimalizační kritérium $W_e$ je univerzálnější a je vhodné jej použít při minimalizaci průměrného zpoždění v síti a při dekompozici obvodu do subobvodů realizovaných vícevrstvou technologií. Hodnotu $W_e$ lze tak interpretovat jako počet průchozích otvorů. Pro realizaci experimentů byl použit MBOA algoritmus popsaný v článku A-8 s BDD pravděpodobnostním modelem a byla navržena koncepce paralelizace konstruování BDD stromů.

V článku A-9 je také prezentována idea využití BDD grafu s *BD* metrikou nejen jako pravděpodobnostního modelu MBOA algoritmu, ale také pro konstrukci BDD grafů pro reprezentaci jedno a více-výstupových booleovských funkcí. Vychází se z analogie použitých datových struktur. V případě booleovské funkce je populací pravdivostní tabulka funkce a rozklad se realizuje v jednom běhu vůči hodnotám výstupní funkce.

Dosud používané heuristiky a klasické genetické algoritmy pro konstrukci BDD pro reprezentaci logických funkcí jsou orientovány zejména na klíčový problém určení pořadí proměnných pro rozklad booleovské funkce s využitím Shannonova teorému [Sasa96]. Autorem prezentovaný algoritmus je dvoufázový. V prvé fázi je konstruován binární rozhodovací strom (BDT) s implicitním pořadím proměnných podle *BD* metriky. Ve druhé fázi je BDT minimalizován rychlou heuristikou na základě dvou redukčních pravidel: 1) redukce uzlů se stejným následníkem a 2) sdílením všech ekvivalentních subgrafů [Sasa96].

Navržený algoritmus byl testován na často používané funkci multiplexoru až 20ti proměnných – bylo vždy dosaženo globálního optima. Nevýhodou použité metody je nutnost specifikace funkce pravdivostní tabulkou. Obecně také nelze zaručit, že výsledný BDD diagram je orientovaným diagramem (OBDD). Řešení tohoto problému je předmětem dalšího výzkumu.

## 5 Přehled výzkumné činnosti a ohlasy

Výzkum je považován za integrální součást pedagogické činnosti. Výzkumná činnost byla podporovaná v rámci grantů:

1. Paralelizace Bayesovského evolučního algoritmu, UIVT FEI VUT, FR0171/2001/G1, 2001
2. Výzkum informačních a řídicích systémů, CEZ MŠMT, MSM 262200012, 1999-2003
3. Výzkum a aplikace heterogenních systémů, GAČR, GA102/98/0552, 1999-2000
4. Vývoj flexibilních číslicových architektur, GAČR, GA102/95/1334, 1995-1997
5. Modely výpočtů a jejich paralelní implementace, GAČR, GA102/94/1096, 1994-1996
6. Paralelní výpočty a architektury, FVU VUT, C32/94-95, 1994-1995
7. Síť pro rozvoj pokročilého vzdělávání v informatice, EU Tempus, JEN 0449, 1994-1995

V dřívějším období byl výzkum realizován v rámci státních úkolů:

9. III-2-1/8: Návrh prvků a podsystémů počítače pomocí počítače, VUT Brno, 1975
10. III-2-1/5: Počítačový návrh logických systémů, VUT Brno, 1980

Výzkumná činnost byla také podpořena řadou diplomových prací D-1 až D-24. Výsledky některých prací se přímo využívají ve výuce - návrhový systém FREG2/M2 (D-17) pro návrh fuzzy alikací, GaDesign (D-21) pro návrh klasických genetických algoritmů a DEBOA (B-17) pro rychlé prototypování Bayesovských optimalizačních algoritmů.

Získané poznatky byly začleňovány do původních předmětů Navrhování prvků počítačů a Konstrukce počítačů. V současné době jsou využívány v novém předmětu Aplikované evoluční algoritmy, poprvé ve školním roce 2001/2002. Další využití se naskytne v horizontu dvou let při výuce nových předmětů "Soft Computing" a Počítačový návrh.

Vybrané citace článků:

A-3:

- Larrañaga, P., Lozano, J., A, editors: Estimation of Distribution Algorithms. Kluwer Academic Publishers, London 2002, ISBN 0-7923-7466-5
- Pelikán, M., Goldberg, D., E.: Genetic Algorithms, Clustering, and the Breaking of Symmetry. The 6th International Conference PPSN VI, Paris, France, 2000, pp. 385-394

A-4:

- Pelikán, M., Goldberg, D., E., Sastry, K.: Bayesian Optimization Algorithm, Decision Graphs, and Occam's Razor, Výzkumná zpráva No. 2000020, genetická laboratoř university v Illinoa, USA
- Pelikan, M., Sastry, K., Goldberg, D., E.: Evolutionary Algorithms + Graphical Models = Scalable Black-Box Optimization, IlliGAL Report No. 2001029, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, Illinois, 2001

# 6 Závěr

Cílem předkládané habilitační práce bylo prezentovat novou třídu evolučních algoritmů založených na pravděpodobnostních modelech, které by odstraňovaly problémy klasických genetických algoritmů s konvergencí a nutností ladění řídících parametrů. Byly uvedeny teoretické základy nového evolučního paradigmatu a navrženy nové varianty Bayesovského evolučního algoritmu umožňující řešit složité kombinatorické optimalizační úlohy s jedním a více kriterii s využitím apriorních informací o řešeném problému. Účinnost těchto algoritmů byla testována na vybrané skupině kombinatorických úloh dekompozice a alokace systémů, které je možné reprezentovat hypergrafy, popřípadě ohodnocenými hypergrafy. Obě úlohy byly řešeny také v kontextu fyzického návrhu číslicových obvodů, kam nesporně patří úloha členění a rozmísťování. Experimentální výsledky potvrzují schopnost algoritmů BOA, KBOA, Pareto-BOA a MBOA řešit složité nelineární úlohy a nalézt globální optimum známých testovaných úloh [Benc98], [Merz00], [Buim96].

Je nesporným faktem, že význam evolučních algoritmů stále roste, rovněž v souvislosti s prudce se rozvíjející oblastí inteligentních výpočtů (Soft Computing). Evoluční algoritmy (EA), fuzzy systémy (FS) a neuronové sítě (NS) samostatně, ale stále více ve vzájemné kooperaci, se stávají efektivním nástrojem při řešení složitých problémů z oblasti informačních technologií (IT). Klasické výpočetní metody včetně distribuovaných systémů často nestačí v reálném čase produkovat přijatelná řešení pro tento typ úloh. Tato skutečnost vedla ke vzniku klasických kooperačních dvojic algoritmů: EA/FS pro nastavování parametrů fuzzy regulátorů a EA/NS pro nastavování parametrů a struktury neuronových sítí. Obrácená kooperace FS/EA, využívající fuzzy regulátoru pro nastavení parametrů genetických operátorů je stalé více používaná i když výsledky do značné míry závisí na znalostech experta. Totální kooperace EA/FS/NS principiálně poskytuje velkou výpočetní sílu, vyvážení všech komponent však není snadné. Jestliže použijeme EDA variantu evolučního algoritmu EA v některé z výše zmíněných kooperacích, získáme robustnější výpočetní prostředek a celkový počet nastavovaných parametrů bude významně redukován.

Pro rychlé prototypování aplikací EA na bázi Bayesovského optimalizačního algoritmu byl implementován vývojový systém DEBOA B-17, který byl navržen na základě zkušeností prezentovaných v článcích A-3 a A-4. Jeho aplikační možnosti budou dále rozvíjeny, zejména v oblasti multikriteriální optimalizace a paralelního zpracování, viz B-14 až B-16.

Otevřeným problémem Bayesovských evolučních algoritmů zůstává řešení velmi rozsáhlých optimalizačních úloh. Kromě paralelizace jednotlivých etap BOA algoritmu je možná kooperace BOA a SGA algoritmu, což by umožnilo využít předností obou algoritmů. Další možností je návrh BOA algoritmu, který optimalizuje systém produkčních pravidel, následně použitých pro vlastní optimalizační proces. Dalším záměrem autora je návrh kooperujícího algoritmu na bázi BOA a PBIL algoritmu (Population Based Incremental Learning Algorithm). Uvedená témata budou předmětem dalšího výzkumu v rámci grantových projektů a diplomových prací.

## 7 Přehled vybrané literatury

[Alpe95] Alpert, C. J., Kahng, A. B.: Recent Directions in Netlist Partitioning: A Survey, Integration: The VLSI Journal 19 (1995), pp. 1-81.

[Alpe96] Alpert, C. J., Hagen, W., Kahng, A. B.: A Hybrid Multilevel/Genetic Approach for Circuit Partitioning: A Design Workshop, 1996, pp. 100-105.

[Balu94] Baluja, S.: Population-Based Incremental Learning: A method for Integrating Genetic Search Based Function Optimization and Competitive Learning, Carnegie-Mellon Report, CMU-CS-94-163, 1994.

[Balu97] Baluja, S., Davies, S.: Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space. Proceedings of the 14th International Conference on Machine Learning, pp. 30-38, Morgan Kaufmann, 1997.

[Back96] Back, T.: Evolutionary Algorithms in Theory and Practice, Oxford University Press, 1996, ISBN 0-19-509971-0.

[Benc98] A benchmark set, University of California, Los Angeles, VLSI CAD Laboratory, http://vlsicad.cs.ucla.edu/~cheese/benchmarks.html.

[Brad00] Bradlez, P. Louis, T. ,A.: Bayes and Empirical Bayes Methods for Data Analysis. Chapman & Hall/CRC, 2000, pp. 419, ISBN 1-58488488-170-4.

[Breu77] Breuer, M. A.: Min-cut Placement, Journal of Design Automation and Fault Tolerant Computing, Vol. 1, No.4., Oct.1977, pp. 343-362.

[Breu82] Breuer, M. A.: Forced-oriented Placement Algorithm, Experimental Study with IC Benchmarks, Chapter 5 and 6, pp. 89 - 162, delivered under personal request.

[Buim96] Bui, T. N., Moon, B. R.: Genetic Algorithm and Graph Partitioning. IEEE Transactions on Computers, Vol. 45, No.7, July 1996, pp. 841-855.

[Dasg98] Dasgupta, D., Michalewicz, Z.: Evolutionary Algorithms in Engineering Application. Springer Verlag, Berlin 1998, pp. 554, ISBN 3-540-62021-4.

[Drec98] Drechsler, R.: Evolutionary algorithms for VLSI CAD. Kluwer Academic Publishers, London 1998, ISBN 0-7923-8168-8.

[Esbe92] Esbensen, H.: A genetic Algorithm for Macro Cell Placement, Proceedings of the European Design Automation Conference, pp. 52-57, September 1992.

[Etxe99] Etxeberria, R., Larrañaga, P.: Global optimization using Bayesian networks. II. Symposium on Artificial Intelligence, CIMAF99, 1999.

[Fide92] The FIDE (Fuzzy Inference Development Environment) User and Reference Manual, Aptronix 1992.

[Fidu82] Fiduccia, C.,M., Mattheyses, R.: A Linear Time Heuristic for Improving Network Partitions, Proc. ACM/IEEE DAC, 1982, pp.175-181.

[Foge00] Fogel, D.: Evolutionary computation. IEEE Press, New York 2000, pp. 270, ISBN 0-7803-5379-X.

[Frie98] Friedman, N., Goldszmidt, M.: Learning Bayesian Networks with Local Structure, In Jordan ed. Learning and Inference in Graphical Models, 1998.

[Gelm98] Gelman, A., et al: Bayesian Data Analysis. Published by Chapman & Hall, London, 1998, ISBN 0 412 03991 5.

[Gold89] Goldberg, E., D.: Genetic Algorithms in Search, Optimization, and Machine Learning, Addison- Wesley, 1989.

[Gold99] Goldberg, E.,D., et al : Messy Genetic Algorithms Revisited. Nonuniform Size and Scale. Complex system 4 (1990), pp. 415-444.

[Gott98] Gottvald, A., Malczyk, R.: Bayesian Evolutionary Optimizations versus Fourier Parameter Estimations in NMR Data Inversions. Proceedings of the 3rd Japan-Central Europe Joint Workshop on Modelling and Simulation of Non-linear Engineering Systems and Related Phenomena. - Bratislava, Comenius University 1998, S. 290.

[Gott99] Gottvald, A.: Bayesian Evolutionary optimization. 5-th International Mendel Conference on Genetic Algorithms, Optimization Problems, Fuzzy Logic, Neural Networks, Rough Sets, June 24-26, 1999, Brno, Czech Republic, pp. 30-35, ISBN 80-214-1131-7.

[Heck95] Heckerman, D., Geiger, D., & Chickering, M. (1994). Learning Bayesian Networks: The combination of Knowledge and Statistical Data (Technical Report MSR-TR-94-09), Redmont, WA: Microsoft Research, 1995, pp. 1-53.

[Holl75] Holland, J., H.: Adaptation in Natural and Artificial System. The Univesity of Michigan Press, Ann Arbor, MI, 1975.

[Horn94] Horn, J., Nafpliotis, N., Goldberg, D.E.: A Niched Pareto Genetic Algorithm for Multiobjective Optimization, In Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, Vol. 1, pp. 82-87, Piscataway, New Jersey, June 1994. IEEE Service Center.

[Kvas00] Kvasnička, V,. Pospíchal, J., Tiňo, P.: Evolučné algoritmy, STU Bratislava, 2000, pp. 215, ISBN 80-227-1377-5.

[Kva00a] Kvasnička, V.: Umelá evolúcia, seminár kognitívne vedy-CogSci2000, katedra matematiky CHTF STU Bratislava, 2000, pp. 1-17.

[Larr99] Larrañaga, P. et al.: A review of the cooperation between evolutionary computation and probabilistic graphical models. Artificial Intel Review, 1999, preprint.

[Larra02] Larrañaga, P., Lozano,J., A, editors: Estimation of Distribution Algorithms. Kluwer Academic Publishers, London 2002, ISBN 0-7923-7466-5.

[Leng90] Lengauer, T.: Combinatorial Algorithms for Integrated Circuit Layout. John Wiley & Sons, 1990, pp. 697, ISBN 0 471 92838 0.

[Mari01] Mařík, V., Štěpánková, O., Lažanský, J. a kol.: Umělá inteligence (3). Academia Praha , 2001, ISBN 80-200-0472-6.

[Mazu99] Mazumder, P.: Genetic Algorithms for VLSI Design, Layout & Test Automation. Prentice Hall, 1999, pp. 338, ISBN 0-13-011566-5.

[Merz00] Merz P., Freisleben, B.: Fitness Landscapes, Memetic Algorithms and Greedy Operators for Graph Bi-Partitioning. Evolutionary Computation, Vol. 8, No. 1, pp. 61-91, 2000.

[Mueh98] Muehlenbein H., Rodriguez A. O.: Schemata Distributions and Graphical Models in Evolutionary Optimization. GMD Forschungs Zentrum Informationstechnik, 53754-St. Augustin, 1998, pp.1-21.

[Mueh96] Muehlenbein H., Paas, G.: From recombination of genes to the estimation of distributions, I. Binary parameters. Parallel Problem Solving from Nature, PPSN IV, pp. 178-187.

[Pel98a] Pelikan M., Muehlenbein H.: Marginal Distribution in Evolutionary Algorithms. 4th International Mendel Conference on Genetic Algorithms, Optimization Problems, Fuzzy Logic, Neural Networks, Rough Sets, June 24-26, 1998, Brno, Czech Republic, pp. 91-95, ISBN 80-214-1199-6.

[Pel99b] Pelikan, M., Goldberg, D. E., & Cantú-Paz, E.: Linkage Problem, Distribution Estimation, and Bayesian Networks. IlliGal Report No. 98013, February 1999, pp. 1-25.

[Pel99a] Pelikan, M.: A Simple Implementation of Bayesian Optimization Algorithm in C++ (Version 1.0). Illigal Report 99011, February 1999, pp. 1-16.

[Pel99c] Pelikan, M., Goldberg, D. E., & Lobo, F.: A Survey of Optimization by Building and Using Probabilistic Model, Illigal Report 99018, September 1999, pp. 1-12.

[Pel00a] Pelikan, M., Goldberg, E., Sastry, K.: Bayesian Optimization Algorithm, Decision Graphs, and Occams Razor. Illigal Report No. 2000020, May 2000, pp.1-24.

[Pel00b] Pelikan, M., Goldberg, D. E., & Cantú-Paz, E.: Bayesian Optimization Algorithm, Population Sizing, and Time to Convergence, Illigal Report 200001, January 2000, pp. 1-13.

[Peli02] Pelikan, M.: Bayesian optimization algorithm: From single level to hierarchy. Ph.D. thesis, University of Illinois at Urbana-Champaign, Urbana, IL. Also IlliGAL Report No. 2002023, 2002.

[Sadi96] Sadic, M., Sait, Habib, Youssef: Iterative Computer Algorithms with Application in Engineering. IEEE Computer Society , ISBN 0-7695-0100-1.

[Sasa96] Sasao T., Fujita, M., editors: Representations of Discrete Functions. Kluwer Academic Publisher, London 1996.

[Shah91] Shahookar K., Mazumder P.: VLSI Cell Placement Techniques, ACM Computing Surveys, Vol. 23, No. 2, June 1991, pp. 195-220.

[Schn96] Schnecke V.: Hybrid Genetic Algorithm for Solving Constrained Packing and Placement Problems, PhD. Thesis, University of Osnabrueck, 1996, pp. 1- 186.

[Solv35] Premuim Solver V35: Dostupné na URL: http://evonet.dcs.napier.ac.uk/evoweb/resources/software/ software174.html.

[Stro99] Stroobandt, D.: Pin Count Prediction in Ratio Cut Partitioning for VLSI and ULSI. In M. A. Bayoumi, Editor, Proceedings of the IEEE International Symposium on Circuits and Systems, Pages VI/262-VI/265, May 1999.

[Weic91] Wei, Y.- C., Cheng, C.-K: Ratio Cut Partitioning for Hierarchical design, IEEE Trans. Computer Aided Design Integrated Circuit & System, Vol.10 (No.7), pp. 911-921, July 1991.

[Zitz99] Zitzler, E.: Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications, PhD thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, November 1999.

## 8 Přehled prací tvořící předloženou habilitační práci (skupina A)

A-1 Schwarz, J.: The optimum placement based on a method of limited cut. Computer and Artificial Intelligence, 5 (1986), No. 4, pp. 375-384.

A-2 Schwarz, J.: Experimental study on parallel genetic algorithm for placement optimalization. Proceedings of the 3rd International Mendel Conference on Genetic Algorithms, Optimization Problems, Fuzzy Logic, Neural Networks, Rough Sets, Mendel '97, Technical University of Brno, 1997, pp. 148-153, ISBN 80-214-0884-7.

A-3 Schwarz Josef, Očenášek Jiří: Experimental study: Hypergraph partitioning based on the simple and advanced genetic algorithm BMDA and BOA. Proceedings of the 5th International Conference on Soft Computing, Mendel '99, Technical University of Brno, 1999, pp. 124-130, ISBN 80-214-1131-7, (70/30%).

A-4 Schwarz Josef, Očenášek Jiří: A problem knowledge-based evolutionary algorithm KBOA for hypergraph bisectioning. Proceedings of the Fourth Joint Conference on Knowledge-Based Software Engineering, Brno, Czech Republic, 2000, pp. 51-58, ISBN 1-58603-060-4, (70/30%).

A-5 Schwarz Josef, Očenášek Jiří: Partitioning-oriented placement using advanced genetic algorithm BOA. Proceedings of the 6th International Conference on Soft Computing, Mendel 2000, Technical University of Brno, 2000, pp. 145-150, ISBN 80-214-1609-2, (70/30%).

A-6 Schwarz Josef, Očenášek Jiří: Evolutionary Multiobjective Bayesian Optimization Algorithm: Experimental Study. Proceedings of the 35th Spring International Conference MOSIS '01, Vol. 1, Hradec nad Moravicí, CZ, MARQ 2001, pp. 101-108, ISBN 80-85988- 57-7, (70/30%).

A-7 Schwarz Josef, Očenášek Jiří: Multiobjective Bayesian Optimization Algorithm for Combinatorial Problems: Theory and Practice. NEURAL NETWORK WORLD, roč. 11, č. 5, CZ, pp. 423-441, ISSN 1210-0552, (80/20%).

A-8 Schwarz Josef, Očenášek Jiří: Ratio cut hypergraph partitioning using BDD based MBOA optimization algorithm. Proceedings of the Fifth International Workshop IEEE Design and Diagnostics of Electronic Circuits and Systems, Brno University of Technology, Faculty of Information Technology & IEEE Computer Society, 2002, pp. 87-96, ISBN 80-214-2094-4, (70/30%).

A-9 Schwarz Josef, Očenášek Jiří: Bayes-Dirichlet BDD as a probabilistic model for logic function and evolutionary circuit decomposer. Proceedings of the 8th International Mendel Conference on Soft Computing, Brno, Mendel 2002, Technical University of Brno, 2002, pp. 117-124, ISBN 80-214-2135-5, (60/40%).

# 9 Seznam dalších prací autora

## 9.1 Seznam prací autora souvisejících s předloženou habilitační prací

B-1 Schwarz Josef: Algoritmy stochastické a simulované evoluce. Sborník kolokvia Vybrané problémy simulačních modelů, Ostrava, CZ, 1993, pp. 89-91, ISBN 80-901229-6-5.

B-2 Schwarz Josef: Simulation oriented placement methods for VLSI chips. Proceedings of MOSIS'93 conference, Olomouc, CZ, MARQ, 1993, pp. 295-300.

B-3 Schwarz Josef: Placement optimization using the genetic algorithm. Proceedings of MOSIS'94, Zábřeh na Moravě, CZ, MARQ, 1994, pp. 171-176, ISBN 80-901229-8-1.

B-4 Schwarz Josef: Simulation based placement algorithm. Proceedings of EDS '95 conference, Brno, CZ, 1995, pp. 217-228.

B-5 Schwarz Josef: PGPLACE: Genetic algorithm for placement optimization. Proceedings of the 1st International Mendel Conference on Genetic Algorithms, Mendel '95, Brno, 1995, Technical University of Brno pp. 139-144, ISBN 80-214-0672-0.

B-6 Schwarz Josef, Němec Viktor: Parallel genetic algorithms implemented on transputers. Proceedings of the 2nd International Mendel Conference on Genetic Algorithms, Optimization Problems, Fuzzy Logic, Neural Networks, Rough Sets. Technical University of Brno, Mendel '96, Brno, 1996, pp. 85-90, ISBN 80-214-0769-7, (70/30%).

B-7 Schwarz Josef: The FIDE system flexibility in the process of the fuzzy system design. Proceedings of MOSIS'96, Krnov, CZ, 1996, pp. 82-87, ISBN 80-85988-02-X.

B-8 Schwarz Josef: Educational fuzzy development systém. Proceedings of Conference MOSIS '97, Hradec nad Moravicí, CZ, MARQ, 1997, pp. 233-238, ISBN 80-85988-16-X.

B-9 Schwarz Josef: Case study: Genetic object designer. ASIS 1998, Krnov, CZ, MARQ, 1998, pp. 109-114, ISBN 80-85988-26-7.

B-10 Schwarz Josef: Genetic algorithm for partitioning circuits. Proceedings of the 4th International Mendel Conference on Genetic Algorithms, Optimization Problems, Fuzzy Logic, Neural Networks, Rough Sets, Technical University of Brno Mendel '98, Brno, 1998, pp. 126-131, ISBN 80-214-1199-6.

B-11 Schwarz Josef: Utilizing Genetic Algorithms for VLSI Physical Design-a Brief Survey. Electronic Devices and Systems 1999-Proceedings, BRNO, CZ, 1999, pp. 88-91, ISBN 80-214-1466-9.

B-12 Schwarz Josef, Očenášek Jiří: Pareto Bayesian Optimization Algorithm for the Multiobjective 0/1 Knapsack Problem. Proceedings of the 7th International Mendel Conference on Soft Computing, Brno, Mendel 2001, Technical University of Brno, pp. 131-136, ISBN 80-214-1894-X, (50/50%).

B-13 Schwarz Josef: The probability models for combinatorial optimization problems. Proceedings of the 4th Japan-Central Europe Joint Workshop on Energy and Information in Non-Linear Systems. Brno, Czech Republic, November 10-12, 2000, Brno, CZ, 2000, pp. 72-75.

B-14 Očenášek Jiří, Schwarz Josef: The Parallel Bayesian Optimization Algorithm, Proceedings of the European Symposium on Computational Inteligence, Košice, SK, Springer, 2000, pp. 61-67, ISBN 3-7908-1322-2, ISSN 1615-3871, (80/20%).

B-15 Očenášek Jiří, Schwarz Josef: The Distributed Bayesian Optimization Algorithm for Combinatorial Optimization. EUROGEN 2001 - Evolutionary Methods for Design, Optimisation and Control with Applications to Industrial Problems, Athens, GR, 2001, pp.115-120, ISBN 84-89925-97-6, (80/20%).

B-16 Očenášek, J., Schwarz, J.: Estimation Distribution Algorithm for mixed continuous-discrete optimization problems. Proceedings of 2nd Euro-International Symposium on Computational Intelligence, Kosice, Slovakia, IOS Press, 2002, pp. 227-232, ISBN 1-58603-256-9, ISSN 0922-6389, (80/20%).

B-17 Očenášek Jiří, Schwarz Josef: Development system DEBOA for rapid prototyping of evolutionary applications. Proceedings of International Conference MOSIS '02, Ostrava, CZ, MARQ, 2002, pp. 169-176, ISBN 80-85988-71-2, (80/20%).

## 9.2 Ostatní publikace, související částečně s řešenou problematikou

C-1 Eysselt, M., Schwarz, J.: Modelování prvků a podsystémů číslicových počítačů, KPOČ FE (ÚIVT FEI) VUT v Brně, CZ, Brno, 1975, s.1-13.

C-2 Schwarz, J., Eysselt, M.: Připojení tiskárny DZM180 k zařízení ZPD1200, KPOČ FE (ÚIVT FEI) VUT v Brně, CZ, Brno, 1976, s. 20.

C-3 Schwarz, J., Eysselt, M.: Připojení magnetopáskové jednotky k ZPD1200, KPOČ FE (ÚIVT FEI) VUT v Brně, CZ, Brno, 1976, s. 34, (50/50%).

C-4 Schwarz Josef: Programové prostředky pro návrh fuzzy systémů. Sborník kolokvia Vybrané problémy simulačních modelů, Brno, CZ, MARQ, 1994, pp. 11-14, ISBN 80-901229-9-X.

C-5 Dvořák Václav, Kunovský Jiří, Schwarz Josef, Zbořil František: Problem Solving with Parallel Processing. Proceedings of the Workshop'95, Praha, CZ, 1995, pp. 215-216, (25%).

C-6 Schwarz Josef: Programové prostředky pro fuzzy aplikace. Proceedings of Workshop of Advanced Simulation of Systems, Zábřeh na Moravě, CZ, 1995, pp. 258-264, ISBN 80-901751-7-1.

C-7 Schwarz Josef: Motorola microcontroller as the platform for fuzzy application. Proceedings of CSS '96, Brno, CZ, 1996, pp. 239-242, ISBN 80-214-0768-9.

C-8 Dvořák Václav, Schwarz Josef: Crosstalk analysis in multiconductor interconnect systém. Journal of Electrical Engineering, roč. 48, č. 1-2, SK, 1997, pp. 12-17, ISSN 0013-578X, (50/50%).

C-9 Schwarz Josef: Fuzzy trafic light controller. Proceedings of MOSIS'98, Bystřice pod Hostýnem, CZ, MARQ, 1998, pp. 67-73, ISBN 80-85988-25-9.

C-10 Schwarz Josef, Peringer Petr: Simulation Model of the Traffic Light Crossing with Fuzzy Control. Proceedings of MOSIS'99, Rožnov, CZ, MARQ, 1999, pp. 99-106, ISBN 80-85988-33-X, (50/50%).

## 9.3 Seznam prací studentů pod vedením autora habilitační práce

D-1 Pantůčková, J.: Propojovač dvouvrstvých spojů na bázi vlnového algoritmu. Diplomová práce, KPO FE VUT Brno, 1982.

D-2 Klimeš, M.: Kombinatorické metody rozmístění součástek na desce s plošnými spoji. Diplomová práce, KPO FE VUT Brno, 1983.

D-3 Hrbáček, M.: Optimalizace rozmístění IO na desce s plošnými spoji. Diplomová práce, KPO FE VUT Brno, 1983.

D-4 Hubálek, M.: Rozmisťování pouzder IO na desce s plošnými spoji. Diplomová práce, KPO FE VUT Brno, 1983.

D-5 Tajovský, M.: Lineární propojování. Diplomová práce, KPO FE VUT Brno, 1986.

D-6 Vičan, L.: Podsystém automatizovaného návrhu rozmístění integrovaných obvodů. Diplomová práce, KPO FE VUT Brno, 1988.

D-7 Obrusník , P.: Algoritmy rozmisťování s lineární časovou složitostí. Diplomová práce, KPO FE VUT Brno, 1988.

D-8 Ondroušek, J.: Optimalizace rozmístění pomocí metody simulovaného žíhání. Diplomová práce, UIVT FE VUT Brno, 1992.

D-9 Fiala, P.: Dekompoziční metody rozmisťování. Diplomová práce, UIVT FE VUT Brno, 1992.

D-10 Kahema, M.: Program pro rozmisťování součástek. Diplomová práce, UIVT FE VUT Brno, 1993.

D-11 Venkrbec, J.: Optimalizace rozmístění prvků pomocí metody simulované a stochastické evoluce. Diplomová práce, UIVT FE VUT Brno, 1993.

D-12 Hudec, T.: Simulace přeslechů na vázaných spojích. Diplomová práce, UIVT FE VUT Brno, 1993.

D-13 Kraus, J.: Fuzzy logika-návrh integrovaného vývojového prostředí. Diplomová práce, UIVT FEI VUT Brno, 1994.

D-14 Bartoš, J.: Genetické algoritmy pro optimalizaci rozmístění. Diplomová práce, UIVT FEI VUT Brno, 1994.

D-15 Kráľ, D. : Návrh univerzálního systému pro návrh aplikací s MH 68HC11. Diplomová práce, UIVT FEI VUT Brno, 1994.

D-16 Mičánek, D. : Fuzzy vývojový systém. Diplomová práce, UIVT FE VUT Brno, 1996.

D-17 Hroník, R.: Genetický algoritmus s fuzzy řídícím modulem. Diplomová práce, UIVT FEI VUT Brno, 1997.

D-18 Kaška, M.: Implementace fuzzy vývojového systému FREG/M2 na platformě OS Windows 95. Diplomová práce, UIVT FEI VUT Brno, 1997.

D-19 Čech, M.: Program pro návrh rozvrhu hodin. Diplomová práce, UIVT FEI VUT Brno, 1998.

D-20 Bláha, P. Návrh genetického algoritmu pro alokaci funkčních bloků na VLSI čipu. Diplomová práce, UIVT FEI VUT Brno, 1998.

D-21 Kříž, L.: Návrh výukového vývojového systému pro rychlý návrh genetických algoritmů. Diplomová práce, UIVT FEI VUT Brno, 1999.

D-22 Švehlák, L.: Návrh genetického algoritmu pro fyzický návrh VLSI obvodů. Diplomová práce, UIVT FEI VUT Brno, 1999.

D-23 Očenášek, J.: Návrh pokročilého genetického algoritmu. Diplomová práce, UIVT FEI VUT Brno, 1999.

D-24 Kučera, P.: Návrh výukového fuzzy vývojového systému. Diplomová práce, UIVT FEI VUT Brno, 1999.

## 10 Příloha A – Xerokopie článků tvořících předloženou habilitační práci

A-1 Schwarz, J.: The optimum placement based on a method of limited cut. Computer and Artificial Intelligence, 5 (1986), No. 4, pp. 375-384.

A-2 Schwarz, J.: Experimental study on parallel genetic algorithm for placement optimalization. Proceedings of the 3rd International Mendel Conference on Genetic Algorithms, Optimization Problems, Fuzzy Logic, Neural Networks, Rough Sets, Mendel '97, Technical University of Brno, 1997, pp. 148-153, ISBN 80-214-0884-7.

A-3 Schwarz Josef, Očenášek Jiří: Experimental study: Hypergraph partitioning based on the simple and advanced genetic algorithm BMDA and BOA. Proceedings of the 5th International Conference on Soft Computing, Mendel '99, Technical University of Brno, 1999, pp. 124-130, ISBN 80-214-1131-7, (70/30%).

A-4 Schwarz Josef, Očenášek Jiří: A problem knowledge-based evolutionary algorithm KBOA for hypergraph bisectioning. Proceedings of the Fourth Joint Conference on Knowledge-Based Software Engineering, Brno, Czech Republic, 2000, pp. 51-58, ISBN 1-58603-060-4, (70/30%).

A-5 Schwarz Josef, Očenášek Jiří: Partitioning-oriented placement using advanced genetic algorithm BOA. Proceedings of the 6th International Conference on Soft Computing, Mendel 2000, Technical University of Brno, 2000, pp. 145-150, ISBN 80-214-1609-2, (70/30%).

A-6 Schwarz Josef, Očenášek Jiří: Evolutionary Multiobjective Bayesian Optimization Algorithm: Experimental Study. Proceedings of the 35th Spring International Conference MOSIS '01, Vol. 1, Hradec nad Moravicí, CZ, MARQ 2001, pp. 101-108, ISBN 80-85988- 57-7, (70/30%).

A-7 Schwarz Josef, Očenášek Jiří: Multiobjective Bayesian Optimization Algorithm for Combinatorial Problems: Theory and Practice. NEURAL NETWORK WORLD, roč. 11, č. 5, CZ, pp. 423-441, ISSN 1210-0552, (80/20%).

A-8 Schwarz Josef, Očenášek Jiří: Ratio cut hypergraph partitioning using BDD based MBOA optimization algorithm. Proceedings of the Fifth International Workshop IEEE Design and Diagnostics of Electronic Circuits and Systems, Brno University of Technology, Faculty of Information Technology & IEEE Computer Society, 2002, pp. 87-96, ISBN 80-214-2094-4, (70/30%).

A-9 Schwarz Josef, Očenášek Jiří: Bayes-Dirichlet BDD as a probabilistic model for logic function and evolutionary circuit decomposer. Proceedings of the 8th International Mendel Conference on Soft Computing, Brno, Mendel 2002, Technical University of Brno, 2002, pp. 117-124, ISBN 80-214-2135-5, (60/40%).

Schwarz, J.:

## The optimum placement based on a method of limited cut.

A-1

Computers and Artificial Intelligence, 5 (1986), No. 4, 375—384

# THE OPTIMUM PLACEMENT BASED ON A METHOD OF LIMITED CUTS

Josef SCHWARZ
*Department of Computer Science, Electrotechnical Faculty, Technical University in Brno, Božetěchova 2, 612 66 Brno, Czechoslovakia*

**Abstract.** The paper deals with an optimum placement of integrated circuits to discrete locations of printed circuit board with a goal to simplify the following routing phase. A method of limited cuts is suggested which results in optimization of wire density distribution as well as in minimization of the total wire length.

**Оптимизация размещения интегральных схем на печатной плате на основе метода лимитных сечений**

Й. Шварц

**Резюме.** В статье описывается метод оптимизации размещения интегральных схем в дискретных позициях печатной платы с целью достижения возможно простой реализации последующего этапа соединения. Предлагается метод лимитных сечений, приводящий к оптимизации размещения плотности соединений и одновременно к минимизации общей длины соединений.

## 1. INTRODUCTION

The placement problem consists in finding optimal locations for the set of interconnected objects and often appears in solutions of many technical tasks.

This paper deals with the component placement (e.g. integrated circuits) into discrete locations on a printed circuit board (PCB). Note that the PCB design incorporates three stages — partitioning, placement and routing.

The automation level of individual stages in the PCB design is different at present. Whereas in Czechoslovakia several program systems [1, 2, 3] for the routing of two- or multilayer PCB are available, little attention has been paid to the automation of partitioning and placement.

As the complex system of an automated physical design has not yet been realized, all tasks of partitioning and placement of integrated circuits have been solved intuitively in majority of cases.

## 1.1. Placement problem definition

The placement problem solved in this paper is based on the following assumptions: A logic scheme of a board (E, S) is given, characterized by a set of elements $E=\{e_1, e_2, \ldots, e_n\}$ and a set of nets $S=\{S_1, S_2, \ldots, S_t\}$ such that $S \subset \mathcal{P}(E)$, where $\mathcal{P}(E)$ is a power set. Further, a set of locations is given $P=\{p_1, p_2, \ldots, p_m\}$ where $m \geq n$.

Then by the placement problem we mean finding a one-to-one mapping $f_R: E \rightarrow P$ such that some objective is optimized.

The most frequent objective used is to place the elements so as to minimize the total wire length. To determine the total wire length various approximations of the net length are used — usually the minimum tree length or one half-perimeter of the minimal rectangle enclosing elements of the net (the so-called minimal half-perimeter method). However, the used criterium of the minimal total wire length does not seem to be satisfying in a PCB design with great density of elements and wires. In general, the minimal total wire length does not exclude the existence of the so-called critical board areas with excessive wire density (e.g. in the middle of the board or in the connector area) [4, 5].

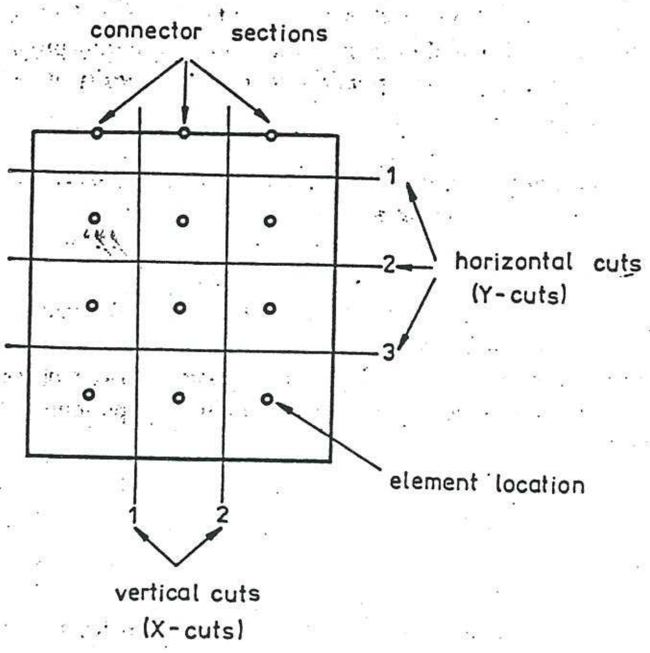


Fig. 1. Board model with canonical cuts.

## 1.2. Cut methods

Application of methods of canonical cuts is an efficient tool for detection and limitation of wire congestion in local board areas. A simple board model is used with regular structure of point locations — a system of $r$ rows and $z$ columns is produced with a unit distance between adjacent locations (Fig. 1).

The set of canonical cuts C is topologically defined as a collection of cut lines between adjacent rows and columns of locations. The set of cuts C can be subdivided into a subset of vertical cuts $C_x$ — so-called $X$ cuts, passing between adjacent columns of locations, and into a subset of horizontal cuts $C_y$ — so-called $Y$ cuts, passing between adjacent rows of locations. The number of vertical and horizontal cuts is denoted by $z'$ and $r'$, respectively, where $r'=r-1$ and $z'=z-1$, respectively. Let us analyse a vertical or horizontal cut dividing the interconnecting board area into two parts. With respect to the fact that these cuts do not cross any element, it is obvious that the considered cut divides the set of elements E into two disjunct subsets E′, E″. Any net which contains elements in both E′ and E″ requires one wire to connect its elements in E′ to its elements in E″.

A basic attribute of the cut is the cut value $v(c)|c \in C$ given by the number of nets crossing this cut. Another attribute is the cut throughput W, determining maximal number of wires crossing the cut that can be realized with a given technology and board parameters.

It is clear that when $v(c)>W$, the interconnection of elements on a board cannot be realized. Therefore, we shall try to find such a decomposition of E into two subsets that would minimize the total wire number between these subsets.

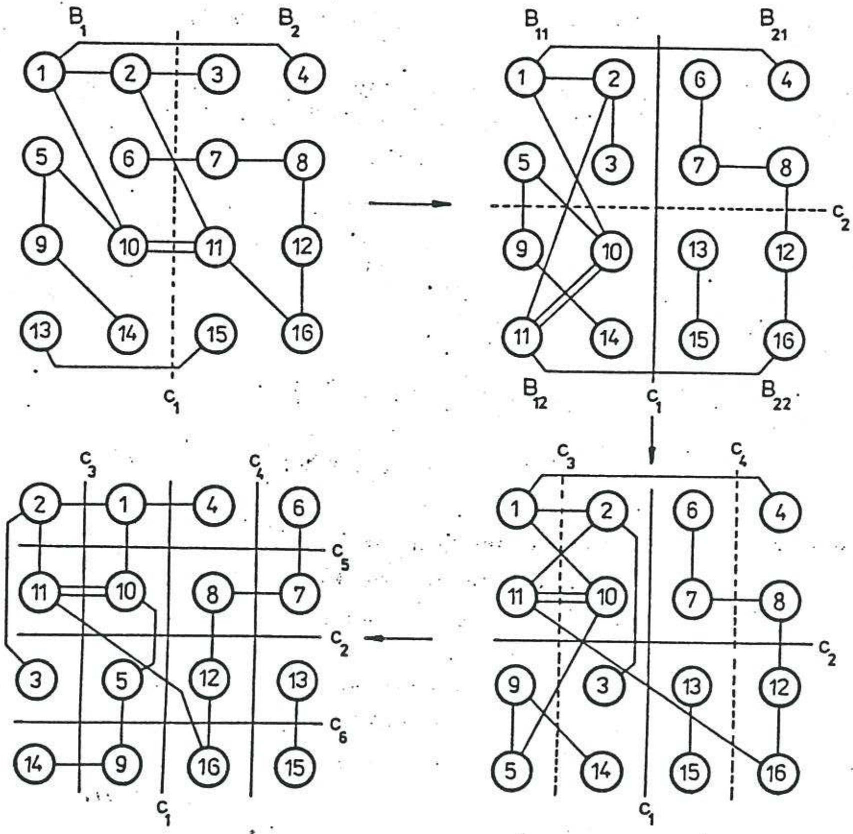


Fig. 2. Illustration of the quadrature placement.

A key problem, i.e. in what order and how to minimize the value of cuts, has been solved in the paper of M. A. BREUER [6]. The author suggested three fixed sequential min-cut placement algorithms, differing from each other in a cut sequence type. The most used algorithm is the quadrature placement illustrated for a 16-element task in Fig. 2.

The interconnecting board area is first bisected by a vertical cut $c_1$, producing two new blocks $B_1$ and $B_2$. Elements from E are assigned to the blocks $B_1$ and $B_2$ so that the cut value $v(c_1)$ is minimal. Then a horizontal cut $c_2$ follows, bisecting the block $B_1$ into blocks $B_{11}$, $B_{12}$ as well as $B_2$ into blocks $B_{21}$, $B_{22}$. Elements from the block $B_1$ are assigned to blocks $B_{11}$, $B_{12}$, whereas elements from $B_2$ are assigned to blocks $B_{21}$, $B_{22}$ and again with the minimal cut value $v(c_2)$.

These four blocks are further bisected by vertical and horizontal cuts alternatively. The whole process is finished as soon as none of the blocks contains more than one element. The use of this block structure offers the possibility to keep the values of previously minimized cuts unchanged. Although the above method of minimal cuts offers the possibility of decreasing the absolute wire density, it has no direct relation to the uniformity of wire density distribution. In the following chapter, the problem of more uniformly distributed wires on a board is solved by means of a new objective function of a total excess value of cuts.

## 2. APPLICATION OF A NEW OBJECTIVE FUNCTION CF [7]

A newly suggested objective function CF of a total excess value of cuts is given by the sum of nets crossing the cuts exceeding a predetermined limit:

$$CF = CF_x + CF_y \tag{1}$$

$$CF_x = \sum_{\substack{c \in C_x \\ v(c) > L_x}} (v(c) - L_x) \tag{2}$$

$$CF_y = \sum_{\substack{c \in C_y \\ v(c) > L_y}} (v(c) - L_y) \tag{3}$$

where

$CF_x$ is a component of the objective function for $X$ cuts,
$CF_y$ is a component of the objective function for $Y$ cuts,
$L_x$ and $L_y$ are limits for $X$ and $Y$ cuts, respectively.

The value of limits is always smaller than the cut throughput W. The advantage of the above objective function lies in its close relation to the total wire length and wire distribution on a board characterized by an average cut deviation.

### 2.1. CF relation to the total wire length

For a classical board model (Fig. 1) with a unit distance between adjacent element locations it holds that the total wire length H, determined by the minimal half-perimeter method equals the sum of canonical cut values R [6]:

$$H = R = \sum_{c \in C} v(c). \tag{4}$$

The sum of canonical cut values, and hence the total wire length, can also be expressed by an average value of $X$ and $Y$ cuts designated by

$$\bar{v}_x, \bar{v}_y \left(\bar{v}_x = \frac{1}{z'} \sum_{c \in C_x} v(c);\ \bar{v}_y = \frac{1}{r'} \sum_{c \in C_y} v(c)\right):$$

$$\mathrm{H} = \mathrm{R} = \mathrm{R}_x + \mathrm{R}_y = \bar{v}_x \,.\, z' + \bar{v}_y \,.\, r', \tag{5}$$

where $\mathrm{R}_x$ and $\mathrm{R}_y$ are sums of the values of $X$ and $Y$ cuts, respectively.

Now, let us assume that we are given an initial placement characterized by values ($\mathrm{CF}_x$, $\mathrm{CF}_y$, $\bar{v}_x$, $\bar{v}_y$, $\mathrm{L}_x$, $\mathrm{L}_y$, H) and an optimized placement characterized by values ($\mathrm{CF}'_x$, $\mathrm{CF}'_y$, $\bar{v}'_x$, $\bar{v}'_y$, $\mathrm{L}_x$, $\mathrm{L}_y$, H'). Our goal is to state conditions for decreasing the total wire length of an optimized placement in dependence on the value of the objective function CF'. Knowing the value of the objective function CF', one can derive from (1) (2) (3) (4) the upper bound of the total wire length:

$$\mathrm{H}'_h = \mathrm{CF}' + \mathrm{L}_x \,.\, z' + \mathrm{L}_y \,.\, r' \tag{6}$$

$$\mathrm{H}'_h \geqslant \mathrm{H}'. \tag{7}$$

From equations (5) (6) and inequality (7) it is obvious that a sufficient condition for decreasing the total wire length is the validity of the inequality $\mathrm{H}'_h < \mathrm{H}$.
One can distinguish between two basic cases:

a) $(\mathrm{L}_x \geqslant \bar{v}_x) \wedge (\mathrm{L}_y \geqslant \bar{v}_y)$

With zero value, which is the minimal value of the objective function CF', from (5) (6) it follows that

$$\mathrm{H}'_h = \mathrm{L}_x \,.\, z' + \mathrm{L}_y \,.\, r' \geqslant \mathrm{H}. \tag{8}$$

With the mentioned values of limits the decrease of the total wire length cannot be guaranteed even with zero value of the objective function CF'.

b) $(\mathrm{L}_x < \bar{v}_x) \wedge (\mathrm{L}_y < \bar{v}_y)$

In this case the total wire length decreases with zero value of the objective function

$$\mathrm{H}'_h = \mathrm{L}_x \,.\, z' + \mathrm{L}_y \,.\, r' < \mathrm{H}. \tag{9}$$

However, the wire length also decreases with a certain non-zero value of the objective function. If it is to hold that $\mathrm{H}'_h < \mathrm{H}$, we use (5) (6) to get

$$\mathrm{L}_x \,.\, z' + \mathrm{L}_y \,.\, r' + \mathrm{CF}' < \bar{v}_x \,.\, z' + \bar{v}_y \,.\, r'. \tag{10}$$

The condition for CF' follows from (10)

$$\mathrm{CF}' < (\bar{v}_x - \mathrm{L}_x) \,.\, z' + (\bar{v}_y - \mathrm{L}_y) \,.\, r' \tag{11a}$$

or alternatively

$$(\mathrm{CF}'_x < (\bar{v}_x - \mathrm{L}_x) \,.\, z') \wedge (\mathrm{CF}'_y < (\bar{v}_y - \mathrm{L}_y) \,.\, r'). \tag{11b}$$

## 2.2. CF relation to the wire distribution

An ideal element placement can be characterized by equal values of canonical cuts close to the value of limits $\mathrm{L}_x$ or $\mathrm{L}_y$. The goal of real placement optimization lies in minimizing the value differences of individual cuts and thereby in achieving a more uniform wire distribution on a board. A standard indicator of a cut value dispersion is an average deviation of cuts or a standard deviation of cuts.

We are going to show that the objective function CF is in a close relation to the average deviation of cuts. The components of the average deviation of cuts D are expressed by means of the positive deviations of cuts:

$$D_x = \frac{1}{z'} \sum_{c \in C_x} |v(c) - \bar{v}_x| = \frac{2}{z'} \sum_{\substack{c \in C_x \\ v(c) > \bar{v}_x}} (v(c) - \bar{v}_x) \tag{12}$$

$$D_y = \frac{1}{r'} \sum_{c \in C_y} |v(c) - \bar{v}_y| = \frac{2}{r'} \sum_{\substack{c \in C_y \\ v(c) > \bar{v}_y}} (v(c) - \bar{v}_y) \tag{13}$$

$$D = (D_x + D_y)/2. \tag{14}$$

From equations (12), (13), (2), (3) it follows that with the choice of limits $L_x = \bar{v}_x$ and $L_y = \bar{v}_y$, the components of the average deviation $D_x$, $D_y$ are proportional to components of the objective function CF:

$$D_x = CF_x \,.\, 2/z' \tag{15}$$

$$D_y = CF_y \,.\, 2/r' \tag{16}$$

$$D = CF_x/z' + CF_y/r'. \tag{17}$$

Minimization of components of the objective function CF is thus a prerequisite of minimization of the average deviation of cuts D. The values of limits are usually chosen so that $(L_x < \bar{v}_x) \wedge (L_y < \bar{v}_y)$, which is a sufficient condition for the decrease of the total wire length, too.

In this case, for optimized placement it holds:

$$D'_x \leq CF'_x \,.\, 2/z' \tag{18}$$

$$D'_y \leq CF'_y \,.\, 2/r' \tag{19}$$

provided that $(\bar{v}'_x \geq L_x) \wedge (\bar{v}'_y \geq L_y)$.

In case that the decrease of the average deviation of cuts should take place and consequently $D'_x < D_x \wedge D'_y < D_y$ should hold, a sufficient condition is the validity of the following relations:

$$CF'_x \,.\, 2/z' < D_x \tag{20}$$

$$CF'_y \,.\, 2/r' < D_y, \tag{21}$$

that is

$$CF'_x < D_x \,.\, z'/2 \tag{22}$$

$$CF'_y < D_y \,.\, r'/2. \tag{23}$$

Conditions for simultaneous minimization of the total wire length H, and the average deviation D follow from equations (11b), (22), (23):

$$CF'_x < \min\left[(\bar{v}_x - L_x) \,.\, z', D_x \,.\, z'/2\right] \tag{24}$$

$$CF'_y < \min [(\bar{v}_y - L_y) . r', D_y . r'/2]. \tag{25}$$

## 3. METHOD OF LIMITED CUTS

For minimization of the objective function CF an iterative method of limited cuts is proposed. This method is characterized by a controlled optimization of a cut sequence — only those cuts are minimized the value of which exceeds a predetermined limit $L_x$ or $L_y$. Minimization of a cut is interrupted as soon as the cut value lies in the given tolerance interval of the limits. During one optimizing cycle the *i*-th cut sequence is optimized $(c_1^i, c_2^i, \ldots, c_j^i)$, where $c_1^i$ is the first cut of the *i*-th sequence selected for optimization. The cut order in the sequence is always determined dynamically on the basis of actual values of cuts.

First the cut $c_1^i$ is optimized with maximal cut value exceeding the value of the limit. The optimization being finished, the cut $c_2^i$ is selected for optimization from all remaining cuts, and again with maximal cut value. This process is repeated for all remaining cuts $c_3^i \div c_j^i$. It is important that in optimizing an actual cut the results of optimization of previous cuts are maintained. The mentioned dependency of cut optimization is represented by a block structure in the same way as in [6].

### 3.1. ALKAR program

The ALKAR program allows a cyclic placement optimization of integrated circuits on a board with regular structure of locations, using the method of limited canonical cuts. It consists of five basic steps:

1. *Initialization*
a) pseudorandom generation of initial placement,
b) setting the values of limits $L_x$, $L_y$,
c) determination of the strategy for alternation of vertical and horizontal cuts,
d) specification of the cut fixation and number $N_c$ of optimization cycles,
e) setting the cycle optimization variable ($i := 1$).

2. *Optimization of the i-th cut sequence*
Dynamic selection of the *i*-th cut sequence and its minimization towards the value of limits $L_x$ and $L_y$ is performed. For cut minimization the modification of Kernighan-Lin algorithm of pair interchanges with selectable efficiency grade and minimization process rate is used.

3. *Cut fixation*
After finishing optimization of the *i*-th sequence, the cuts are determined for fixation the value of which lies in the given tolerance interval of limits $L_x$ and $L_y$.
All values of fixed cuts remain maintained during the following optimization cycle. Fixation may be realized either immediately, after the first optimization cycle, or with a delay, after the $k^*$-th cycle: $1 \leq k^* < N_c$.

4. *Block structure actualization*
Actualization of the block structure of a board, given by fixed cuts, takes place before every

new optimization cycle; the block structure for the $i$-th cycles ($i<k^*$) is trivial — it is represented by the whole board.
Steps 2 — 4 are repeated for all $N_c$ cycles.

5. *Resulting placement*
The program output is the placement with a minimal value of objection function CF gained in the $k$-th cycle such that $1<k\leq N_c$.

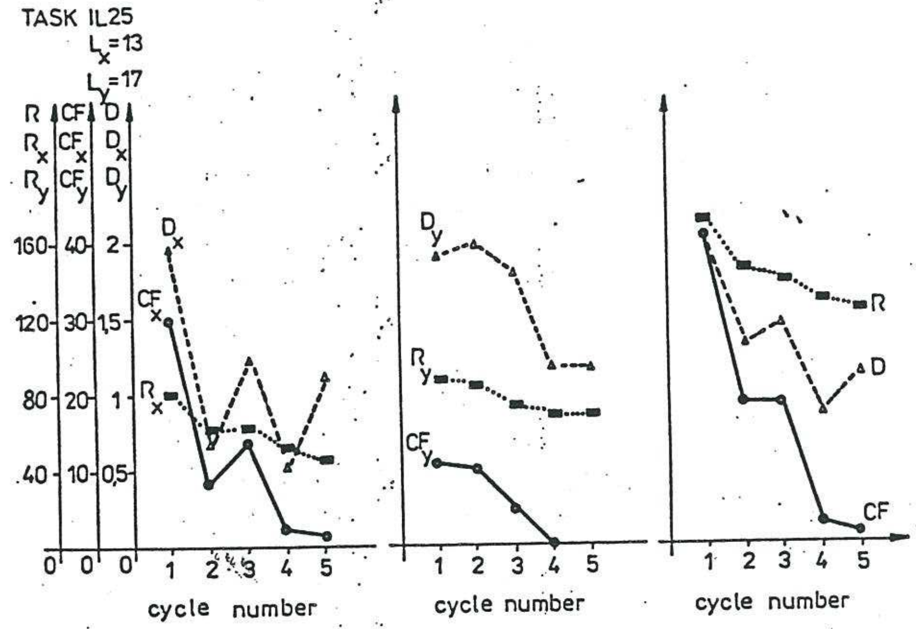


Fig. 3. Graphs of functions CF, R, D and their components.

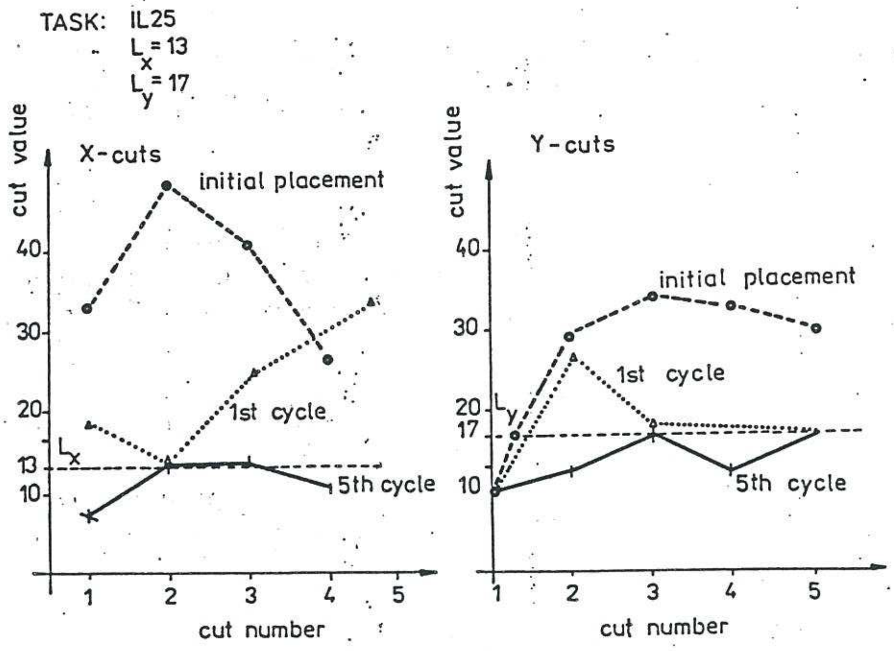


Fig. 4. Diagram of cut values.

### 3.2. Experimental results

The first version of the ALKAR program was verified on the set of ten test tasks, differing in the number of elements ($n = 16 \div 106$), in the number of nets ($t = 16 \div 286$) and in the complexity of relationships between elements. One half of test tasks belongs to the so-called classical test problems, used for effectivity comparison of different placement methods.

The results of experiments contain:

a) resulting placement with pseudointerconnection

b) graphs of CF, D, R functions and their components

c) diagram of cut values

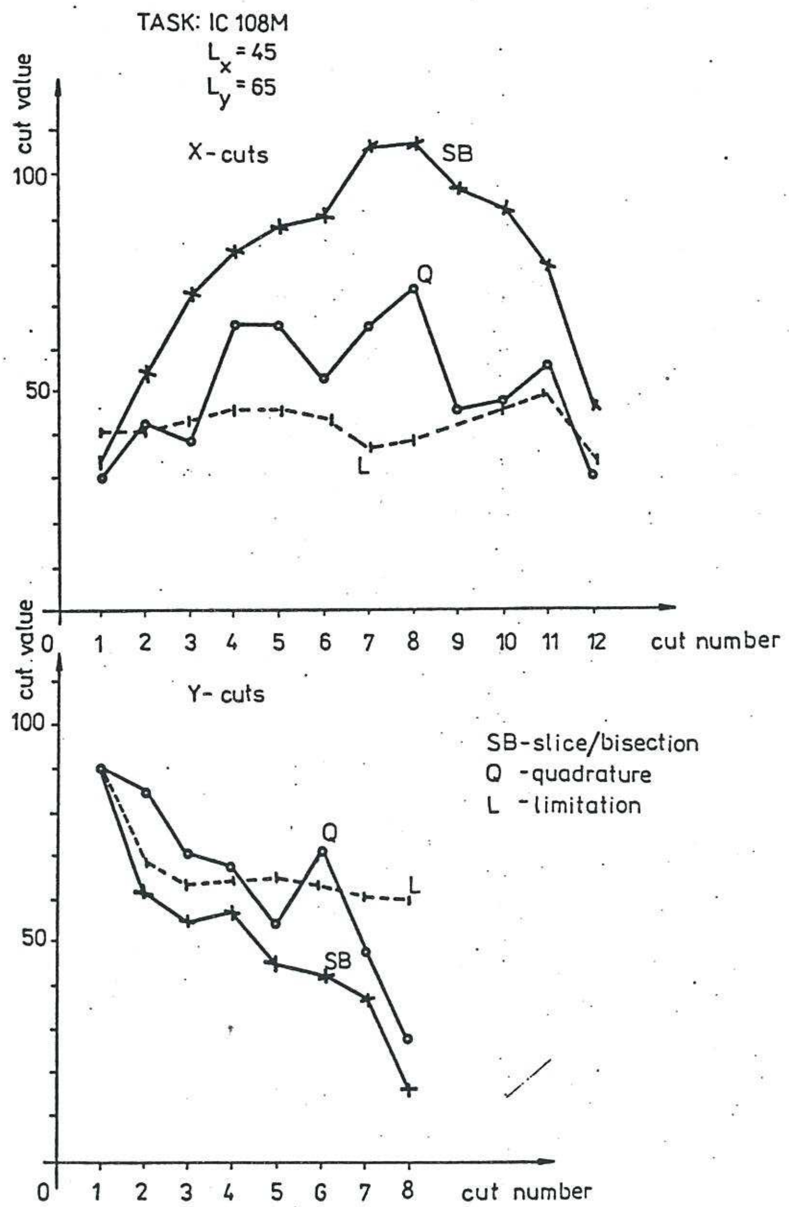


Fig. 5. Comparison of wire distribution graphs produced by the standard min-cut placement procedure (SB, Q) and program ALKAR (L).

d) mean time of the cycle.

Figs. 3 and 4 illustrate intermediate results (graphs of functions and wire distribution) for the task IL25 with 25 elements and 65 wires.

Fig. 5 shows the comparison between efficiency of the ALKAR program and standard min-cut placement procedure [6].

From the analysis of results given in [7] it follows that the ALKAR program makes possible effective solving of the problem of wire congestion on a board simultaneously with minimization of the total wire length.

The computation complexity is not greater than $O(n^{5/2})$. The mean time of an optimization cycle is 3 min. for the task IC67 and 4.5 min. for the task IC 108M ($n=67$ and 106, respectively). The ALKAR program is written in PASCAL and runs on the EC 1025 computer under the operation system DOS IV, occupying cca 200 kbytes of the main memory.

## 4. CONCLUSION

In the paper a new objective function of the total excess value of cuts was discussed that has close relation to both the total wire length and wire distribution on a board.

Optimization of the mentioned objective function leads to such a placement of elements that makes the following interconnection task easier.

The algorithm of limited cuts, ALKAR, was proposed for optimization of the objective function. At present, this algorithm handles up to 200 elements. There is a real assumption that the range of soluble tasks will increase using the published procedures [8].

The ALKAR program is part of the AKR system [7] that was designed to solve both the component placement and partition problem. The algorithms used in the system are based on the modified methods of pair exchanges, relaxation methods, and the method using linear programming techniques.

## REFERENCES

[1] PŘIBÁŇ, M.: Program system for the computer design automation. AVT 1/75, VÚMS Prague pp. 17—22 (in Czech).
[2] JANKŮ, A.: PCB design automation. PhD Thesis, ČVUT FEL, Prague 1982, 94 pp. (in Czech).
[3] SERVÍT, M.—FRIŠ, Z.: Computer aided PC design. ČSVTS-FEL, Prague 1978, 88 pp. (in Czech).
[4] SERVÍT, M.: Prerouting analysis of the printed circuit boards. Comp. Aid. Design, 13, 1981, No. 6, pp. 367—375.
[5] KAREH, A. H.: Topological design of a complex printed circuit. PhD Thesis, University of California, Berkeley 1978, 128 pp.
[6] BREUER, M. A.: Min-cut placement. J. of Design Automation and Fault Tolerant Computing, Vol. 1, 1977, No. 4, pp. 343—362.
[7] SCHWARZ, J.: Design automation of partitioning and integrated circuits placement. Ph. D. Thesis, VUT Brno, 1983, 136 pp. (in Czech).
[8] FIDUCCIA, C. M.—MATTHEYSES, R. M.: A linear time heuristic for improving network partitions. IEEE Design Automation Conference, 1982, pp. 175—181.

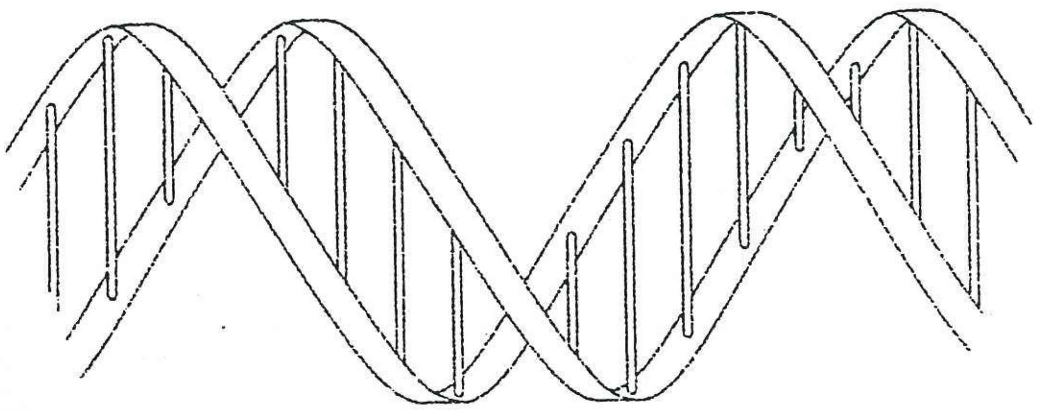
Received September 29, 1985 — revised version January 27, 1986

**Schwarz, J.:**

**Experimental study on parallel genetic algorithm for placement optimalization.**

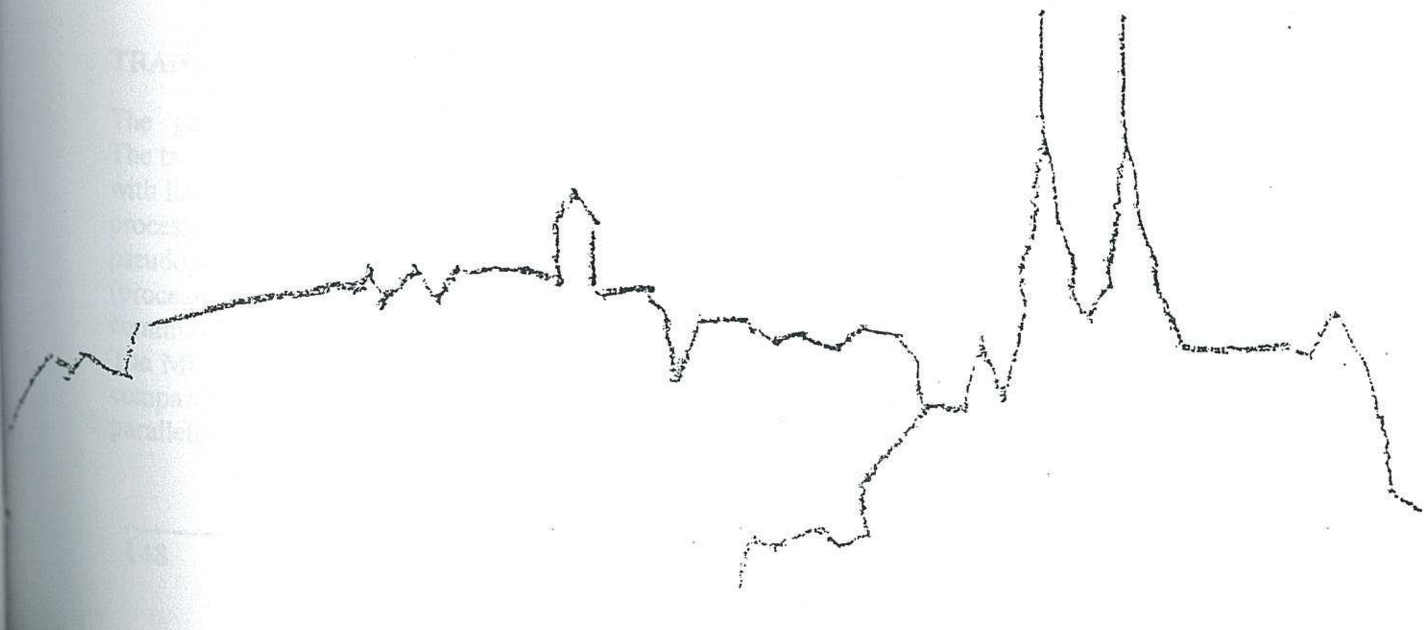
A-2

TECHNICAL UNIVERSITY OF BRNO
FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE



# MENDEL '97

3rd International Mendel Conference on Genetic Algorithms, Optimization problems, Fuzzy Logic, Neural networks, Rough Sets



June 25-27, 1997, Brno, Czech Republic

# Experimental study on parallel genetic algorithm for placement optimalization

Josef Schwarz

Technical University of Brno
Faculty of Engineering and Computer Science
Department of Computer Science and Engineering
CZ - 61266 Brno, Bozetechova 2
e-mail: schwarz@dcse.fee.vutbr.cz

**Abstract:** This paper is an experimental study on a course-grain parallel genetic algorithm for solving the placement problem on the level of the printed circuit board (PCB) design. It can be proved that this is NP problem. The placement problem is defined as a mapping of the hypergraph nodes into the regular structure of locations. A CPGEN program was created in C language, which enables wide class experiments via extensive menu for setting main parameters of genetic optimization. The main attention was paid to the adaptation of main genetic operators including the migration operator, mutation and heuristic procedure. To test CPGEN algorithm the variety of experiments was done.

**Key words:** placement, optimalization, genetic operators, parallel genetic algorithm, transputers

## INTRODUCTION

Placement is a crucial task in the layout of VLSI circuits and PCBs. It has a great impact on the cost and routability [1]. It can be characterizied by an assignment of circuit elements(nodes) to locations on a chip or PCB. Let $E=\{e_1,....e_n\}$ be a set of circuit elements and set of nets, $S=\{s_1, ....,s_k\}$, where a net is a set of elements to be interconnect. A set of locations $L=\{l_1,...l_n\}$ is given. The locations (slots) are organized upon the layout styles. To simplify the problem the set of slots forms a regular structure of locations with *c* columns and *r* rows with unit distance of neighbour location. The objective function is minimal total sum of wires, which realize the nets. To precise the problem the hypergraph is used for the representation of the circuit and the position graph is used for the slots representation.

## TRANSPUTER PLATFORM

The genetic algorithms have been implemented on transputer station T800 and T9000 [2]. The transputer is nowadays a well known microcomputer subsystem with its own local memory and with links for connecting one transputer to another. It is possible to execute unlimited number of parallel processes on one transputer because of its internal architecture (in this case we will obtain a pseudoparallel structure). A transputer network can be built by interconnecting four transputers (processor nodes) via links. Processes executed on different nodes of this network run in parallel and can communicate mutually through channels.
The Microsoft C programming language is used for programming the transputers. The language C in comparison to OCCAM allows user to write program core easily but resides a weakness in parallelization and greater communication costs.

## CPGEN IMPLEMENTATION

A genetic algorithm in general has three main parts: initialization (creation of initial subpopúlation - this step is made only once at the beginning of the algorithm), reproduction and evaluation (these two steps are repeated in a loop). There can be more approaches to parallelization of such algorithm.
We have used the coarse - granularity approach which brings a new aspect to the decomposition of the genetic algorithms. The main idea lies in dividing the *population* into more *subpopulations* which exchange genetic material in certain intervals.
The distributed placement algorithm/procedure is in fact a standard sequential genetic algorithm. This algorithm evaluates and reproduces certain amount of individuals in sequential way. These procedures are mapped onto a transputer network and executed in parallel. The procedures form a ring topology which is deadlock free. A new genetic operator *migration* is introduced. Migration transfers genetic individuals from one environment to another that improves the convergence of the optimization process.
In the following section the main featúres of the parallel genetic procedure[3] will be described:

### CPGEN procedure

```
{KEY SYMBOLS:
$P_s$ - subpopulation, $N_p$ - size of subpopulation
p - number of processes
$P_o$ - offspring , $N_o$ - size of offspring
L - local best idividual/placement
G - global best individual/placement
$R_{mp}$, $R_{mo}$, $R_{mig}$ - mutation rate of parents, offspring and migration rate}

Initialize; setting the parameters $N_p$, $N_o$, $R_{mo}$, $R_{mp}$, $R_{mig}$
Generate($P_s$); initial subpopulation $P_s$ randomly generated
While stopping criterion is false do
begin
Evaluate($P_s$); fitness function
Offspring generation and mutation;
Parents mutation;
Evaluate($P_o$); fitness function
Selection of the new subpopulation;
Heuristic procedure for the local optimization;
Migration;among subpopulatins
end
Final local optimization(G);
Output(G);
```

#### *Subpopulation and offspring size*

The subpopulation size $N_p$ and offspring size $N_o$ determines processing time and resulting quality. The both parameters are provided by the user. The parameter $N_p$ and $N_o$ are considered to be from 5 to 240 according to number of processes to be used. The suitable number of generated offspring $N_o$ is about 30% of subpopulation size [4].

#### *Fitness function*

The fitness function used in the procedure *Evaluation* is based on the classical objective function of the minimal total wire length. The length of wires is estimated by the minimal semiperimeter of rectangle enclosing elements of each net (MSP). It is easy to compute and it is often used as standard measure of the net complexity. In Fig. 1 is shown the MSP and MSP (the minimal spanning tree) model as well. It is clear that the MSP is mostly lower than the MST.

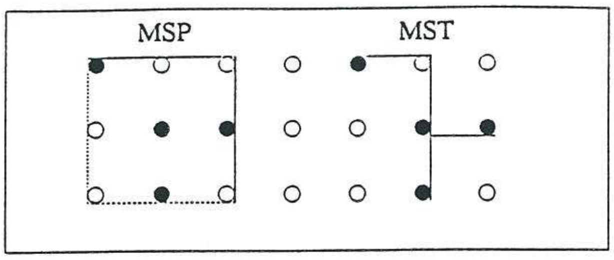


Fig. 1 Estimation of the wire length using the MSP and MST model.

It can be proved that the sum of semiperimeters equals the sum of cut values (number of nets crossing the regular strucure of horizontal and vertical cuts that are placed between neighbouring columns and rows) used as a criterion in decomposition algorithms (Breuer mincut algorithm). In case of standard cell placement problem it would be necessary to extend the simple objective function with penalty subfunction to effect the overlapping of standard cells.

The fitness function is based on the objective function represented by the MSP model. The total MSP wire length (cost) of each individual is computed first. The fitness function of each individual is then computed as the division of the global cost value of the whole subpopulation by the cost of the individual. The smaller the individual cost is the higher is its fitness function.

### *Offspring generation*

The traditional crossover operator which is used typically for the bit string is not suitable for the placement problem because it produces irregular configuration (conflict crossing) very often. Instead, the powerful conflictless operator called PMX (Partially Map Crossover) [5] is used.

### *Mutation*

It is possible to use two mutation operators together or separately. Both of them are based upon the classical random exchange of nodes. The number of nodes to be exchanged is fixed to two, but it can be expanded to five. The first mutation operator is used for parents mutation. The mutation rate $R_{mp}$ is defined as percentage of the total number of symbols/nodes in the subpopulation, that are mutated in each generation. For the n - node/element placement problem and for the size $N_p$ of the subpopulation $nN_pR_{mp}/2$ pairwise interchanges are performed. Typically average value $R_{mp}= 0.02$. The mutation rate can be set dynamically using some special procedure called AMR which allows to scan saturation level of the fitness function of the whole population. The saturation of a subpopulation means the degradation of individual diversity. The second mutation operator is used to a group of offspring.
It has been also implemented a special fuzzy program module which enables to control the mutation rate on its actual value, saturation level of the current subpopulation and the cost difference for the near-by generations. This module was developed on the Borland C language platform using the fuzzy inference design environment (FIDE)[6] and tested in an extra sequential genetic algorithm. It cannot be implemented on the transputer platform because the target C code is not compatible with the C version used in transputer station.
The block diagram of the fuzzy module is in Fig. 2. The operational unit includes a procedure with optional delay added to FIU output.

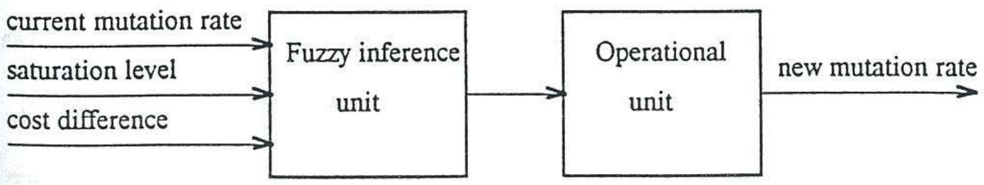


Fig.2. Block diagram of the fuzzy module.

### *Heuristic procedure*

Some heuristic approach is used to improve the convergence speed of the optimization process. It can be understood as a hybrid optimization. It is based on the reconfiguration of some individuals of the current subpopulation using the pairwise interchange of nodes with cost decrement. The procedure is activated seldom only during an epoch formed by a number of generations. It is possible to set the death and rate of the reconfiguration.

### *Migration*

The migration operator enables to combine the algorithms running on different processors into a single distributed genetic algorithm. The migration rate $R_{mig}$ specifies the epoch length that starts the transmission of best individuals from each subpopulation into ring topology of processes. We used elitism concept so as the only such a individual is accepted from the other process that is better than the current one. The effectiveness of the migration depends namely on the migration rate. An additional optional approach can be activated which allows the acceptance of the better or worse individual in each generation.

### *Selection*

After generating offspring the procedure *Selection* is activated to choose the new subpopulation from joined set of parents and offspring. In the CPGEN algorithm a competetive strategy is implemented - all the parents and offspring compete each other and the $N_p$ fittest individuals survive and are moved into the next generation.

## EXPERIMENTAL RESULTS

Numerous experiments were done to demonstrate the behaviour and the efficiency of the CPGEN algorithm. The presented experiments runnig on the T9000 station are intented on the placement task IC67 which is specified by 67 nodes, 136 nets and PCB with 75 locations (15 columns and 5 rows). In the first experiment (see Fig.3) the optimalization curves of the resulted minimal value of the cost with different number of processes/subpopulation is shown. In a) case the total population size is fixed so that the size of each subpopulation depends indirectly on the number $p$ of processes being used. In case b) the subpopulation size is fixed.

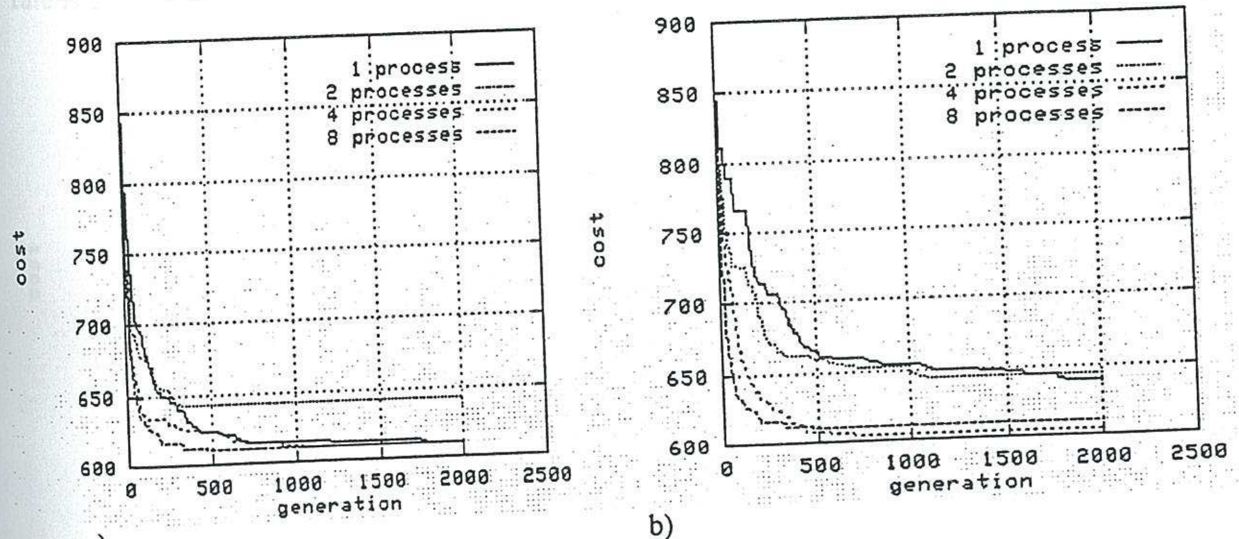


Fig.3. The optimization curves for different number of parallel processes a) fixed total population size $N_{tp}$= 240, b) fixed size of subpopulations, $N_p$ = 30.

Both types of experiments demonstrate benefits of parallel approach to the genetic algorithms. If eight logical processes are used the optimalization procedure converges faster than in case when one logical process is used. Obviously, the solution for more processes is equal or even better than for one logical process (which in fact represents the simple GA).
The most important advantage is the considerable speedup of computations when more processors are available. In the above mentioned case the speedup coefficient for four processes can be theoretically almost equal to four but in our case is lowered due the 17% communication losses in the used ring topology (see Fig. 4).

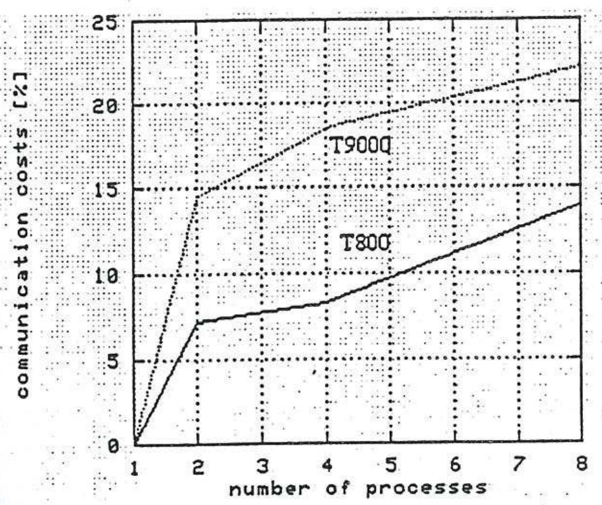


Fig.4 Communication costs in the ring topology.

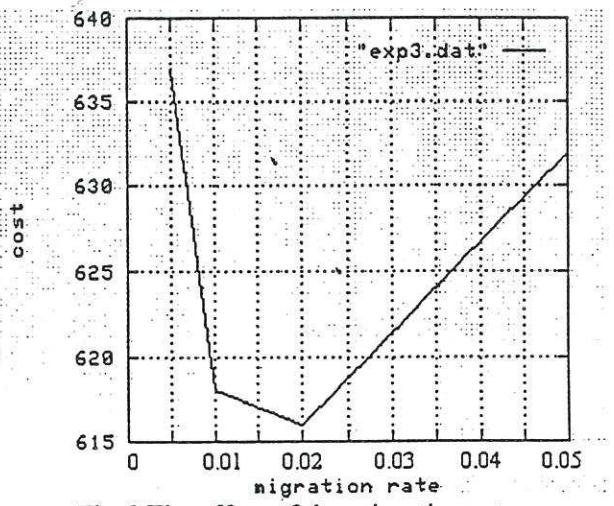


Fig.5 The effect of the migration rate.

The influence of processes number on the communication costs to total computational time ratio for T9000 and T800 station as well is shown. The relatively large communication costs is probably due to the features of C language used. The OCCAM platform provides lower costs [7].

Fig. 5 shows the effect of the migration rate on the cost. The proper range of the migration rate is from 0.01 to 0.03. The mutation rate does influence the convergence of the optimization process. In Fig. 6 the test of a placement problem with 25 nodes/elements is presented. The proper values of the mutation rate is problem size dependent an lies mostly in the range from 0.02 to 0.05.

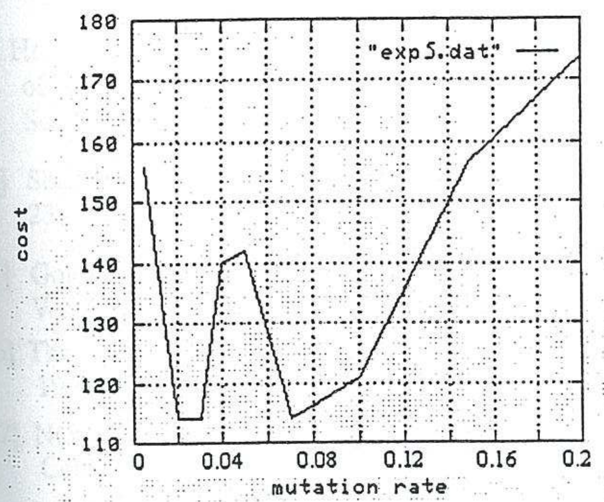


Fig.6 The influence of the migration rate on the cost.

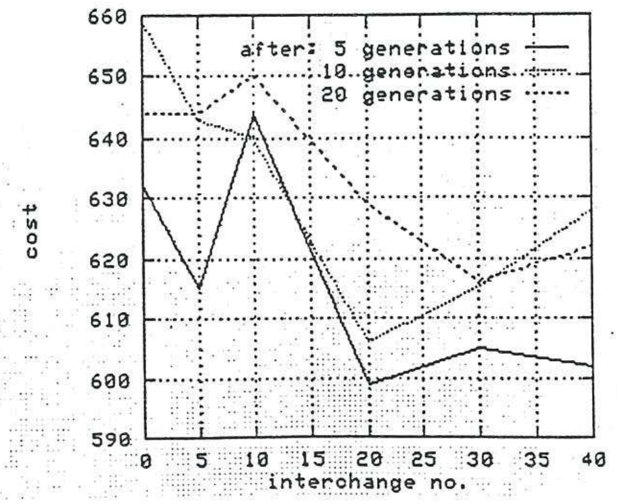


Fig.7 The effect of the heuristic procedure.

Fig. 7 shows the influence of the heuristic procedure used to the cost minimization. The technique of the pairwise interchange of symbols/nodes is used. It can be seen that with the increasing number of pairs the cost is decreasing. The rate of the procedure activation is specified by an epoch length lasting 5, 10 or 20 generations. The higher rate leads to lower cost. On the other hand it is evident that this process wastes the computational time.

## CONCLUSION

This paper describes a new parallel placement algorithm for the PCB layout with set of the discrete slots. The CPGEN program was developed in C language and implemented on the transputer station T800 and T9000. The set of experimental works were done to demonstrate the ability and efficiency of the parallel genetic algorithm. The migration operator and heuristic approach for a local optimization seems to be the main tool for the speedup and better convergence. The mutation strategy plays a great role. It was interesting to examine a dynamic setting the migration rate using fuzzy inference module. However, the results obtained were unambiguous and were too much problem sensitive. Therefore the AMR procedure for automatic setting the migration rate was used. Nevertheless the research in fuzzy logic domain is getting on and the fuzzy rule model is further developed.
A slightly modified version of the single genetic algorithm for the partitioning problem was developed. The results are very hopeful and the algorithm could be used for the k-partitioning of hypergraphs which can represent e.g. the decomposition of a large logic design (logic scheme) into subsystems (smaller replaceable units/modules).

## ACKNOWLEDGEMENT

This work was done under the support of the grant agency GAČR contract number 102/95/1334.

## REFERENCES

[1] Schwarz J.: Design Automatization of Composition and Placement of Integrated Circuit. PhD. Theses, Technical University of Brno, Faculty of Engineering and Computer Science, Department of Computer Science and Engineering, Brno, 1993, 136 pp., in Czech.

[2] INMOS: Transputer databook, Prentice-Hall, 1989.

[3] Hroník R.: Genetic Algorithm with Fuzzy Program Module, Diploma Project, Technical University of Brno, Faculty of Electrotechnical Engineering and Computer Science, Department of Computer Science and Engineering, Brno, 1996, 55 pp., in Czech.

[4] Shahookar K., Mazumder P.: VLSI Cell Placement Techniques, ACM Computing Surveys, Vol. 23, No. 2, June 1991, pp. 195-220.

[5] Goldberg E.,D.: Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, 1989.

[6] The FIDE (Fuzzy Inference Development Environment) User and Reference Manual, Aptronix 1992.

[7] Němec V., Schwarz J.: Parallel Genetic Algorithm on Transputers, 2.nd International Mendel Conference on Genetic Algorithms, June 26-28, 1996, Brno, Czech republic, pp. 85-90.

Schwarz Josef, Očenášek Jiří:

**Experimental study: Hypergraph partitioning based on the simple and advanced genetic algorithm BMDA and BOA.**

A-3

# MENDEL '99

5th International Conference on Soft Computing

Evolutionary Computation, Genetic Programming,
Fuzzy Logic, Rough Sets, Neural Networks, Fractals

June 9-12, 1999, Brno, Czech Republic

# EXPERIMENTAL STUDY: HYPERGRAPH PARTITIONING BASED ON THE SIMPLE AND ADVANCED GENETIC ALGORITHM BMDA AND BOA

Josef Schwarz, Jiří Očenášek

Technical University of Brno
Faculty of Engineering and Computer Science
Department of Computer Science and Engineering
CZ - 61266 Brno, Božetěchova 2
e-mail: schwarz@dcse.fee.vutbr.cz
e-mail: xocena00@stud.dcse.fee.vutbr.cz

*Abstract: This paper is an experimental study on hypegraph partitioning using the simple genetic algorithm (GA) based on the schema theorem and the advanced algorithms based on the estimation of distribution of promising solution. Primarily we have implemented a simple GA based on the GaLib library[Gal94] and some hybrid variant included a fast heuristics to speed up the convergence of the optimization process. Secondly we have implemented the Univariate Marginal Distribution algorithm (UMDA) and the Bivariate Marginal Distribution algorithm (BMDA), both have been published even recently[Pel98] and used a share version of a superior new program BOA based on the Bayesian Optimization Algorithm [Pel99]. We have also extended the BMDA algorithm to a new version with finite alphabet encoding of chromozomes and new metric that enables the m-way partitioning graphs. The aim of our paper is to test the efficiency of new approaches for discrete combinatorial problems represented by hypergraph partitioning.*

*Key words: decomposition, hypergraph partitioning, simple and hybrid GA, estimation of distribution algorithm, Bayesian network.*

## 1 Introduction

Hypergraph partitioning is a well known problem of graph theory. In case of so called 2-way partitioning the bisection term is used. The graph representation can be used for many application problem e.g. for the system segmentation, network partitioning and VLSI layout.
More formal, the particular partitioning problem can be defined as follows: Let us assume a hypergraph $G=(V,E)$, with $n = |V|$ nodes and $t = |E|$ edges. Let $m$ be a natural number. A $m$-way partition is specified by disjoint partitions of nodes $A_0, A_1, \dots A_{m-1}$ with equal or predefined cardinality. The cost of the partition is defined as a function of the hyperedges having nonempty intersection with at least two partitions from the partitions $A_0, A_1, \dots A_{m-1}$. We call these hyperedges external ones. From the previous premises it follows that an unconstraint balanced partitioning problem is solved.
Many heuristics are used to solve this NP-complete problem. We can refer to the recent paper [Oom96] where a good overview of the known local search techniques is done and automaton-based algorithm is described in more detail. The hybrid genetic algorithm is described in [Pat95],[Bui96]. The mentioned simple genetic algorithms are based on the schema theory [Gol86]. It is known that during evolving a new population standard genetic operators often cause disruption of schemata mainly of large defining length. To prevent the problem a techniques of reordering of graph nodes in chromosomes was proposed by [Bui96]. We focus in this paper on another promising approach based on an estimation of the joint distribution of promising solutions proposed in [Pel98], [PGC98], [MUE98].

## 2 Problem formulation

The hypergraph is often modelled by bigraph (see Fig 1), where a bigraf $G(V,S)$ with the set of nodes $V$ and set of nets $S$ is presented. This is an example of 3-way partitioning problem with a 9 nodes and a 10 nets. Each partition/assembly $A_0, A_1, A_2$ contains 3 nodes. The cost/objective function $L$ is based on external nets that connect nodes from different assemblies:

$$L = \sum_{s_i \in S_e} [D(s_i) - 1] \qquad (1)$$

where $D(s_i)$ is the degree of the external nets $S_e = \{s_1, s_2, s_3\}$.

The incidence of external nets to each assembly is represented by dashed lines. The minimal set *{L1, L2, L3, L4}* of external connections is represented by full lines produced by the spanning tree technique. In the following 3-way partitioning the cost $L$ equals to 4.
Another way how to calculate the external connection is based on the number of nets that incident with *i-th* assembly. The $S_i$ items can be simply stated : $S_0=5$, $S_1=5$, $S_2=4$; for the known total number of nets $t=10$ we get $L=4$.

$$L=\sum_{i=0}^{m-1} S_i - t \tag{2}$$

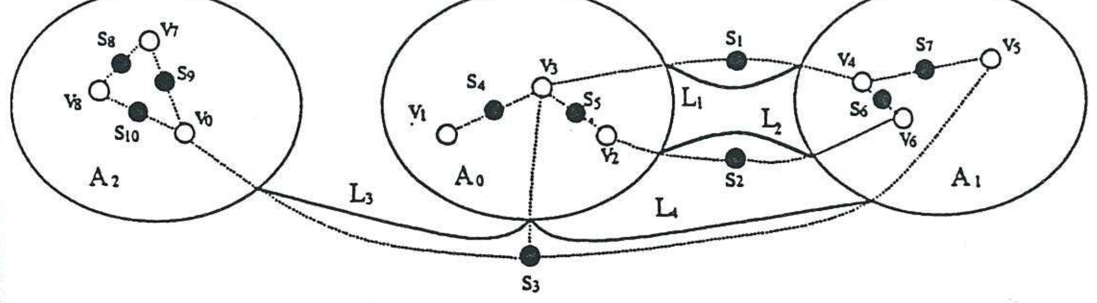


Fig.1 The 3- way partitioning case of a hypergraph.

In case that each net connects only 2 nodes the hypergraph can be reduced to a simple graph $G(V,E,W)$, where the connection matrix $W=[wij]$ represents the weight of edge/connection between node $i$ and $j$.

## 3 Genetic algorithms

To solve the partitioning problem we have implemented 2 types of genetic algorithms- simple genetic algorithm SGA and its heuristic versions, advanced BMDA algorithm and used the BOA program. For all of them the following ordinary string/chromosome encoding is used:

| Genotype | | Meaning |
|---|---|---|
| Trisection | 0 0 0 1 1 1 2 2 2 | Gene value /Assembly number |
| Bisection | 1 1 0 0 1 0 1 0 0 | |
| | 0 1 2 3 4 5 6 7 8 | Locus/Node number |

Table 1. The ordinary string encoding for bisection and trisection of hypergraph.

The gene value represents the assembly number, the index of locus specifies the node number. The efficiency of the BMDA and BOA algorithm is determined by the level of gene dependency. That is why it is useful to express the cost function using the string encoding. For the simplest case of 2 - way partitioning/bisection of a simple graph $G(V,E,W)$ we derived on the binary string $X=(x_0, x_1,..., x_{n-1})$ the following quadratic cost function:

$$L=\sum_{\substack{i=0\\ j>i}}^{n-1} w_{ij}(x_i+x_j-2x_ix_j) \tag{3}$$

with the balance condition $\sum_{i=0}^{n-1} x_i = \sum_{i=0}^{n-1}(1-x_i)$,

where the coefficient $w_{ij}=1$ in case the net/edge exists between node $i$ and $j$, else $w_{ij}=0$. In case of m-way partitioning of a single graph the function is not binary because the string is alphabetic:

$$L=\sum_{\substack{i=0\\ j>i}}^{n-1} w_{ij}\, f_k(x_i,x_j)\ ;\ \ f_k(x_i,x_j)=1 \ \ for\ \ x_i \neq x_j\, ;\ f_k(x_i,x_j)=0 \ \ for\ \ x_i=x_j \tag{4}$$

In case of m-way partitioning of hypergraphs the useful term for cost is quite complex and is beyond the scope of this paper.

### 3.1 Simple genetic algorithm

We have implemented a simple genetic algorithm SGA described in [Gol86] using the well known GaLib library [Gal94]. The fitness function is based on the cost $L$: *Fitness* = *1/(L+1)*. The selection scheme is roulette wheel with the linear-scaled fitness. Offspring are generated by the one-point crossover operator. To implement mutation the values of several genes are changed to any of the possible allele values (*flip mutator)*. The genetic operators described do not guarantee the equal size of all assemblies. After crossover or mutation completion, strings should be balanced/normalized randomly to keep equal or predefined number of different alleles. In the replacement stage the principle of elitism is used. We designed also SGAN version with advanced/positive normalization of string contributed to cost decrease. To improve the convergence speed of the optimization process for more complex problems a heuristic procedure was added to the SGA algorithm to get a hybrid genetic algorithm SGAH. It is based on the reconfiguration of some offspring in the current population using the pairwise interchange of genes with cost decrement. The procedure is activated seldom only during an epoch $H_e$ formed by a number of generations. It is possible to set the intensity $H_i$ and the range $H_r$ of the reconfiguration.

## 3.2 UMDA and BMDA algorithm

The following methods are based on probability theory and statistics. They use statistical information contained in the set of selected parents to detect gene dependencies. The estimated probability model is used to generate new promising solutions according to this distribution. Generally, UMDA, BMDA and BOA belong to an EDA class of algorithm (Estimation of Distribution Algorithm) [Mue98], which can be described as follows:

```
Generate initial population of size N (randomly);
While termination criteria is false do
begin
  Select parent population P of M individuals according to a selection method ($M \leq N$);
  Estimate the distribution of the selected parents;
  Generate new offspring (according to the estimated model);
  Replace some individuals in current population by generated offspring;
end
```

UMDA [ Pel98 ] (Univariate Marginal Distribution Algorithm) assumes that genes are mutually independent. Let us denote a chromosome length by *n*. For each gene position $i \in \{0..n-1\}$ and each possible value of this gene $x_i \in \{0,1\}$, the univariate marginal frequency $p_i(x_i)$ is defined as the frequency of strings that have $x_i$ on *i-th* position in the parent population *P*: $p_i(x_i) = n_i(x_i)/N$, where $n_i(x_i)$ is a number of appearances of the allele $x_i$ on *i-th* position. Each new individual $X = (x_0, x_1, \dots, x_{n-1})$ is generated by UMDA according to the distribution

$$p(X) = \prod_{i=0}^{n-1} p_i(x_i), \tag{5}$$

so the value of *i*-th gene is set to value *a* with the probability equal to $p_i(a)$. UMDA is able to cover linear problems only.

BMDA [ Pel98], [Oce99 ] (Bivariate Marginal Distribution Algorithm) is an extension of UMDA. In addition, the pair dependencies are allowed. The bivariate marginal frequency $p_{i,j}(x_i, x_j)$ is defined as the frequency of individuals in parent population *P*, that have values $x_i$ and $x_j$ on positions *i* and *j* at the same time: $p_{i,j}(x_i, x_j) = n_{i,j}(x_i, x_j)/N$. Conditional probability of occurrence of the value $x_i$ on *i*-th position in the case of occurrence of $x_j$ on *j*-th position is determined

$$p_{i,j}(x_i \mid x_j) = \frac{p_{i,j}(x_i, x_j)}{p_j(x_j)} \tag{6}$$

We extended the concept of UMDA and BMDA for the alphabet encoding, so that $x_i \in \{0, \dots, r_i - 1\}$ and $x_j \in \{0, \dots, r_j - 1\}$. For each pair of positions *i, j* the count of each combination of values can be summarized into following contingency table:

| $x_i \backslash x_j$ | 0 | 1 | ... | $r_j - 1$ | $\Sigma$ |
|---|---|---|---|---|---|
| 0 | $n_{i,j}(0,0)$ | $n_{i,j}(0,1)$ | ... | $n_{i,j}(0, r_j - 1)$ | $n_i(0)$ |
| 1 | $n_{i,j}(1,0)$ | $n_{i,j}(1,1)$ | ... | $n_{i,j}(1, r_j - 1)$ | $n_i(1)$ |
| ... | ... | ... | ... | ... | ... |
| $r_i - 1$ | $n_{i,j}(r_i - 1, 0)$ | $n_{i,j}(r_i - 1, 1)$ | ... | $n_{i,j}(r_i - 1, r_j - 1)$ | $n_i(r_i - 1)$ |
| $\Sigma$ | $n_j(0)$ | $n_j(1)$ | ... | $n_j(r_j - 1)$ | N |

Gene dependencies are discovered by Pearson's chi-square statistics [Pel98], we use the following form of equation [Oce99]:

$$X_{i,j}^2 = N\left(\sum_{k=0}^{r_i-1}\sum_{l=0}^{r_j-1}\frac{n_{i,j}^2(k,l)}{n_i(k)\,n_j(l)}-1\right) \tag{7}$$

This metric is symmetrical, $X_{i,j}^2 = X_{j,i}^2$, so for each couple of positions it has to be computed only once. Genes are considered to be independent if the result does not meet certain threshold. For example binary genes are independent for 95% if $X_{i,j}^2 < 3.84$.

The dependency information is used to build up the acyclic dependency graph, which can be seen as the set of trees. The root nodes correspond to the positions where the values are generated using the univariate distribution, the values of positions connected to already generated positions in the graph are subsequently generated using the conditional probability. After chromosome generation the balancing procedure is used.

We have also implemented the following (non-symmetrical) metric giving as good results as Pearson's statistics:

$$K2_{i,j} = \left(\prod_{l=0}^{r_i-1}\frac{(r_j)!}{(r_j+n_i(l))!}\prod_{s=0}^{r_j-1}(1+n_{i,j}(l,s))!\right)\Bigg/\left(\frac{(r_j)!}{(r_j+N)!}\prod_{z=0}^{r_j-1}(1+n_j(z))!\right) \tag{8}$$

This equation is derived from K2 metrics used in BOA algorithm discussed in the next chapter.

## 3.3 BOA program

BOA [PGC98] (Bayesian Optimization Algorithm) uses Bayesian network to encode the structure of a problem. It is an analogy of BMDA dependency graph, but the higher order gene dependencies can be covered too. For each variable $X_i$ a set of variables $\Pi_{X_i}$ is defined it depends on, so the distribution of individuals is encoded as

$$p(X) = \prod_{i=0}^{n-1} p\left(X_i \mid \Pi_{X_i}\right) \tag{9}$$

Generally, the existence of directed edge from $X_j$ to $X_i$ in the network implies the belonging of the variable $X_j$ to the set $\Pi_{X_i}$. To reduce the space of networks, number of incoming edges into each node is limited to *k*.

The Bayesian Dirichlet (BD) metric is used to measure the quality of the network [PGC98]. A special case of BD metric, so-called K2 metric, is used when no prior information about the problem is available. Actually, the equation (8) is derived from K2 metrics for *k=1* and alphabet encoding. It determines the relative metric improvement for one edge addition.

In the shared implementation of BOA [Pel99] a simple greedy algorithm is used to search for a good network. In each step the best edge is added. By the term 'best edge' we mean the edge giving the highest K2 metric for the network B' that can be constructed from the actual network B by adding this edge. It must also keep the network acyclic and meet the limit of incoming edges.

After network construction new instances are generated using the univariate and conditional probability in a similar way as for BMDA algorithm.

## 4 Experimental results

Numerous experiments were done to demonstrate the behaviour and the efficiency of the individual algorithms. The two types of graph structures are used:

1. Regular graphs RLXB with grid structure [Schw98], where the notion X specifies the number of nodes, B specifies the existence of bottle-neck in the graph structure; the global optimum is known. As an example a bisection of graph RL64B is represented in Fig.2a having a 2-edge bottle-neck in the dashed cut line. The regular graphs with bottleneck appear to be a proper test benchmark with many local optima.
2. Hypergraphs representing real circuits labelled by ICX [Schw86]. The global optima is not known. The structure of circuits can be characterized as a random logic. The hypergraph IC67 consists of 67 nodes and 134 edges/nets, the IC151 consists of 151 nodes and 286 edges/nets.

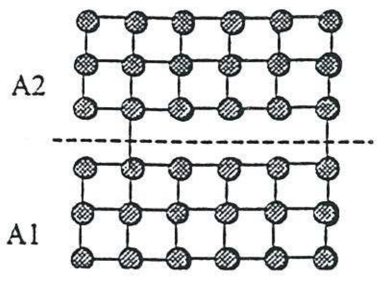


Fig.2 Regular graph RL36B.

The main experimental results are listed in Table 2. In case of regular graphs (see part A) the items in the table represent the average number of fitness for 5 successful runs with different initial random population. The parameters of the algorithms are set to the minimal possible values to get global optimum in all 5 runs. In case of graphs IC67 and IC151 (see part B) trials are focused on the goal to get a local minimum; the number of fitness evaluations is limited. The BOA and BMDA algorithms perform well, the SGA and SGAN converge slowly, the SGAH is the best of all algorithms.

| Graphs | m [assemblies] | A. Average number of evaluations to get global optimum L | | | | | Global optimum L |
|---|---|---|---|---|---|---|---|
| | | SGAH | SGAN | SGA | BMDA | BOA, k=3 | |
| RL36B | 2 | 7115 | 1970 | 60942 | 5650 | 4242 | 2 |
| RL36B | 4 | 42812 | 3161 | 421827 | 28055 | - | 8 |
| RL36B | 6 | 65255 | 12133 | 1214150 | 293000 | - | 14 |
| RL48B | 2 | 4580 | 12850 | *177933 | 20700 | 6036 | 2 |
| RL48B | 4 | 21375 | 138500 | - | 250320 | - | 8 |
| RL64B | 2 | 14900 | 8200 | 808571 | 64500 | 7905 | 2 |
| RL64B | 4 | 69830 | 306800 | - | 283200 | - | 10 |
| RL100B | 2 | 127825 | 105275 | - | 680000 | 19550 | 2 |
| | | B. Cost/Average number of evaluations | | | | | |
| IC67 | 2 | 36/4050 | 41/22950 | 45/37900 | 41/15290 | 40/12375 | unknown |
| IC67 | 4 | 69/16530 | 86/42460 | 103/39780 | 72/25850 | - | unknown |
| IC151 | 2 | 58/19990 | 73/44220 | 110/46500 | 66/25465 | 67/22000 | unknown |
| IC151 | 4 | 126/40760 | 218/43550 | 256/41510 | 208/26895 | - | unknown |

Tab.2. Experimental results for SGA variants, BMDA and BOA algorithms.

In Fig.2 a comparison of BOA and BMDA algorithms is done for the case of bisection of graphs RL36B, RL48B and RL64B. The BOA algorithm for *k=3* performs very well. In a) part the dependency of the minimal population size on the problem size and in b) part the dependency of number of fitness evaluations on the problem size are shown.

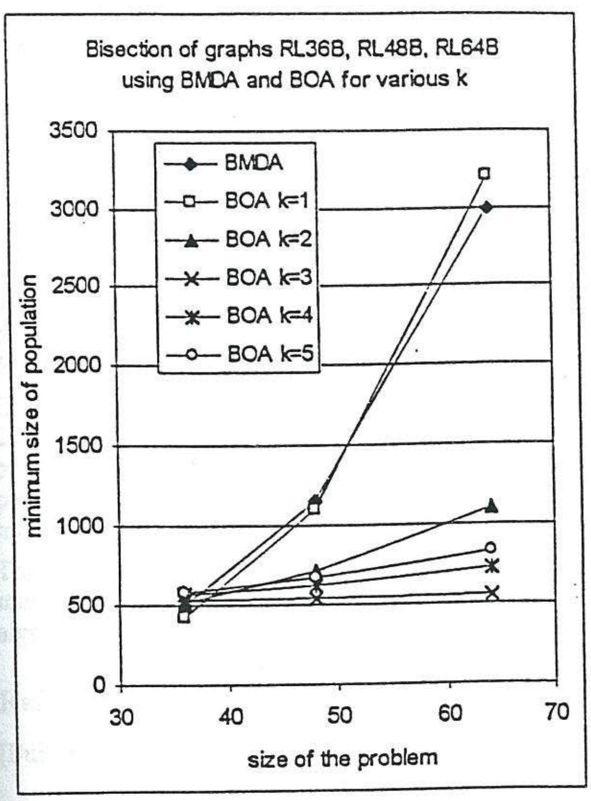
In Fig.3a the performance of BMDA, BOA for k=3, SGA and SGAH is presented for the bisection of graphs mentioned above and for RL100B graph. It is evident that the BOA algorithm gains on all the algorithms used for bisection. BMDA converges much slower namely for RL100B graph. The SGAH algorithm is good enough, better than BMDA and worse than BOA. The SGA algorithm performs very purely with great number of fitness evaluations; in case of graph RL100B the global optimum was not reached during five independent trials. In case of graph RL48B the global optimum was reached only in three of five trials (the item in Table 2 was signed by asterisk).

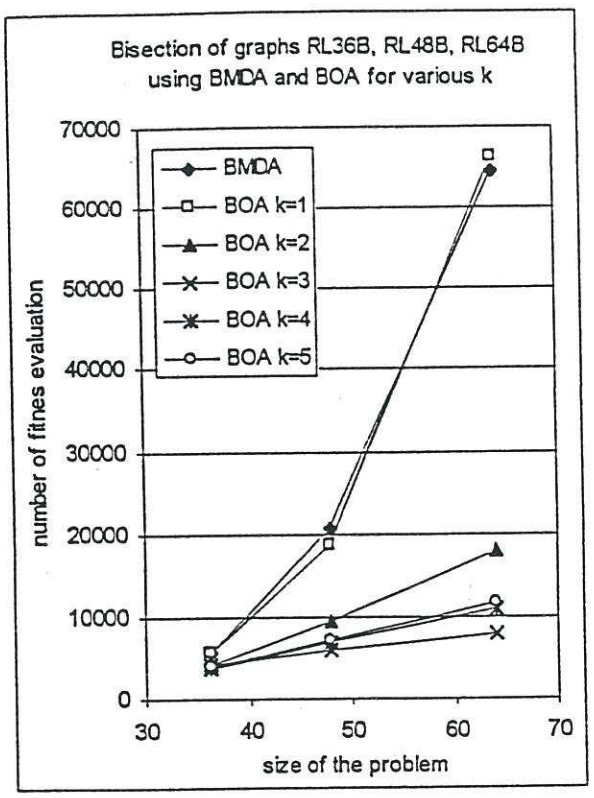
In Fig.3b the results of multi-way partitioning of graph RL36B are presented for m=2,4 and 6. The SGAN algorithm with positive normalization performs quite well, the SGA performs purely. BMDA and SGAH algorithms perform good enough for quadrisection but for 6-way partition the number of fitness evaluations rapidly increases. The relatively good results are due to the small size graph used for partitioning. The multi-way partitioning for *m>4* seems to be a hard problem for great size of problems.

During all the experiments the following range of parameters was used: for BOA and BMDA the 50% truncation selection (a subset of population for estimation of distribution), for BOA the parameter *k=1-5* of Bayesian network. In case of SGA and SGAN algorithms the crossover rate $R_c$=0.5, mutation rate $R_m$=0.05 is used. In case of SGAH algorithm crossover rate Rc=0.5, mutation rate $R_m$=0.01- 0.05, $H_a$=5 -10, $H_i$ = 10-30, $H_r$ = 10-30 is used.

Generally, for SGA variants, the problems with proper setting of various heuristic parameters to prevent the premature convergence occur.

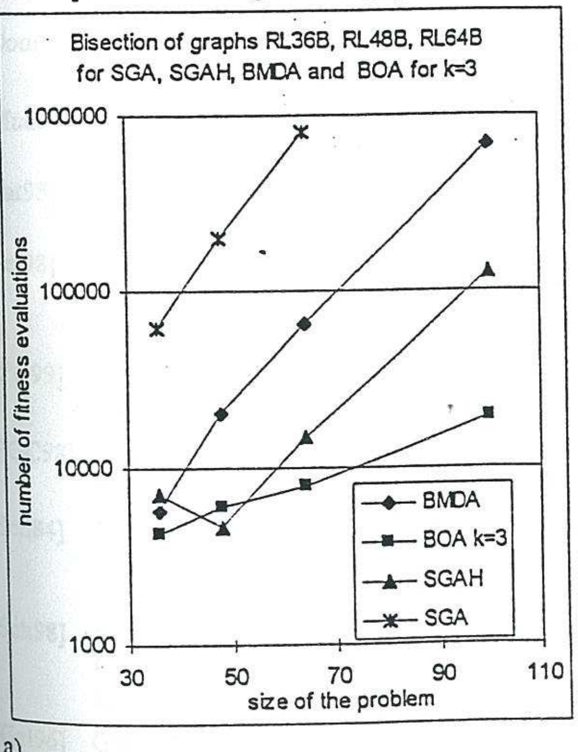
In case of partitioning of the hypergraphs IC67, IC151 (see Table 2, part B), the population size is set to a minimum value N=550 for BOA and BMDA as well; the number of evaluations is limited to 27500. For SGAH algorithm $R_m = 0.01$, $H_e=5$, $H_i = 20$, $H_r = 20$, for SGA and SGAN $R_m = 0.05$, $R_c = 0.5$; the population size N=100, the number of evaluations is limited to 50000 for all the SGA variants.

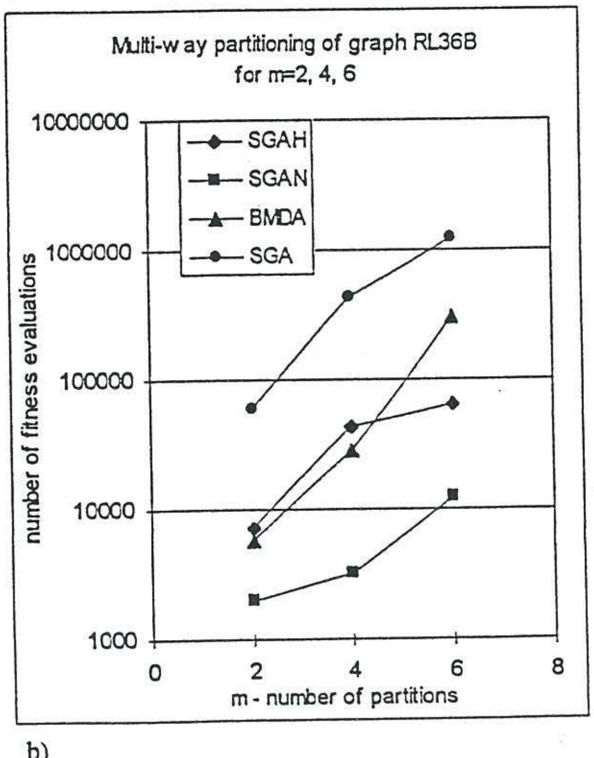




a) b)

Fig2 a) The minimum size of population found to reach the global optimum for regular graphs RL36B, RL48B, RL64B, b) Number of fitness evaluations for bisection of regular graphs RL36B, RL48B, RL64B to reach the global optimum. The range of used population size was 500-3000 for BMDA and 430-3200 for BOA.





a) b)

Fig.3 a) Number of fitness evaluations for bisection of regular graphs RL36B, RL48B, RL64B, RL100B to reach the global optimum, using algorithms SGA, SGAH, BMDA and BOA for k=3,. The range of used population size was 200-700 for SGA, 100-200 for SGAH, 500-20000 for BMDA(3000 for RL36B) and 535-850 for BOA, b) Number of fitness evaluation as a function of number of assemblies m for graph RL36B, (for Y axis log-scaling is used).

## 5 Conclusions

The paper describes shortly three types of genetic algorithms.. The first one (SGA) is based on the schemata theory, the second one (BMDA) on an estimation of the distribution of promising solution, the last one BOA uses the extra techniques to model data by Bayesian network.

We have focused on the implementation of SGA and BMDA version of genetic algorithm and adaptation of BOA program with the aim to compare their performance and efficiency. From the experiments it is evident that for bisection of regular test graphs the best performance offers the BOA algorithm with small amount of evaluations and size dependency. BMDA algorithm works well on the bisection of graphs up to 64 nodes; for RL100B graphs the number of evaluations increases dramatically. The SGA algorithm performs very purely, the number of evaluations is very high and it failed for the RL100B graph. The SGAH version with heuristics seems to be a good tool but it is very sensitive on setting various parameters and it often gets stuck in local optima. The bad performance of SGA seems to be caused by phenomenon of building block disruption.

The multi-way partitioning is a hard problem for all the algorithms. The BMDA algorithm works well enough but only for relatively small problems. The SGAN with positive normalization of partitions works best of all but for small problem and with the tendency to get stuck in local optima.

Our contribution can be seen in the extension of BMDA algorithm to the finite alphabet encoding of chromosomes to be able to solve the task of multi-way partitioning graphs. We have also proved the efficiency of the modified BMDA algorithm using the same heuristic procedure as in the SGAH algorithm. The performance of this version of BMDA is very similar to the original one. We have also implemented the modified K2 metric for BMDA instead of Pearson's statistics without remarkable influence.

The future work will be focused namely on an exploration of the BOA version with finite alphabet encoding for multiple partitioning graphs and placement problem. Other activities will be directed towards the usage of EDA algorithms with problem knowledge.

## References

[Bui96] Than Nguen Bui, Byung Ro Moon: Genetic Algorithm and Graph Partitioning. IEEE Transactions on Computers, Vol.45, No.7, July 1996, pp. 841-855.

[Gol89] Goldberg E.,D.: Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, 1989.

[Oce99] Očenášek J.: Advanced genetic techniques for problem optimization, Diploma Project, Department of Computer Science and Engineering, Technical University of Brno, 1999, pp.1-23, in Czech.

[Oom96] Oomen J. B., Croix V.: Graph partitioning using learning automata. IEEE Trans. on Computers, vol. 45, No2, February 1996, pp.195- 206.

[Mue98] Muehlenbein H., Rodriguez A. O.: Schemata Distributions and Graphical Models in Evolutionary Optimization. GMD Forschungs Zentrum Informationstechnik, 53754-St. Augustin, 1998, pp.1-21.

[Pat95] Ananta K.Majhi, L.M. Patnaik, Srilata Raman: A Genetic algorithm-based circuit partitioner for MCMs. Microprocessing and Microprogramming 41 (1995), pp. 83-96.

[Pel98] Pelikan M., Muehlenbein H.: Marginal Distribution in Evolutionary Algorithms. 4th International Mendel Conference on Genetic Algorithms, Optimization Problems, Fuzzy Logic, Neural Networks, Rough Sets, June 24-26, 1998, Brno, Czech Republic, pp. 91-95.

[Pel99] Pelikan M.: A Simple Implementation of Bayesian Optimization Algorithm in C++(Version1.0). Illigal Report 99011, February 1999, pp. 1-16.

[PGC98] Martin Pelikan, David Goldberg, and Erick Cantú Paz: Linkage Problem, Distribution Estimation, and Bayesian Networks. IlliGal Report No. 98013, November 1998,pp. 1-25.

[Sch84] Schwarz J.: Design Automatization of Composition and Placement of Integrated Circuit. PhD. Theses, Technical University of Brno, Faculty of Engineering and Computer Science, Department of Computer Science and Engineering, Brno, 1984, 136 pp., in Czech.

[Sch98] Schwarz J.: Genetic algorithm for partitioning circuits. 4th International Mendel Conference on Genetic Algorithms, Optimization Problems, Fuzzy Logic, Neural Networks, Rough Sets, June 24-26, 1998, Brno, Czech Republic, pp.126-131.

[Gal96] Galib: A C++ Library of Genetic Algorithm Components, http://lancet.mit.edu/edu/ga/.

| | |
|---|---|
| **Schwarz Josef, Očenášek Jiří:**<br>**A problem knowledge-based evolutionary algorithm KBOA for hypergraph bisectioning.** | A-4 |

# Knowledge-Based Software Engineering

Proceedings of the Fourth Joint Conference on Knowledge-Based Software Engineering Brno, Czech Republic, 2000

Edited by

Tomáš Hruška
*Brno University of Technology, Czech Republic*

and

Masa-aki Hashimoto
*Kyushu Institute of Technology, Japan*

IOS Press
Ohmsha

Amsterdam • Berlin • Oxford • Tokyo • Washington, DC

*Knowledge-Based Software Engineering*
*T. Hruška and M. Hashimoto (Eds.)*
*IOS Press, 2000*

# A Problem Knowledge - Based Evolutionary Algorithm KBOA for Hypergraph Bisectioning

Josef SCHWARZ, Jiří OČENÁŠEK
*Brno University of Technology, Faculty of Engineering and Computer Science*
*Department of Computer Science and Engineering*
*CZ - 61266 Brno, Božetěchova 2*
*e-mail: schwarz@dcse.fee.vutbr.cz, ocenasek@dcse.fee.vutbr.cz*

**Abstract.** This paper is an experimental study on an utilization of additional knowledge about the decomposition problem to be solved. We have demonstrated this approach on the hypergraph bisectioning that can serve as a model of system decomposition in common, data base decomposition etc. We have focused on the extension of the Bayesian Optimization Algorithm BOA. The extension of the original BOA algorithm is based on the usage of a prior information about the hypergraph structure. This knowledge is used for both setting initial Bayesian network and the initial population using injection of clusters to improve the convergence of the decomposition process. The behaviour of our version KBOA is tested on the set of benchmarks, such as grid and random geometric graphs as well as real hypergraphs.

## 1 Introduction

The decomposition of the system such as communication networks, complex data base system and VLSI layout are usual tasks to be solved. These tasks can be often formalized as a hypergraph partitioning into *k* partitions – it is a well known problem of graph theory. This paper deals with the case of 2-way partitioning called bisectioning. If necessary the k-way partitioning can be realized by a series of the hypegraph bisectioning.

Many heuristics are used to solve this NP-complete problem. We can refer to the well known paper [1], where a good overview of the efficiency of known local search techniques as well as simulated annealing is presented. A very good survey about netlist partitioning techniques was done in [2], where the geometric representation, combinatorial formulations, move-based and clustering approaches are discussed. The interesting hybrid genetic algorithm is described in [3]. The authors have used a simple genetic algorithm (GA) [4], with reordering of graph nodes to receive shorter schemata/building blocks so as to prevent the disruption of schemata mainly of large defining length.

The problem of schemata disruption has been intensively studied in GA community during few last years. As a result a new class of promising approaches based on an estimation of the joint distribution of promising solutions (EDA) was proposed in [5], [6] and recently new ideas in [7], [8].

We focus in this paper on an extension of Bayesian Optimization Algorithm BOA [5], [9], [10] based on the problem knowledge to speed up the convergence of optimization process during hypergraph bisectioning.

The paper is organized as follows. In chapter 2 and 3 the problem formulation is declared and cost function is defined. Chapter 4 presents the original BOA algorithm which allows to understand the ideas of knowledge based approach presented in chapter 5. In chapter 6 benchmark tests are specified and experimental results are discussed. In chapter 7 the conclusions and research activities for the future are summarized.

## 2 Problem formulation

More formal, the particular bisectioning problem can be defined as follows: Let us assume a hypergraph $H=(V,E)$, with $n = |V|$ nodes and $m = |E|$ edges. We are supposed to find such a bisection $(V1, V2)$ of $V$ into equal sized parts ( $/V1/=/V2/$, $V= V1 \cup V2$ ) that minimizes the number of hyperedges which have nodes in different set $V1$, $V2$. The set of external hyperedges can be labelled as $E_{cut}(V1,V2)$. The primary objective function/cost of the hypergraph bisectioning is the number of external hyperedges, shortly called cut size:

$$c\,(V_1,V_2)= /E_{cut}\,(V_1,V_2)\,/ = /\,\{e \in E\,/\,e \cap V_1 \neq 0,\ e \cap V_2 \neq 0\}\,/ \qquad (1)$$

This is the case of strongly balanced bisectioning. Next, we define the set $M(v)$ of hyperedges incident to node $v$ and the set $N(v)$ of nodes that are neighbours of node $v$:

$$M(v) = \{e \in E\,/v \in e\} \qquad (2)$$

Using the previous equation the cost can be expressed as

$$c(V1,V2)= /\bigcup_{v \in V_1} M(v)\,/ + /\bigcup_{v \in V_2} M(v)\,/ - m \qquad (3)$$

The immediate/first neighbours of a node $v$:

$$N(v) = \{u\ /\exists e\ \ v,\, u \in e,\, v \neq u\} \qquad (4)$$

The degree of node $v$ is defined as $D(v) = /N(v)\,/$.

An example of the hypergraph with 8 nodes and 5 hyperedges is shown in Fig.1. The bisection of the hypergraph resulted in $V_1=\{v_1, v_2, v_7, v_4\}$, $V_2 = \{v_5, v_6, v_3, v_8\}$. The external hyperedges $E_{cut}(V_1,V_2) = \{e_3, e_4\}$, cut size $c(V_1,V_2) = 2$.

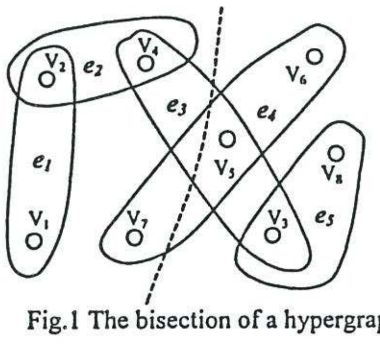


Fig.1 The bisection of a hypergraph

## 3 Solution encoding

For the BOA algorithm the following ordinary encoding of the solution is used:

Table 1. The ordinary solution encoding for bisection of hypergraph.

| Chromosome/String | | Meaning |
|---|---|---|
| gene value | 1 1 0 1 0 0 1 0 | Partition number |
| locus value | 1 2 3 4 5 6 7 8 | Hypergraph nodes |

Each solution of the bisection is represented by a chromosome, which is a binary string $X=(x_1, x_2, ..., x_n)$. The gene value $x_i$ represents the partition number, the index of locus specifies the node in the hypergraph. It is possible to express the cost function using the string encoding. In case of hypergraph bisectioning the cost function can be expressed using the equation (3):

$$c(V1,V2) = /\underset{\forall x_i=1}{\cup M(i)}/ + /\underset{\forall x_i=0}{\cup M(i)}/ - m, \quad for\ i=1,...,n \tag{5}$$

In case we define a dummy node $v_0$ with $M(v_0)=\varnothing$, we can transform the last equation into:

$$c(V1,V2) = /\cup M(i\ x_i)/ + /\cup M(i\ (1-x_i))/ - m, \quad for\ i=1,...,n\ , \tag{6}$$

where $M(i)$ is a set of hyperedges incident of node $i$.

## 4 BOA algorithm

In the simple genetic algorithms the standard crossover and mutation operators for offspring generation are used. In the last few years there has been a growing interest in the field of Estimation of Distribution Algorithms (EDAs) also called probabilistic model-building genetic algorithms, where crossover and mutation operators are replaced by probability estimation and sampling techniques. They use statistical information contained in the set of selected parents to detect gene dependencies. The estimated probability model is used to generate new promising solutions according to this distribution. The process can be described as follows:

```
Generate initial population of size N (randomly);
While termination criteria is false do
   begin
     Select parent population of S individuals according to a selection method ($S \leq N$);
     Estimate the distribution of the selected parents;
     Generate new offspring (according to the estimated model);
     Replace some individuals in current population by generated offspring;
   end
```

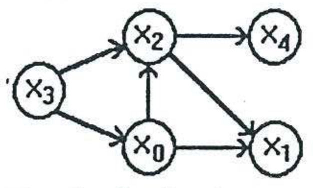
There are various probability distribution models with different complexity.

In BOA (Bayesian Optimization Algorithm) [5] Bayesian network (BN) is used to encode the structure of a problem. Each gene in the chromosome is treated as a variable and represented by a node in the dependency graph. For each variable $X_i$ it is defined a set of variables $\Pi_{X_i}$ called parents it depends on, so the distribution of individuals is encoded as

$$p(X) = \prod_{i=0}^{n-1} p(X_i \mid \Pi_{X_i}) \tag{7}$$

Generally, the existence of oriented edge from $X_j$ to $X_i$ in the network implies that the variable $X_j$ belongs to the set $\Pi_{X_i}$. To reduce the space of possible networks, the number of incoming edges into each node is limited to $k$. According to results in [10] we have limited the parameter $k$ in our experiments to be three.

Example of Bayesian network for 5 variables:



The corresponding joint probability distribution is

$$p(X) = p(X_3)\cdot p(X_0 \mid X_3)\cdot p(X_2 \mid X_0, X_3)\cdot p(X_1 \mid X_0, X_2)\cdot p(X_4 \mid X_2) \quad (8)$$

The Bayesian Dirichlet metric (BD) [11] is used to measure the quality of the network:

$$p(D,B \mid \xi) = p(B \mid \xi)\prod_{i=0}^{n-1}\prod_{\pi_{X_i}}\frac{m'(\pi_{X_i})!}{(m'(\pi_{X_i}) + m(\pi_{X_i}))!}\prod_{x_i}\frac{(m'(x_i,\pi_{X_i}) + m(x_i,\pi_{X_i}))!}{m'(x_i,\pi_{X_i})!} \quad (9)$$

where $\xi$ is the prior information about the problem, $D$ is the population of parents and $p(B \mid \xi)$ is the prior probability of the network $B$. The product over $x_i$ runs over all instances of the variable $X_i$ and the product over $\pi_{X_i}$ runs over all instances of the set of its parents $\Pi_{X_i}$. The term $m(\pi_{X_i})$ stands for the number of occurrences of the tuple $\pi_{X_i}$ in the population D and $m(x_i,\pi_{X_i})$ stands for the number of individuals in D having both $X_i$ set to $x_i$ as well as $\Pi_{X_i}$ set to $\pi_{X_i}$. The prior knowledge on the probability distribution is represented by numbers $m'(\pi_{X_i})$ and $m'(x_i,\pi_{X_i})$ representing the expected $m(\pi_{X_i})$ and $m(x_i,\pi_{X_i})$ values. The detailed description of the prior parameters is in the next section.

If no prior information is available, the simpler K2 metrics can be used. It is a special case of BD metrics having all prior counts $m'(x_i,\pi_{X_i})$ set to 1 and all prior counts of parent configurations $m'(\pi_{X_i})$ set to number of possible values on *i*-th position (equal to 2 for binary chromosome).

Many algorithms can be used to build up the network. The optimal search is NP-hard, thus in the implementation a simple greedy algorithm was used with only one edge addition in each step. The algorithm starts with an empty network *B* and the edge giving the highest increase of metric value is then added to the network *B*. This process is repeated until no more addition is possible.

After network is constructed, new instances/solutions are generated. First, the variables/nodes of BN are ordered in the topological order and each iteration, the nodes whose parents are already calculated are generated using the conditional probabilities. This is repeated until all the variables are generated.

## 5 Knowledge based KBOA algorithm

For many combinatorial problems the information about the structure of the problem is available. If the information is complete, the dependence graph can be constructed in-hand by the expert. This method is used in the FDA (Factorized Distribution Algorithm) [6]. In our case, the structure of the problem is given by the list of edges between hypergraph nodes but the decomposition of the problem in the sense of FDA is not simple because the variables of the cost functions are overlapping. We can only expect that adjacent nodes in the hypergraph are dependent but we have no exact knowledge about the degree of independence between non-adjacent nodes.

The goal of our approach is to use modified BOA algorithm to investigate the structure of the problem together with the partial (local) information about the linkage to improve the performance of that algorithm. Three possible approaches were tested:

- The first one was based on the prior information about the problem in the BD metric. The term $p(B\,|\,\xi)$ represents the prior probability of the network *B*. We use a simple assignment $p(B\,|\,\xi)=c\kappa^{\delta}$, where *c* is a normalization constant, $\kappa \in (0,1]$ is a penalty constant factor and $\delta$ is the number of edges in the final Bayesian network having no match in the hypergraph to be bisected. The greedy algorithm that constructs the dependency graph using only one edge addition in each step, prefers edges existing between adjacent nodes of hypergraph (local information about the problem) with the setting $\delta$ to be zero. When no such edge exists, a penalization is used by setting $\delta=1$.
- The prior knowledge about the optimized problem is also represented by numbers $m'(\pi_{X_i})$ and $m'(x_i,\pi_{X_i})$. It can be shown, that the values of $m'(\pi_{X_i})$ and $m'(x_i,\pi_{X_i})$ express our belief in the accuracy of the values $m(\pi_{X_i})$ and $m(x_i,\pi_{X_i})$. When these values increase, the BD metric will tend more towards the prior assignment of distribution, when those numbers decrease the influence of sampled values $m(\pi_{X_i})$ and $m(x\ ,\pi_{X_i})$ becomes more significant. Our experiments with this type of information were not too promising, so we used the original uniform assignment.
- The standard BOA uses the random set of solutions in the initial population. We have tested the usage of the initial population based upon the knowledge of the hypergraph structure. Initial solutions are affected by injection of clusters of predefined size. The detection of such a cluster in a hypergraph is shown in the Fig. 2:

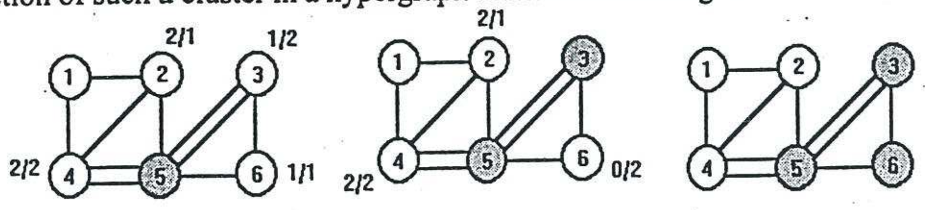


Fig.2. Clustering technique

First, a random node is selected to be the seed of the cluster. Then the effect of addition of each neighbouring node is expressed by the number of external and internal hyperedges incident of this node. Each step the node having minimal ratio external/internal edges is selected. When maximum size of cluster is reached or all neighbours are selected, next cluster is created. The cluster injection serves as a source of low-order building blocks.

## 6 Experimental results

### 6.1 Test graphs

The four types of graph structures are used:

1. Regular graphs *Gridn.c* with grid structure [3], [10] where the notion *n* specifies the number of nodes, *c* specifies the minimal cut size. As an example a bisection of graph *Grid100.2* is represented in Fig.3a having a 2-edge bottle-neck in the dashed cut line.
2. Random geometric graphs *Un.d* [1], [3]. Random geometric graph on *n* vertices is placed in the unit square and coordinates of its nodes are chosen uniformly.

There exists an edge between two vertices if their Euclidean distance is $l$ or less, where the expected vertex degree is specified by $d = n\pi l^2$.

3. Caterpillar graphs *CATk_n* [3], with $k$ articulations, six legs for each articulation, and $n$ nodes, see Fig. 3c with $k=3$ and $n=21$. This type of graph serves as a hard benchmark.

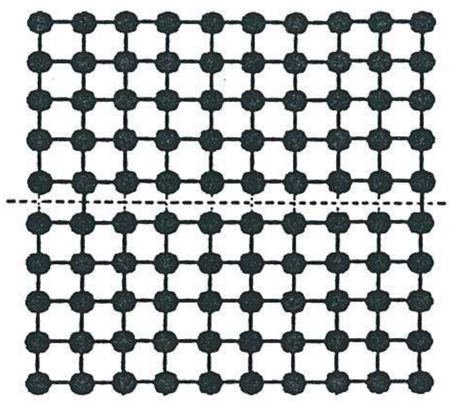


Fig.3a Grid graph

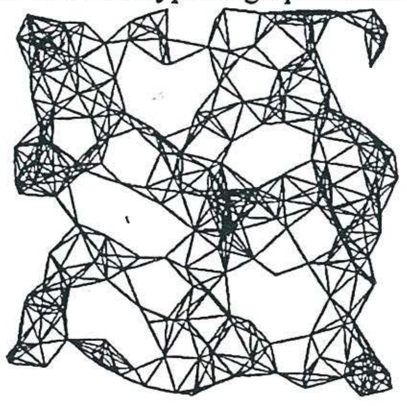


Fig.3b Geometric random graph

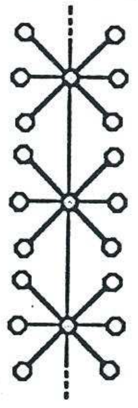


Fig.3c Caterpillar graph

4. Hypergraphs representing real circuits labelled by *ICn* [12]. The global optima is not known. The hypergraphs can be also specified by pair nodes/edges: *IC67/138, IC151/419, IC116/329, Fract149/147.* The structure of these circuits can be characterized as a random logic.

## 6.2 Results of the hypergraph and graph bisectioning

We have performed various experiments to demonstrate the efficiency of the developed algorithm KBOA comparing to the original BOA algorithm.

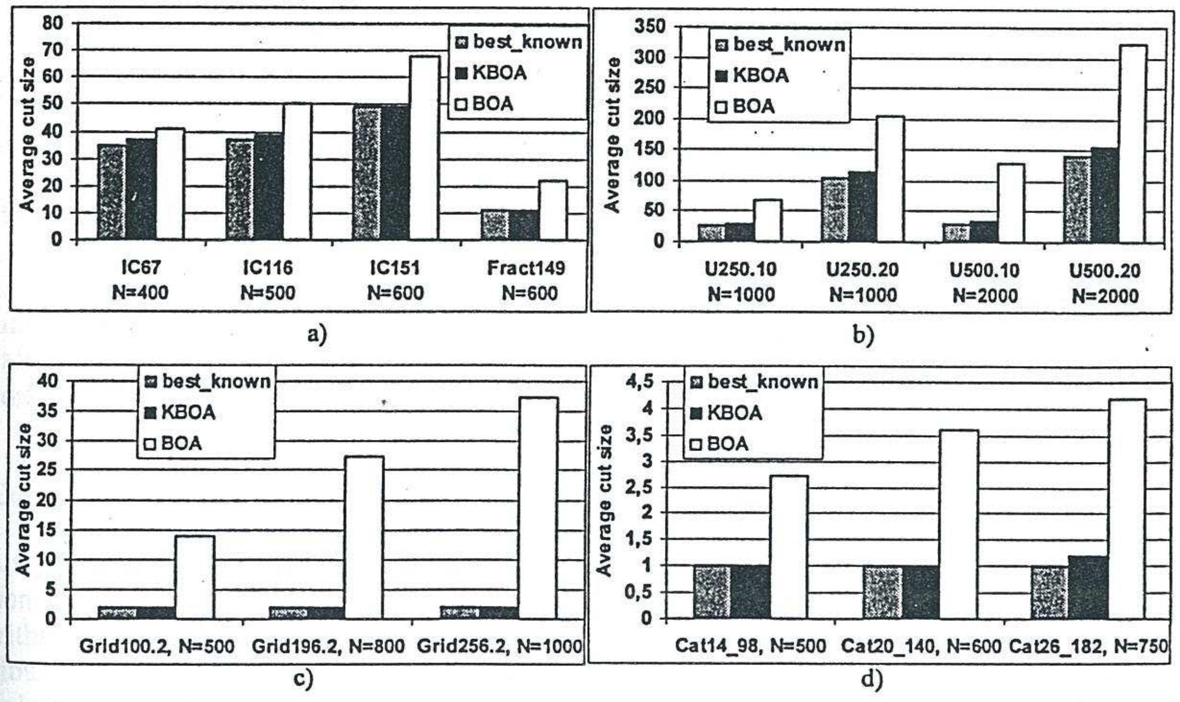


Fig.4 Average and best known cut size for BOA and KBOA a) on real hypergraphs, b) on geometric random graphs c) on grid graphs, d) on caterpillar graphs

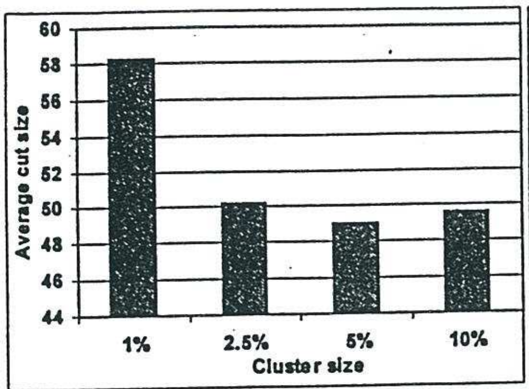


Fig. 5 Cluster size versus average cut size for IC151

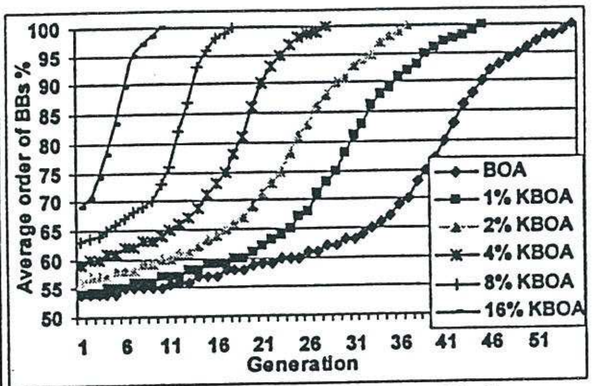


Fig. 6 Evolution of building blocks

In Fig. 4 the comparison of BOA and KBOA is shown for four groups of graphs. The best known solutions are determined by KBOA itself or well known hMetis program[13]. We performed ten independent runs for KBOA and BOA on each of the 14 test graphs of various type and size. The population size is the same for BOA and KBOA and it is set to the value for which the KBOA was successful for all ten runs for the case of grid graphs. Generally, for other types of graphs the population size was increased linearly with the size of the problem (as suggested in [8]). The KBOA outperforms BOA in all tests of hypergraph bisectioning. The time consumption for both algorithms is comparable due to equal population size. Let us note that BOA algorithm is theoretically capable to reach the similar cut size as KBOA but with about five times greater population size (for our test graphs) and thus with a rapid increase of computation time.

In Fig. 5 the performance of KBOA algorithm on the hypergraph IC151 for different sizes of injected clusters is shown. The size of clusters is expressed as a percentage of the number of hypergraph nodes. The proper value about 5% is used in all our trials. In the case of the smaller cluster injection, the growth of proper building blocks is weak and the opposite case of the injection of too large clusters leads into local optima.

In Fig.6 the growth of average value of building blocks order (ABB) in the population is depicted. This experiment is done for grid graph Grid100.2 with cluster size as a parameter. For the experiment the population size is set to N=300-2400 for KBOA and to N=2400 for BOA to get the global optimum for all sizes of cluster. Because two symmetrical optimal solutions exist, we separate all the chromozomes into two groups according to their similarity to each of the possible solution and the calculation of ABB based on the Hamming distance is summarized. It is clear that for small cluster size and BOA the average order of BBs in the first generation is about 50% which is typical for initial random population. A strong influence of the cluster size on the speed up of the evolution is evident. Let us note that in the 1% KBOA the cluster injection is not used and it is affected only by the phenomenon of prior probability (penalization) of the Bayesian network (see chapter 5) and only this phenomenon differentiates 1% KBOA from the BOA.

## 7 Conclusions and future works

In this paper we presented KBOA algorithm as a promising enhancement of the original version of probabilistic model-building genetic algorithm BOA [8], [9], [10]. The both algorithms use Bayesian network to model multivariate data as a mean of estimation of distribution of promising solutions. Although the BOA is powerful tool for solving hard optimization problems inclusive class of deceptive problems there are open questions to be solved e. g. the time complexity for very large problems, the setting of population size and parameter $k$ (the complexity parameter of BN) for current problem, etc.

Our goal was to incorporate a problem knowledge into the whole process of BN construction. Firstly we applied the prior probability of the Bayesian network expressed by the term $p(B|\xi) = c\kappa^{\delta}$. The essence of this approach lies in the penalization of edges of BN having no match in the hypergraph. The influence of this phenomenon can be recognized in Fig.6 on the comparison of 1%KBOA and BOA as a slight increase of convergence speed for 1% KBOA. The concept of cluster injection into the initial population detected on the hypergraph structure seems to be a really promising tool for enhancement of the population genotype. This phenomenon leads to the meaningful reduction of cost function and population size. This approach can be used for another optimization problems but with particular problem knowledge.

Future activities will be directed towards an efficient construction of the dependency graphs using the techniques of parallel processing and on a hybridizing of KBOA algorithm using local improvement to enhance genotype not only in the initial population but also during the evolution.

**Acknowledgements**

This research has been carried out under the financial support of the Research intention No. CEZ: J22/98: 262200012 – "Research in information and control systems" and it was also supported by the Grant Agency of Czech Republic grant No. 102/98/0552 "Research and Application of Heterogeneous Models".

**References**

[1] Johnson, D. S., Aragon, C., McGeoch, L., & Schevon, C.: Optimization by Simulated Annealing: An experimental Evaluation, Part 1, Graph Partitioning, Operations Research, vol.37, pp. 865-892, 1989.

[2] Alpert, C. J., Kahng, A. B.: Recent Directions in Netlist Partitioning: A Survey, Integration: The VLSI Journal 19 (1995), pp. 1-81.

[3] Bui, T. N., Moon, B. R.: Genetic Algorithm and Graph Partitioning. IEEE Transactions on Computers, Vol.45, No.7, July 1996, pp. 841-855.

[4] Goldberg, D. E.: Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, 1989.

[5] Pelikan, M., Goldberg, D. E., & Cantú-Paz, E.: Linkage Problem, Distribution Estimation, and Bayesian Networks. IlliGal Report No. 98013, November 1998,pp. 1-25.

[6] Muehlenbein, H., Rodriguez, A. O.: Schemata Distributions and Graphical Models in Evolutionary Optimization. GMD Forschungs Zentrum Informationstechnik, 53754-St. Augustin, 1998, pp.1-21.

[7] Pelikan, M., Goldberg, D. E., & Lobo, F.: A Survey of Optimization by Building and Using Probabilistic Model, Illigal Report 99018, September 1999, pp. 1-12.

[8] Pelikan, M., Goldberg, D. E., & Cantú-Paz, E.: Bayesian optimization Algorithm, Population Sizing, and Time to Convergence, Illigal Report 200001, January 2000, pp. 1-13.

[9] Pelikan, M.: A Simple Implementation of Bayesian Optimization Algorithm in C++(Version1.0). Illigal Report 99011, February 1999, pp. 1-16.

[10] Schwarz, J., Očenášek, J.: Experimental Study: Hypergraph Partitioning Based on the Simple and Advanced Genetic Algorithm BMDA and BOA, In: Proceedings of the Mendel'99 Conference, Brno University of Technology, Faculty of Mechanical Engineering, Brno, 1999, pp. 124-130, ISBN 80-214-1131-7.

[11] Heckerman, D., Geiger, D., & Chickering, M. (1994). Learning Bayesian Networks: The combination of Knowledge and Statistical Data (Technical Report MSR-TR-94-09, Redmont, WA: Microsoft Research, 1995, pp. 1-53.

[12] A benchmark set, University of California, Los Angeles, VLSI CAD Laboratory, http://vlsicad.cs.ucla.edu/~cheese/benchmarks.html.

[13] Karypis, G., Kumar, V.: Hmetis - A Hypergraph Partitioning Package, version 1.5.3, University of Minnesota, Department of Computer Science&ENGINEERING, Army HPC Research Center Minneapolis, http://www-users.cs.umn.edu/~karypis/metis/hmetis/download.shtml.

Schwarz Josef, Očenášek Jiří:

Partitioning-oriented placement using advanced genetic algorithm BOA.

A-5

# MENDEL 2000

6th International Conference on Soft Computing

Evolutionary Computation, Genetic Programming, Fuzzy Logic, Rough Sets, Neural Networks, Fractals, Bayesian Methods

June 7–9, 2000, Brno, Czech Republic

# PARTITIONING-ORIENTED PLACEMENT BASED ON ADVANCED GENETIC ALGORITHM BOA

Josef Schwarz
Jiří Očenášek

Technical University of Brno
Faculty of Engineering and Computer Science
Department of Computer Science and Engineering
CZ - 61266 Brno, Bozetechova 2
e-mail: schwarz@dcse.fee.vutbr.cz
e-mail: ocenasek@dcse.fee.vutbr.cz

*Abstract: This paper deals with an adaptation of the genetic partitioning algorithm BOA, based on the estimation of distribution of promising solution, for the placement of the hypergraph nodes into the regular structure of allocations. This task is a simplification of the placement problem encountered in PCB design or more complex case of physical layout of gate array on VLSI chip level. We present top-down placer based on the recursive bisectioning of the hypergraph/circuits. The hypergraph is repeatedly divided into densely connected subgraphs so the number of nets among them is minimized. We compare performance of our placer with the one based on the hybrid GA algorithm and Breuer's force-directed algorithm on artificial and real hypergraphs.*

*Key words: placement problem, hypergraph bisectioning, simple and advanced GA, estimation of distribution, BOA algorithm, Bayesian network, top-down placer.*

## 1 Introduction

With modern technology the circuits used nowadays become more and more complex and their design must be based on sophisticated methods. In the printed circuit boards PCB and VLSI design process five steps are offered: specification, logic design, physical design, fabrication and testing. The physical design phase is an important part of this process. It can be formulated as a mapping of circuit description into a physical layout in a target technology. The circuit is often represented in the form of a netlist. The resulted layout describes the geometric representation of all components of the circuit and shapes of the interconnection wires. Multiple, competing criteria have to be optimized with a large number of non-trivial constraints. The main concern is to find a layout with routable nets and fulfil constraints on the timing delay of critical nets/signals. The important phase of physical design is the placement of circuit elements/cells into the layout area.

### 1.1 Survey of heuristics

Many different techniques are used to solve the placement problem. Min-cut placement methods use the known strategy of divide and conquer, recursively applying min-cut bisection to embed the circuit into layout carrier [1], [2], [3]. Numerous other placement methods have been proposed including force-directed [4] as well as simulated annealing [5]. Recently a new enhancement of min-cut partitioning(useful for placement) was published which validates the multilevel partitioning paradigma for hypergraphs with efficient implementation [6]. A renewed bisection and quadrisection method for standard cell placement is described in [7], [8]. Each of the methods can be adapted for many layout styles-PCB, standard cells, sea of gates, gate arrays and macro-cells.

### 1.2 Fuzzy logic for VLSI CAD

Fuzzy set theory has been applied in many areas of engineering and science. Application of fuzzy controllers in control system and commercial applications are well known. Computer-Aided VLSI design represents a complex hierarchical structure of design phases with multiple competing objectives and various constraints. The majority of algorithms used are heuristics that are based on the human knowledge about the problems. To express such a knowledge we can use fuzzy logic with its linguistic terms. One of the typical application is the multiobjective decision making [9]. In the process of cell allocation a set of rules is investigated resulting in deciding whether the placement including the cell is acceptable. Fuzzy logic is also able to transform the multiobjective function which is naturally a vector function to a scalar function that serves as a simple knowledge for making decision in iterative placement procedures.

Another contribution to VLSI design is in [10] where a Fuzzy partitioning system FPS is described. The partitioning system employs two basic data structures, cells and net array. To each cell the set of associated net is specified and for each net a set of its cells is declared. To each net four parameters are specified: cell-cell connectivity $C_c$, cell net connectivity $C_n$, net-associativity $N_a$ as number of cells associated to the net and $N_w$ representing the weight of the net. These four parameters enter in the inference engine which computes the two outputs - cell and net index. This crisp values enter into the classical non-fuzzy procedure which forms initial clusters and then coalesces them into bigger cluster. The FPS system was tested on the benchmark circuits of the ISCAS89 suite. The FPS is comparable to well known Fiduccia-Mattheyses iterative-improvement techniques but only for smaller problem size up to 500 nodes. Because of very low time complexity of FPS algorithm it can serve as a quick procedure producing a good initial solution for another iterative algorithms e.g. for a placer based on the partitioning techniques.

### 1.3 Genetic algorithms

Genetic algorithms are often used to solve hard discrete optimization problems due to their robustness and ability to find a few suboptimal solutions concurrently what allows to choose a proper solution under additional criterion. The VLSI macrocell placement problem which lies in allocation of macrocells of different shapes and size on the chip carrier minimizing its area appears often in chip layout. To solve this hard problem, placement of objects on the free plane a hybrid genetic algorithms were developed [11], [12] using binary and/or binary slicing tree for encoding of the solution. A simpler problem seems to be the placement of objects into discrete positions of the PCB or gate array. In the past we designed and tested simple [13], [14], and parallel [15] genetic algorithm for PCB layout style with path type chromozome for problem size up to 150 circuits elements. For larger problems we found that it is useful to accept the recursive decomposition of circuit to reduce the problem size. Instead of often used heuristic we attempted to investigate a new approach of bisectioning-oriented placer based on the genetic partitioner BOA published recently in [16].

## 2 The placement problem

The placement problem solved in this paper is focused namely on PCB layout style which allows to compare performance of our placer with the placer based on the hybrid genetic algorithm [13] and older but very interesting force-directed placement algorithm [4]. Let us note, that the proposed algorithm is adequate for gate array layout too. The problem of circuit placement can be formalized as follows: Let us assume a hypergraph $H=(V,E)$, representing a circuit with $n=|V|$ nodes coresponding to a $n$ netlist elements (cells, gates), and $m = |E|$ edges corresponding to signal nets and a position graph $G=(P,D)$ representing) $p= |P|$ locations and $d = |D|$ edges connecting the adjacency positions with unit distance. The positions (slots) are organized upon the layout styles. We consider the layout style of PCB and gate array, see Fig.1. The set of slots forms a regular structure of locations for circuit elements (represented by circles) with $c$ columns and $r$ rows. The external signals are connected to the connector (represented by rectangles). In case of gate array one segment of connector is assigned to each row and column of locations. In case of PCB only one row of connector segments is used.
The placement problem can be defined as a placement of hypergraph nodes $V$ into a regular structure of discrete locations $P$ in two-dimensional plane called carrier. The goal is to find one-to-one mapping $f_p : V \rightarrow P$ such that the objective function is optimized. The placement is legal if elements are not overlapped and are placed to prescribed locations.

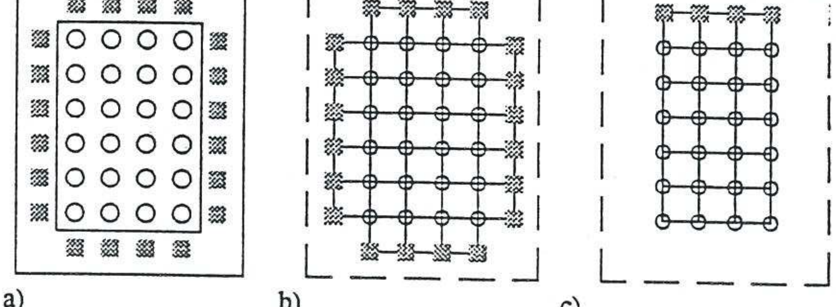


Fig.1 Layout style a) for gate array; the associated position graph b) for gate array, c) for PCB.

In our case the goal is to minimize the total net length. The length of nets is estimated by the half perimeter (HP) of the minimum enclosing bounding box of the placed elements of each net. It is easy to compute and it is often used as standard measure of the net complexity. A better model of the net length is the minimal spanning tree MST.

## 3 The BOA partitioner

The BOA partitioner [16] for hypergraph bisectioning is based on the BOA implementation published in [17]. The particular bisectioning problem is defined as follows: Let us assume a hypergraph $H=(V,E)$, with $n = |V|$ nodes and $m = |E|$ edges. We are supposed to find such a bisection $(V1,V2)$ of $V$ into equal sized parts ( $/V1/ = /V2/$, $V = V1 \cup V2$ ) that minimizes the number of hyperedges that have nodes in different set $V1$, $V2$. The set of external hyperedges can be labelled as $E_{cut}\,(V1,V2)$. The objective function is the number of external hyperedges, shortly called cut size:

$$c\,(V_1,V_2) = /E_{cut}\,(V_1,V_2)/ = /\{e \in E \,/\, e \cap V_1 \neq 0,\ e \cap V_2 \neq 0\}/ \qquad (1)$$

It can be proven that the sum of half perimeters equals the sum of cut size (canonical set of horizontal and vertical cuts placed between neighbouring columns and rows) which is often used as a criterion in decomposition algorithms [1].

The BOA pertains into the field of Estimation of Distribution Algorithms (EDAs)[18] often called probabilistic model-building genetic algorithm, where crossover and mutation operators are replaced by probability estimation and sampling techniques. It uses statistical information contained in the set of selected parents to detect gene dependencies. The estimated probability model is then used to generate new promising solutions according to this distribution. The process can be described as follows:

```
Generate initial population of size N (randomly);
While termination criteria is false do
begin
  Select parent population of M individuals according to fitness function  (M≤N);
  Estimate the distribution of the selected parents and construct the Bayesian network BN;
  Generate new offspring according to the estimated model and BN network;
  Replace some individuals in current population by generated offspring;
end
```

In BOA the Bayesian graph/network encodes the structure of a problem. The bisection of the hypergraph is encoded as a binary string $X=(X_0, X_1, ..., X_{n-1})$. Variable $X_i$ be equal to zero or one according to assigning of *i-th* node of hypergrah to one or second partition. The variables are not independent due the phenomenon of epistasis. For each variable $X_i$ it is defined a set of variables $\Pi_{X_i}$ called parents it depends on, so the probability distribution of individuals is encoded as

$$p(X) = \prod_{i=0}^{n-1} p\left(X_i \mid \Pi_{X_i}\right) \qquad (2)$$

As an illustration the six-nodes elementary hypergraph is considered for the bisectioning and one associated BN network is shown in Fig. 2. Using a contingency table (including bivariate marginal probability of variables in current population) each variable using conditional probability can be generated sequentially beginning from $X_1$ to $X_6$.

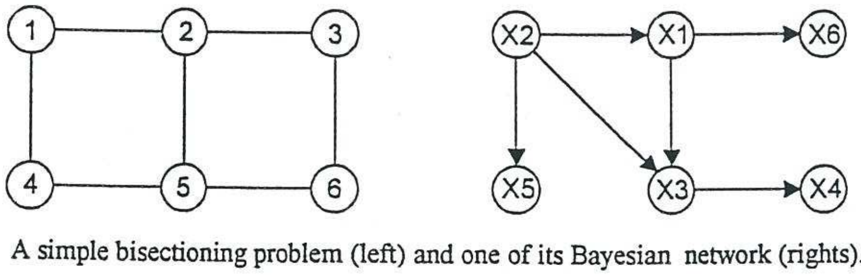


Fig.2. A simple bisectioning problem (left) and one of its Bayesian network (rights).

Notice, the existence of oriented edge from $X_j$ to $X_i$ in the network implies that the variable $X_j$ belongs to the set $\Pi_{X_i}$. To reduce the space of possible networks, the number of incoming edges into each node is limited to k=3.

## 4 The top-down recursive placer

The idea of top-down recursive bisectioning-oriented placement is based on repeated division of a given circuit/hypergraph into subhypergraphs to optimize a given bisectioning objective e.g. cut size $c\,(V_1,V_2)$. With each bisectioning of the circuit, the given layout area/carrier is bisected in either the horizontal or the vertical direction. Each subhypergraph is assigned to a partition. This process is repeated recursively until the subgraph has only one node/elements - the element can be mapped to an unique position on the carrier. When bisectioning a subhypergraph it is necessary to account internal nets in the current subhypergraph as well as the external connector segments and other elements in another higher-level bisection. This concept called terminal propagation was involved by Dunlop and Kernighan [19] that adds to the current netlist dummy elements that are fixed in the currently processed bisection. In our approach these elements remains in the centre of their higher-level block and are included in the calculation of half perimeter of all nets associated to current element. The top-down recursive bisectioning algorithm is described as follows:

```
Push the basic block representing original placement problem onto the queue;
While queue of blocks is not empty do
begin
   Pull the block from queue;
   if Block is trivial (includes only one element) then
     Map the element to unique position else
     begin
        Specify internal netlist and dummy elements;
        Apply BOA bisectioner for block bisectioning into two subblocks;
        Push each block onto the queue;
     end
end
```

In Fig.3 the process of recursive bisectioning of layout area is shown. The original block is divided by vertical cut into two blocks B1, B2. In the blocks two free elements V and W are shown. The block B1 is divided into block B11 and B12 by horizontal cut. During the cutting the element W is taken as dummy element and placed in the centre of B2. When block B2 is divided by horizontal cut into blocks B21 and B22, the element V in the centre of B11 serves as dummy element that influences the shift of the element W into block B21. Finally the division of block B11 into B111 and B112 is shown resulting in the shift of the element V into block B112 closer to element W. The effect of dummy modules in our implementation is replaced by minimization of HP associated to elements V and W.

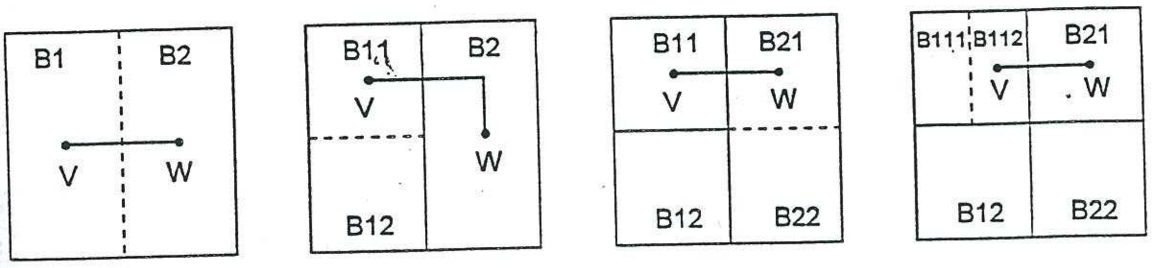


Fig.3 Example of cut sequence and dummy elements phenomenon.

## 5. Experimental results

### 5.1 Test graphs

The two types of graph structures are used:

1. Grid graphs *Rn, Rn.B* with grid structure [2], [16] where the notion *n* specifies the number of nodes and *B* the existence of bottleneck in graph structure. As an example a grid graph *RL100* is represented in Fig.4a and *RL100B* in Fig.4b having a 2-edge bottle-neck in the edge structure.
2. Hypergraphs representing real circuits labelled by *ICn* [4], [16], [20]. The global optima is not known. The hypergraphs can be also specified by pair nodes/edges*: IC67/138, IC151/419.*

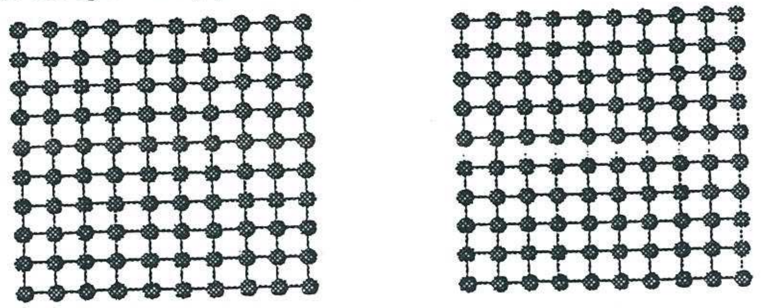


Fig.4 Grid graphs a) RL100 with full grid structure, b) RL100B with the bottleneck

The parameters of the placement problem solved are in the Table 1. The fixed elements are placed into the connector segments. The number of the rows includes one row of the connector segments.

| Circuit | Parameters | | | | | |
|---|---|---|---|---|---|---|
| | Elements | Free elements | Nets | Connector segments | Rows | Columns |
| RL64 | 64 | 56 | 112 | 8 | 8 | 8 |
| RL64B | 64 | 56 | 106 | 8 | 8 | 8 |
| RL100 | 100 | 90 | 180 | 10 | 10 | 10 |
| RL100B | 100 | 90 | 172 | 10 | 10 | 10 |
| IC67 | 67 | 52 | 138 | 15 | 5 | 15 |
| IC151 | 151 | 136 | 419 | 15 | 11 | 15 |

Table.1. Parameters of placement problems.

## 5.2 Performance of the placer

To test the BOA placer some experiments were done. Six circuits was used to compare the performance, see Table 2, of our BOA placer with a classical but sofisticated genetic algorithm GPLACE [13] and force-oriented placement algorithm [4]. The results of FORCE algorithm are relevant to parameter epsilon=0.001 which specifies very small final level of the force system equilibrium leading to very good results. The time complexity is expressed only by number of cycles; each cycle represents one step of the numerical solution of a system of *n* nonlinear equations that models the shift of *n* nodes along the free plane. For each benchmark five independent trials were done and the average values are presented.
The GPLACE genetic algorithm was performed with best known parameter values: in case of graphs IC67 and IC151 the number of population $N_p=30$, offspring rate $R_o=0.3$, mutation rate $R_m=0.02$ are used and inversion operator was switched off. Each tenth generation ten pairwise exchanges of genes leading to cost improvement are done. The stopping criterion is activated when during the epoch of 500 generation the total length of nets represented by HP is not changed. In case of regular graphs the inversion operator an dynamical setting of mutation must be activated. The BOA partitioner as the genetic core of BOA placer is set in the following way: the population size *N* is taken from the interval <650, 1500> as a minimal successful value with respect to the conclusion done in [16], the 50% selection truncation is used ($M/N= 0.5$), the maximal degree of *BN* nodes $k= 3$. The experiments were done on the 100MHz PC computer.

| | GPLACE | | | | | FORCE | | BOA | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Circuit | n | HP | MST | Ng | TIME | MST | Cycles | HP | MST | Time | N |
| RL64 | 64 | 126 | 126 | 4818 | 5 | - | - | 112 | 112 | 3.0 | 800 |
| RL64B | 64 | 127 | 127 | 10400 | 10 | - | - | 106 | 106 | 2.6 | 650 |
| RL100 | 100 | 180 | 180 | 33840 | 26 | - | - | 180 | 180 | 18 | 1600 |
| RL100B | 100 | 188 | 188 | 41443 | 32 | - | - | 172 | 172 | 10 | 1000 |
| IC67 | 67 | 639 | 709 | 1254 | 2.5 | 713 | 184 | 652 | 734 | 3.7 | 650 |
| IC151 | 151 | 2227 | 2304 | 2087 | 17 | 2025 | 388 | 2060 | 2105 | 64 | 1000 |

Table 2 Comparison of the placers : n - number of nodes (size of problem), $N_g$ - number of generations, N-population size for BOA algorithm, HP - half perimeter (cost), MST- minimal spanning tree, TIME - computational time in minutes, cycles - number of steps of the numerical optimization process.

## 6 Conclusions

We have developed a top-down placer based on the BOA recursive partitioner. From Table 2 it is evident that the BOA placer produces global optimum for grid graphs that serve as hard benchmarks. For the hybrid genetic algorithm GPLACE it is difficult to find the global optimum for the grid graphs and all the sofisticated tools of algorithm have to be activated, namely the dynamical setting of the mutation and the inversion operator. In case of random logic - hypergraphs IC67 and IC151 the results are comparable in MST value for all the methods. The time consumption of BOA placer depends namely on the problem size and is mainly influenced by the first bisectioning of the original hypergraph. The time complexity of the BOA placer can be estimated as $O(n^{5/2})$ for n<100. Let us notice that the only minimization of MST value can cause wire congestion in some local area of the layout - it holds namely for the GPLACE and FORCE algorithm. From the nature of BOA placer follows that the probability of the congestion is minimized due to minimization of cut size between each pair of currently divided blocks. Our contribution can be seen namely in the extension of the BOA application and acknowledgement of BOA robustness in optimization of discrete combinatorial problems. The main problem still lies in proper choice of cut lines - we used a slightly modified standard approach based on the periodical change of vertical and horizontal cuts. The future work will be focused namely on an exploration of the BOA or BMDA placer to a quadrisection version which can solve cut ordering problem. Other activities will be directed towards the usage of BOA placer for the other layout styles.

**Acknowledgement**

This research has been carried out under the financial support of the Research intention No. CEZ: J22/98: 262200012 – "Research in information and control systems" and it was also supported by the Grant Agency of Czech Republic grant No. 102/98/0552 "Research and Application of Heterogeneous Models".

## References

[1] Breuer M. A.: Min-cut Placement, Journal of Design Automation and Fault Tolerant Computing, Vol. 1, No.4., Oct.1977, pp. 343-362.

[2] Schwarz J.: Design Automatization of Composition and Placement of Integrated Circuit. PhD. Theses, Technical University of Brno, Faculty of Engineering and Computer Science, Department of Computer Science and Engineering, Brno, 1993, 136 pp., in Czech.

[3] Fiala P.: Decomposition Placement Methods, Diploma Project, UIVT, FEI, VUT Brno, 1992, in Czech.

[4] Breuer M. A.: Forced-oriented Placement Algorithm, Experimental Study with IC Benchmarks, Chapter 5 and 6, pp. 89 - 162, delivered under personal request.

[5] Schwarz, J.: Simulation oriented placement methods for VLSI chips, In: Proceedings of the MOSIS'93 conference, MARQ Ostrava, Olomouc, 1993, pp. 295-300.

[6] Karypis, G., Kumar, V.: Hmetis - A Hypergraph Partitioning Package, version 1.5.3, University of Minnesota, Department of Computer Science&ENGINEERING, Army HPC Research Centre Minneapolis, http://www-users.cs.umn.edu/~karypis/metis/hmetis/download.shtml.

[7] A. E. Caldwell, A. B. Kahng and I. L. Markov, "Can Recursive Bisection Produce Routable Placements?", to appear in Proc. Design Automation Conf., Los Angeles, June 2000.

[8] D. J. Huang and A. B. Kahng, "Partitioning-Based Standard-Cell Global Placement with an Exact Objective", Proc. ACM/IEEE Intl. Symp. on Physical Design, Napa, April 1997, pp. 18-25.

[9] Sharagowitz E., et al.: Application of Fuzzy Logic i Computer-Aided VLSI Design, IEEE Transactions on Fuzzy Systems, Vol. 6, No.1, February 1998, pp.163-172.

[10] Manzoul M. A.., Valavala K.R.: A Fuzzy Partitioning System, IEEE Micro, December 1995, pp. 1-8.

[11] Esbensen H.: A genetic Algorithm for Macro Cell Placement, Proceedings of the European Design Automation Conference, pp. 52-57, September 1992.

[12] Schnecke V.: Hybrid Genetic Algorithm for Solving Constrained Packing and Placement Problems, PhD. Theses, University of Osnabrueck, pp. 1- 186.

[13] Bartoš J.: Genetic Algorithm for the Placement Optimization, Diploma Project, Technical University of Brno, Faculty of Engineering and Computer Science, Department of Computer Science and Engineering, Brno,1994, 66 pp., in Czech.

[14] Schwarz J.:GPLACE-Genetic Algorithm for Placement Optimization. I. International Conference of Genetic Algorithm on the Occasion of 130th Anniversary of Mendel's Law, Brno, September 26-28,1995, pp.139-144.

[15] Schwarz, J.: Experimental Study on Parallel Genetic Algorithm for Placement Optimalization, Mendel '97, Technical University of Brno, Faculty of Mechanical Enginnering, Brno, Czech Republic, 1997, pp. 148-153, ISBN 80-214-0884-7.

[16] Schwarz, J., Očenášek, J.: Experimental Study: Hypergraph Partitioning Based on the Simple and Advanced Genetic Algorithm BMDA and BOA, Proceedings of the Mendel'99 Conference, Brno University of Technology, Faculty of Mechanical Engineering, Brno, 1999, pp. 124-130, ISBN 80-214-1131-7.

[17] Pelikan, M.: A Simple Implementation of Bayesian Optimization Algorithm in C++(Version1.0). Illigal Report 99011, February 1999, pp. 1-16.

[18] Pelikan, M., Goldberg, D. E., & Cantú-Paz, E.: Linkage Problem, Distribution Estimation, and Bayesian Networks. IlliGal Report No. 98013, November 1998, pp. 1-25.

[19] Dunlop A. E., Kernighan, A Procedure for Placement of Standard Cell VLSI circuits, IEEE Transactions on Computer-Aided Design 4(1) (1985), pp. 92-98.

[20] A Benchmark Set, University of California, Los Angeles, VLSI CAD Laboratory, http://vlsicad.cs. ucla.edu /~ cheese/benchmarks.html.

Schwarz Josef, Očenášek Jiří:

Evolutionary Multiobjective Bayesian Optimization Algorithm: Experimental Study.

A-6

Proceedings of the 35th Spring International Conference

# Modelling and Simulation of Systems

Edited Jan Štefan

Ostrava 2001

# Evolutionary Multiobjective Bayesian Optimization Algorithm: Experimental Study

Josef Schwarz*
schwarz@dcse.fee.vutbr.cz
Jiří Očenášek*
ocenasek@dcse.fee.vutbr.cz

**Abstract:** This paper deals with the utilizing of the Bayesian optimization algorithm (BOA) for multiobjective optimization of hypergraph partitioning. The main attention is focused on the incorporation of the Pareto optimality concept. We have modified the standard algorithm BOA for one criterion optimization according to well known niching techniques o find the Pareto optimal set. This approach was compared with standard weighting techniques and the single optimization approach with the constraint. The experiments are focused mainly on the bi-objective optimization because of the visualization simplicity.

**Key Words:** Multiobjective optimization, evolutionary algorithms, Bayesian optimization algorithm, Pareto set, niching techniques, hypergraph bisectioning.

## 1 Introduction

Many real-world problems have multiple often competing objectives. While in the case of single-objective optimization the optimal solution is simply distinguishable, this is not true for multiobjective optimization. Historically, multiple objectives have been combined to form a scalar objective function through weighted sum of individual objectives or by turning objectives into constraints. But setting of weights and specification of penalty functions is not a simple task and these values can be found only experimentally. The better approach lies in finding all possible trade-offs among the multiple, competing objectives. These solutions are optimal, nondominated, in that there are no other solutions superior in all objectives. These so called Pareto optimal solutions lie on the Pareto optimal front. There are many papers that present various approaches to find of Pareto optimal front almost based on the evolutionary algorithms. Let us mention here the well known niched Pareto genetic algorithm NPGA [1]. A wide review of basic approaches and the specification of original Pareto evolutionary algorithms includes the dissertation [2], [3], [4] where the last one describes the original strength Pareto optimization algorithm SPEA. From the last period let us mention an interesting Pareto-Envelope based Selection Algorithm PESA [5] which might outperform the very good algorithm SPEA.

All of these capable algorithms based on evolutionary algorithms (EA) have the common disadvantage - the necessity of ad hoc setting of parameters like crossover, mutation and selection rate. That is why we have analyzed and used one of the Estimation of Distribution Algorithms (EDAs). These algorithms also called probabilistic model-building genetic algorithms have attached a growing interest during the last few years because crossover and mutation operators used in standard GA are replaced by probability estimation and sampling

* Department of Computer Science and Engineering, Božetěchova 2, CZ-612 66 Brno

techniques to avoid the necessity to specify the set of GA parameters. We will focus on one of them - the Bayesian optimization algorithm [6], [7]. Recently we have published our experience with this algorithm in [8] where single criterion optimization of hypergraph bisectioning was described. In this paper we have focused on the bi-objective optimization of hypergraph bisectioning.

## 2 Problem specification

Hypergraph partitioning is a well known problem of graph theory. We have investigated a special case of *k*-way partitioning for *k=2* called bisectioning. If necessary the *k*-way partition can be found by recursive *2*-way bisectioning. The hypergraph model can be used for many application problems e.g. for system segmentation, network partitioning and VLSI layout.
The particular bisectioning problem is defined as follows: Let us assume a hypergraph $H=(V,E)$, with $n=|V|$ nodes and $m=|E|$ edges. The goal is to find such a bisection $(V1,V2)$ of $V$ that minimizes the number of hyperedges that have nodes in different set $V1$, $V2$ (1) and the difference/balance of the partition sizes (2). The set of external hyperedges can be labelled as $E_{cut}(V1,V2)$ and the following cost functions are defined:

$$C1(V_1,V_2)=|E_{cut}(V_1,V_2)|=|\{e\in E\,/\,e\cap V_1\neq\emptyset,\ e\cap V_2\neq\emptyset\}| \qquad (1)$$

$$C2(V_1,V_2)=||V_1|-|V_2|| \qquad (2)$$

For more formal specification of the problem, the following notation is used:

$P=(X_1,X_2,\ldots,X_N)$ with $X_j\in P$, is the population of the solutions/string/individuals
$X$ is a string/individual of the population $P$ the length of which is $n$
$X=(x_0,x_1,\ldots,x_{n-1})$ is a string/individual with $x_i\in\{0,1\}$
$C(X)$ is the cost function of the string $X$.

Each solution of the bisection is represented by a binary string $X=(x_0, x_1, \ldots, x_{n-1})$. The variable $x_i$ represents the partition number, the index specifies the node in the hypergraph. For the case of simple graph $G(V,E,R)$ bisectioning we have derived the following two cost functions on the binary string $X=(x_0,x_1,\ldots,x_{n-1})$ to be minimized:

$$C1=\sum_{\substack{i=0\\j>i}}^{n-1} r_{ij}(x_i+x_j-2x_ix_j) \qquad (3)$$

$$C2=\left|\sum_{i=0}^{n-1}x_i-\sum_{i=0}^{n-1}(1-x_i)\right|, \qquad (4)$$

where the coefficient $r_{ij}\in R$ equals to one in case the net/edge of the graph G exists between node $i$ and $j$, else $r_{ij}=0$. The cost $C1$ represents the cut value of the bisection and the cost $C2$ expresses the balance/difference of the partition sizes. There are three approaches how to solve this 2-objective optimization problem that will be described in the next chapters.

## 3 The BOA algorithm

The BOA algorithm is a population based evolutionary algorithm but the reproduction process of individuals is replaced by probability estimation and sampling techniques. It uses statistical information contained in the current population to detect multivariate parameter dependencies. The learned Bayesian network BN encodes a joint probability distribution based on the conditional probabilities; the BN quality is estimated by Bayesian-Dirichlet metrics. The estimated probability model is then used to generate new promising solutions according to this distribution using the sampling process. The BOA algorithm can be described as follows [7]:

```
Generate initial population of size N (randomly);
While termination criteria is false do
begin
  Select parent population of M individuals according to fitness function f(X) (M<N);
  Estimate the distribution of the selected parents and construct the Bayesian network BN;
  Generate new offspring according to the estimated model and BN network;
  Replace some individuals in current population by generated offspring;
end
```

## 4 Multiobjective BOA algorithm

A general multiobjective optimization/maximization problem MOP can be described as a vector function $f$ that maps a tuple of $n$ parameters to a tuple of $m$ objectives [4]:

$$\begin{aligned} &\max\ y = f(x) = (f_1(x), f_2(x), \dots, f_m(x)) \\ &\text{subject to } x = (x_0, x_1, \dots, x_{n-1}) \in X \\ &\qquad\qquad y = (y_1, y_2, \dots, y_m) \in Y, \end{aligned} \tag{5}$$

where $x$ is called decision vector, $X$ is the parameter space, $y$ is the objective vector, and $Y$ is the objective space.
The set of solution of MOP includes all decision vectors for which the corresponding objective vectors cannot be improved in any dimension without degradation in another - these vectors are called Pareto optimal set. The idea of Pareto optimality is based on the Pareto dominance. A decision vector $\boldsymbol{a}$ dominates decision vector $\boldsymbol{b}$ *iff* $f_i(a) \geq f_i(b)$ *for* $i=1,2,\dots, m$ with $f_i(a) > f_i(b)$ for at least one $i$. The vector $\boldsymbol{a}$ is called Pareto optimal if there is no vector $\boldsymbol{b}$ which dominates vector $\boldsymbol{a}$ in parameter space $X$.
In objective space the set of nondominated solutions lie on a surface known as Pareto optimal front. The goal of the optimization is to find a representative sampling of solutions along the Pareto optimal front. From the theory of Pareto optimal set it is evident that the optimization algorithms should be able to find as many Pareto optimal solutions as possible. The techniques how to do it lies in keeping the diversity using some of the niching techniques. Standard BOA is able to find mostly one optimal solution at the end of the optimization process, when the whole population is saturated by phenotype-identical individuals.
We have implemented one variant of Pareto BOA algorithms (Pareto BOA), one variant of non-Pareto weighted sum BOA (WSO) and non-Pareto single BOA (SOP).

### 4.1 Single BOA with the normalization (SOP)

In this approach only one objective function $f_1(X)=1/(C1(X)+1)$ is used and the second objective function $f_2(X)=1/(C2(X)+1$ is replaced by normalization operator which modifies each individual to keep its balance in the considered bound. This operation can naturally change a partly the objective function $f_1$ of each individual. This effect may cause an extra genetic drift of the population.

### 4.2 Weighted-sum BOA (WSO)

In this approach the original vector-valued objective function is replaced by a scalar-valued objective function. The objective function of the individual $X$ is computed as a weighted sum of all objective functions:

$$f(X) = w_1 f_1(X) + w_2 f_2(X), \tag{6}$$

where $w_1, w_2$ are weight coefficients. It is well known the sensitivity of the optimization process to these values. We have tested two sets of these coefficients. In WSO1 variant we have chosen $w_1=0.5$, $w_2=0.5$, in WSO2 couple of $w_1=0.005$, $w_2=0.995$ was used. These algorithms do not preserve Pareto-optimal solutions but provide mostly solutions from extremes of the Pareto front.

## 4.3 Pareto optimal BOA

The multiobjective optimization represents the difficult multimodal optimization problem which is mostly solved with niching methods that allow to preserve the diversity in the population of individuals/solutions. Our Pareto BOA algorithm is a modification of single BOA where we applied a promising niching techniques published in [4].
Although we solved bi-objective optimization, our algorithm is able to solve *m*-objective optimization problems. Our Pareto BOA algorithm can be described by the following steps:

Step 1: ***Initialization:*** Generate an initial population $P_0$ of size N randomly.

Step 2: ***Fitness assignment:*** Evaluate the initial population.

Step 3: ***Selection:*** Select the parent population as the best part of current population by 50% truncation selection.

Step 4: ***Model construction:*** Estimate the distribution of the selected parents using Bayesian network construction.

Step 5: ***Offspring generation:*** Generate new offspring (according to the distribution associated to the Bayesian network).

Step 6: ***Nondominated set detection and fitness assignment:*** Current population and offspring are joined, nondominated solutions are found, evaluated and stored at the top of the new population. Then dominated offspring and parents are evaluated separately.

Step 7: ***Replacement:*** The new population is completed by offspring and the best part of current population, so the worst individuals from current population are canceled to keep the size of the population constant.

Step 8: ***Termination:*** If maximum number of generations $N_g$ is reached or stopping criterion is satisfied then the last Pareto front is presented, else go to Step 3.

The most important part of our Pareto algorithm is the procedure for detection of nondominated solution (current Pareto front) and sophisticated fitness calculation. The procedure for current nondominated and dominated set detection is described in following steps:

1. For each individual $X$ in the population $P$ compute vector of the objective functions
$$\overline{f}(X) = (f_1(X), f_2(X), \ldots, f_m(X)) \tag{7}$$
2. Detect subset of nondominated solutions
$$\overline{P} = \{X_j \mid X_j \in P, \bar{\exists} X_k \in P : \forall l \in \{1\ldots m\} : f_l(X_k) > f_l(X_j) \tag{8}$$
Note: If two or more individuals have the same fitness vector $f(X)$, then only one of them is accepted.
3. For each nondominated solution $X_j$ compute its strength value as
$$s(X_j) = \frac{\left|\{X_k \mid X_k \in P, \forall l \in \{1\ldots m\} : f_l(X_j) > f_l(X_k)\}\right|}{|P|+1} \tag{9}$$
The fitness for nondominated solutions is equal to the reverse of the strength value $f'(X_j) = 1/s(X_j)$.

4. For each dominated solution $X_i$ determine the fitness as

$$f'(X_i) = 1 \Big/ \left(1 + \sum_{X_j} s(X_j)\right), \tag{10}$$

where $X_j \in P, \forall l \in \{1 \ldots m\} : f_l(X_j) > f_l(X_i)$. In the original approach [4] all individuals dominated by the same nondominated individuals have equal fitness. We proposed an extension by adding a term $c.r(X_i)/(|P|+1)$ into the denominator (10), where $r(X_i)$ is the number of individuals from $P$ (not only from nondominated solutions) which dominate $X_i$ and coefficient $c$ is set to very small number, for example 0.0001. This term is used to distinguish the importance of individuals in the same "niche" (being dominated by the same nondominated solutions).

This type of fitness evaluation has the following advantages:.

- For all nondominated individuals $f'(X_i) \geq 1$, for dominated individuals holds $f'(X_i) < 1$. If we use the replace-worst strategy, implicit Pareto elitism is included.
- Individuals from Pareto front dominated smaller set of individuals receive higher fitness, so the evolution is guided towards the less-explored search space.
- Individuals having more neighbours in their „niche" are more penalised due to the higher $s(X_j)$ value of associated nondominated solution.
- Individuals dominated by smaller number of nondominated individuals are more preferred.

## 5 Experimental results

### 5.1 Test graphs

The three types of graph structures are used [8]:

1. Hypergraphs representing real circuits labelled by *ICn*. The global optima is not known. The structure of circuits can be characterized as a random logic. The hypergraph IC67 consists of 67 nodes and 134 edges/nets, the IC116 consists of 116 nodes and 329 edges/nets.
2. Random geometric graph *Un.d.* on *n* vertices is placed in the unit square and its nodes coordinates are chosen uniformly. An edge exists between two vertices if their Euclidean distance is *l* or less, where the expected vertex degree is specified by $d = n\pi l^2$. We have chosen *n=120, d=5,* see Fig.1a.
3. Caterpillar graphs *CATk_n*, with *k* articulations, *(n-k)/k* legs for each articulation and *n* nodes, see Fig. 1b with *k=3* and *n=21.*

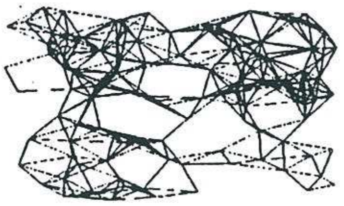


Fig.1a Geometric random graph

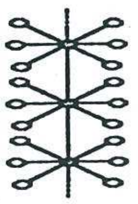


Fig.1b Caterpillar graph

### 5.2 Results of experiments

Let us notice that the objective space is visualized using the original cost function *C1, C2* instead of the objective functions $f_1, f_2$ (let us notice the cost functions *C1, C2* are minimized). Fig.2 shows the dynamics of the *IC67* bisection optimization. For the case of weighted sum algorithm the result fetched in 17-th generation of one run is shown. Population size is set to *N=2500*, the first population is generated randomly to keep the balance uniformly distributed in the range from 0 to 20% of *n*, where *n* is number of hypergraph nodes. The size of each point in the graph is proportional to the number of phenotypic equal solutions found in the current population. The current Pareto front are enlarged and pointed up.

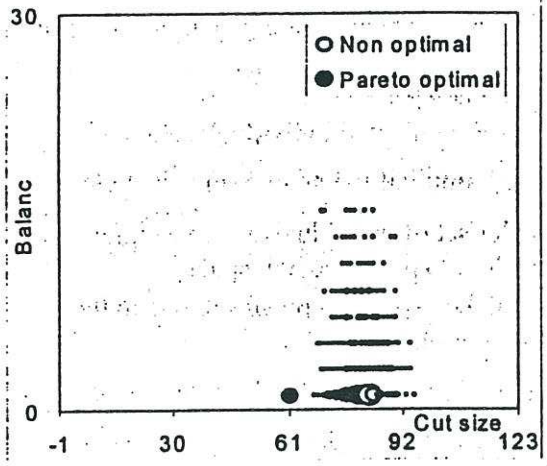


Fig.2a WSO1 algorithm

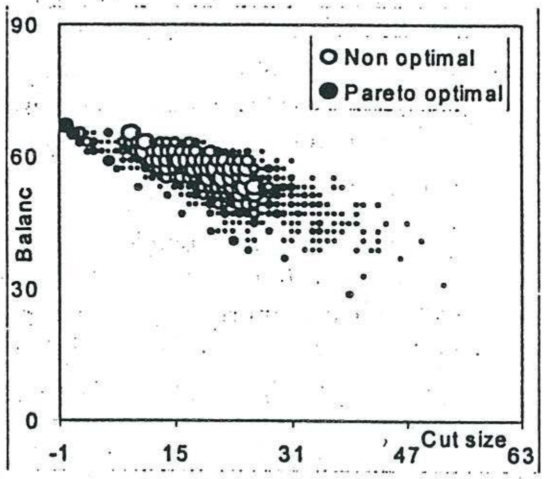


Fig 2b WSO2 algorithm

In Fig. 2a weighted sum algorithm WSO1 with $w_1$=0.5, $w_2$=0.5 is used, population is distributed with slight variability of balance. In Fig. 2b weighted sum algorithm WSO2 with $w_1$=0.995, $w_2$=0.005 is used, high balance of individuals is evident, the algorithm prefers more trivial solutions with minimum cut size and large balance. We used such values of $w_1$ and $w_2$ because even for $w_1$=0.99 and $w_2$=0.01 the algorithm still provides solution shown in Fig. 2a. In Fig. 3 the performance of Pareto, SOP and WSO algorithms for 3 types of graphs is shown.

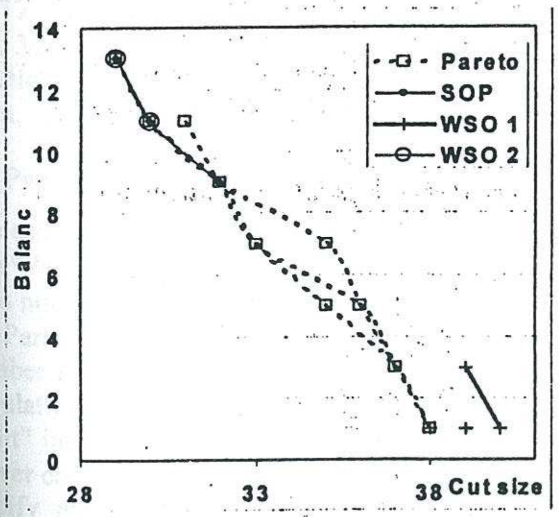


Fig.3a Bisection of *IC116, N=4000*

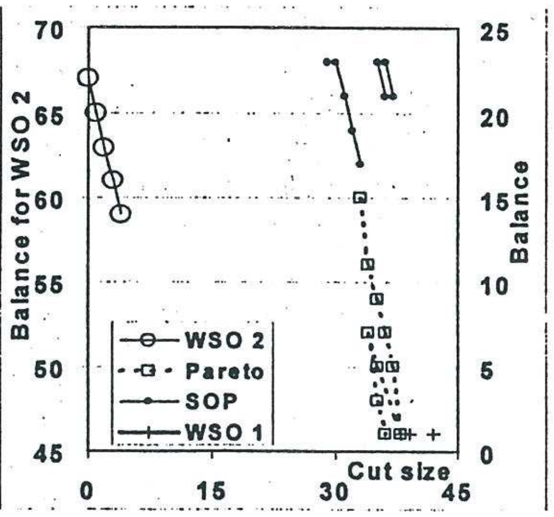


Fig.3b Bisection of *IC67, N=2500*

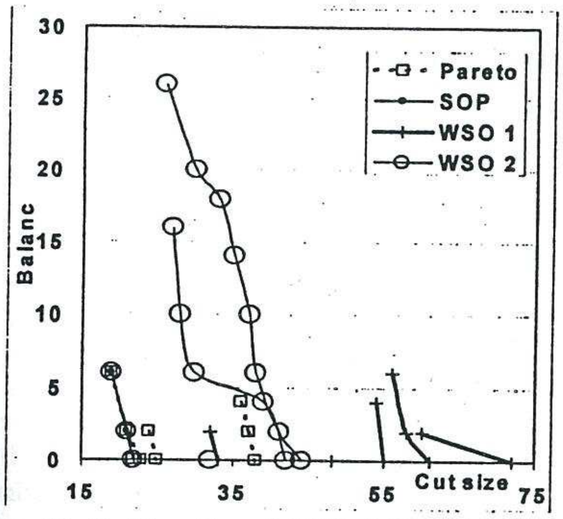


Fig.3c Bisection of *U120.5, N=4000*

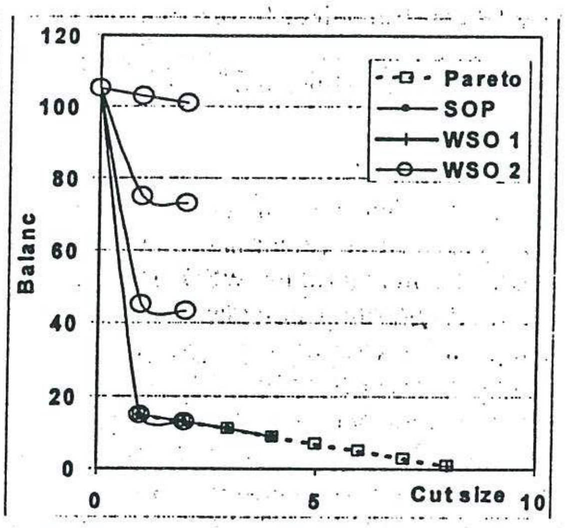


Fig.3d Bisection of *CAT7_105, N=3500*

The five independent runs of each algorithm were performed and five Pareto fronts from final populations are shown. For better visualization of the fronts from each run the points are connected by lines. Balance of the individuals in the initial population was uniformly distributed between zero and *0.2*n*, population size N was set proportional to *n*, the limit of the balance for normalization in SOP was set to maximum value *20%* of *n*. Maximum number of generations is set to $N_g = 200$.
From Fig. 3a it is evident that Pareto algorithm usually produces the largest Pareto set with good quality solutions, whereas SOP and WSO2 produce only solutions with low cut size but high balance and WSO1 produces solutions with low balance and higher cut value. From Fig. 3b the difficulty with WSO2 is evident - WSO2 produces trivial solutions with high balance. The SOP algorithm provides solution with small cut but high balance only. For example, solution with the cut-size=35 and balance=23 was obtained even if solution with lower balance for the same cut-size exists. The geometric graph *U120.5* seems to have many local optima. It is a hard benchmark as it is seen in Fig. 3c. Both WSO1 and WSO2 provide solutions far from optima in 4 runs from five runs. Only in one run the Pareto optimal solution was found. The SOP algorithm and Pareto BOA provide optimal fronts in most runs. The *CAT7_105* graph is an artificial graph known as a hard benchmark, the results are shown in Fig. 3d. The WSO2 produces mostly a trivial solution with high balance, the WSO1 only one solution with minimal balance and maximal cut. The Pareto BOA provides the whole Pareto front, SOP produces only individuals from the upper part of this front.

## 6 Parallel Pareto BOA

The establishment of current Pareto front in each generation for bi-criterial optimization takes *O(N* log N)* comparisons. The asymptotic time complexity of the proposed Pareto algorithm does not exceed the complexity of conventional BOA. The execution time of one generation for Pareto algorithm is nearly the same as for SOP or WSO, but the difference is in the number of generations used. In SOP and WSO algorithm there is an implicit detector of population saturation used to stop the evolution. In Pareto BOA the population is implicitly "split" into several niches and each of them converges to different solution which results in slower convergence. Because it is not simple to specify the stopping criterion, we often must specify maximum number of generations. The Pareto BOA wasted in our experiments five-times more generations than SOP and WSO. To decrease the wasting time, we suggest a parallel construction of Bayesian network as described in [9] for single criterion optimization. The next approach for the future work is the decomposition of the Pareto front into segments which can be constructed in separate but cooperating subpopulations.

## 7 Conclusions

We have implemented multiobjective Pareto BOA algorithm for the hypergraph bisectioning. The Pareto BOA performance was compared to single BOA with relaxed balance and weighted sum algorithms WSO. The WSO is very sensitive to type of problem see fig. 3a, 3b. The SOP algorithm provides mostly an upper part of Pareto front towards higher balance values. In the case of real hypergraphs IC67, IC116, the Pareto set is uniformly distributed along the Pareto front only in case of Pareto BOA. The main problem which remains to be solved is the large computation complexity and large population size. The parallelization of the Pareto BOA is necessary. The next possible improvement lies in more sophisticated niching technique, modification of replacement phase of the algorithm and introduction of problem knowledge into optimization process. The future work will be mainly directed towards the parallelization of BOA algorithm on the platform of SUN workstations, which will include the parallelization of Bayesian network construction and the decomposition of the Pareto front detection. Separate but cooperating subpopulations using the migration operator will be used.

*This research has been carried out under the financial support of the Czech Ministry of Education – FRVŠ No. 0171/2001 "The parallelization of Bayesian Optimization Algorithm" and the Research intention No. CEZ: J22/98: 262200012 – "Research in information and control systems".*

## References

1. Jeffrey Horn, Nicholas Nafpliotis, and David E. Goldberg. *A Niched Pareto Genetic Algorithm for Multiobjective Optimization,* In Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, volume 1, pages 82-87, Piscataway, New Jersey, June 1994. IEEE Service Center.
2. Carlos Artemio Coello Coello. *An Empirical Study of Evolutionary Techniques for Multiobjective Optimization in Engineering Design.* PhD thesis, Department of Computer Science, Tulane University, New Orleans, LA, April 1996.
3. David S. Todd. *Multiple Criteria Genetic Algorithms in Engineering Design and Operation. PhD thesis,* University of Newcastle, Newcastle-upon-Tyne, UK, October 1997.
4. Eckart Zitzler. *Evolutionary Algorithms for Multiobjective Optimization*: Methods and Applications PhD thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, November 1999.
5. Joshua D. Knowles, David W. Corne, and Martin J. Oates. *The Pareto-Envelope based Selection Algorithm for Multiobjective Optimization,* In Proceedings of the Sixth International Conference on Parallel Problem Solving from Nature (PPSNVI), pages 839-848, Berlin, September 2000. Springer.
6. Pelikan, M. *A Simple Implementation of Bayesian Optimization Algorithm in C++(Version1.0).* Illigal Report 99011, February 1999, pp. 1-16.
7. Pelikan, M., Goldberg, D. E., & Cantú-Paz, E. *Linkage Problem, Distribution Estimation, and Bayesian Networks.* IlliGal Report No. 98013, November 1998, pp. 1-25.
8. Schwarz, Josef., Očenášek, Jiří. *Experimental Study: Hypergraph Partitioning Based on the Simple and Advanced Genetic Algorithm BMDA and BOA.,* Proceedings of the Mendel'99 Conference, Brno University of Technology, Faculty of Mechanical Engineering, Brno, 1999, pp. 124-130, ISBN 80-214-1131-7.
9. Schwarz, Josef., Očenášek, Jiří. *The Parallel Bayesian Optimization Algorithm.* Proceedings of the European Symposium on Comutational Inteligence, Physica-Verlag, Košice, Slovak Republic, 2000, pp. 61-67, ISBN 3-7908-1322-2, ISSN 1615-3871.

Schwarz Josef, Očenášek Jiří:

**Multiobjective Bayesian Optimization Algorithm for Combinatorial Problems: Theory and Practice.**

A-7

# MULTIOBJECTIVE BAYESIAN OPTIMIZATION ALGORITHM FOR COMBINATORIAL PROBLEMS: THEORY AND PRACTICE

*Josef Schwarz*[*]
*Jiří Očenášek*[†]

**Abstract:** This paper deals with the utilizing of the Bayesian optimization algorithm (BOA) for the multiobjective optimization of combinatorial problems. Three probabilistic models used in the Estimation Distribution Algorithms (EDA), such as UMDA, BMDA and BOA which allow one to search effectively on the promising areas of the combinatorial search space, are discussed. The main attention is focused on the incorporation of Pareto optimality concept into classical structure of the BOA algorithm. We have modified the standard algorithm BOA for one criterion optimization utilizing the known niching techniques to find the Pareto optimal set. The experiments are focused on tree classes of the combinatorial problems: artificial problem with known Pareto set, multiple 0/1 knapsack problem and the bisectioning of hypergraphs as well.

Key words: *Multiobjective optimization, Pareto and non Pareto algorithms, evolutionary algorithms, probabilistic model, estimation distribution algorithms, Bayesian optimization algorithm, niching techniques.*

*Received: July 9, 2001*
*Revised and accepted: August 30, 2001*

## 1. Introduction

Combinatorial optimization problems, such as the placement problem, number partitioning problem (NPP), decomposition problem, traveling salesman problem

[*]Josef Schwarz
Brno University of Technology, Faculty of Engineering and Computer Science, Department of Computer Science and Engineering, CZ - 61266 Brno, Božetěchova 2, Czech Republic, phone.: 420 5 41141210, fax: 41141270, e-mail: schwarz@dcse.fee.vutbr.cz
[†]Jiří Očenášek
Brno University of Technology, Faculty of Engineering and Computer Science, Department of Computer Science and Engineering, CZ - 61266 Brno, Božetěchova 2, Czech Republic, phone: 420 5 41141283, fax: 41141270, e-mail: ocenasek@dcse.fee.vutbr.cz

(TSP), job-shop scheduling, bin packing problem, facility layout problem, knapsack problem, etc., belong to the class of NP hard problems [1]. The search space is often very large and it is not possible to use enumerative techniques because the complexity of the problems is expressed by $O(n!)$ or $O(r^n)$, where $n$ is the size of the problem and $r < n$ is the cardinality of the alphabet used. To avoid the problem with a finite alphabet string it is possible to replace the original string by $kn$ bit string where $k = log(r)$. In case of $r = 2$, the problem is reduced to a binary optimization problem with $2^n$ complexity. Let us note that we will focus on this type of encoding. Generally, the solution can be represented by a vector of parameters with unknown inter-parameter dependencies. However, many combinatorial optimization algorithms have no mechanism for capturing inter-parameter dependencies. But it is only this approach that allows concentrating the sampling more effectively on regions of the search space which have appeared to be promising in the past. Most optimization algorithms do this only by searching around the location of the best previous solution or by using various types of genetic algorithms (GA). The classical genetic algorithms (GA) have a common disadvantage - the necessity of setting the parameters as crossover, mutation and selection rate and the choice of a suitable type of genetic operators. That is why we have analyzed and also used some of the Estimation of Distribution Algorithms (EDAs) called probabilistic model-building genetic algorithms. The crossover and mutation operators used in standard GA are replaced in EDAs by probability estimation and sampling techniques.

In case of probabilistic methods, statistics about the search space is explicitly maintained by creating models of the good solutions found. The efficiency of such techniques depends naturally on the complexity of model used and on the complexity of problems.

Next we will discuss population based evolutionary algorithms using probabilistic models with various complexity. Let us denote:

$D = (X^1, X^2, \dots, X^N)$ with $X^j \in D$, is the population of the solutions/string/individuals

$X = (X_0, X_1, \dots, X_{n-1})$ is a string/individual of length $n$ with $X_i$ as a variable

$x = (x_0, x_1, \dots, x_{n-1})$ is a string/individual with $x_i$ as a possible instantiation of variable $X_i, x_i \in \{0, 1\}$

$P = (p(x_0), p(x_1), \dots, p(x_{n-1}))$, with $p(x_i) \in [0, 1]$ is the vector of univariate marginal probabilities

$p(x_0, x_1, \dots x_{n-1}) = p(X_0 = x_0, X_1 = x_1, \dots, X_{n-1} = x_{n-1})$ denotes the $n$ dimensional distribution

## 2. Probabilistic models

The performance of EDA algorithms that work on the basis of probabilistic models can be specified in the following common framework:

```
Generate initial population of individuals of size $M$ (randomly);
While termination criteria is false do
begin
    Select the parent population of $N$ individuals according to a selection method;
    Estimate the probability distribution of the selected parents;
    Generate new offspring according to the estimated probabilistic model;
    Replace some individuals in current population with generated offspring;
end
```

Next, we will describe three main types of the probabilistic models used in EDA algorithms according to their complexity – models without inter-dependency, with pairwise dependencies and multivariate dependencies.

## 2.1 Models without dependencies

The probability vector $P$ of univariate probabilities is used to model simple probability distribution. In Univariate Marginal Distribution Algorithm UMDA [2] it is assumed that variables are mutually independent, see Fig. 1a. For each variable position $i \in \{0 \ldots n-1\}$ and each possible value of this variable $x_i \in \{0, 1\}$, the univariate marginal frequency $p_i(x_i)$ is defined as the frequency of strings that have $x_i$ on the $i$-th position in the parent population $D$ :

$p_i(x_i) = n_i(x_i)/N$,

where $n_i(x_i)$ is a number of appearances of the allele $x_i$ on the $i$-th position. Each new individual $X = (x_0, x_1, \ldots, x_{n-1})$ is generated by UMDA according to the distribution:

$$p(X) = \prod_{i=0}^{n-1} p_i(x_i), \tag{1}$$

so the value of the $i$-th variable is set to value $a$ with probability equal to $p_i(a)$. UMDA is able to cover efficiently only linear problems.

## 2.2 Models with pairwise dependencies

The univariate probability is used as in UMDA. In addition, pair dependencies are allowed [3]. The bivariate marginal frequency $p_{i,j}(x_i, x_j)$ used in Bivariate Marginal Distribution Algorithm BMDA is defined as frequency of individuals in parent population $D$ that have values $x_i$ and $x_j$ on positions $i$ and $j$ at the same time: $p_{i,j}(x_i, x_j) = n_{i,j}(x_i, x_j)/N$. Conditional probability of occurrence of the value $x_i$ on the $i$-th position in the case of occurrence of $x_j$ on the $j$-th position is determined

$$p_{i,j}(x_i|x_j) = p_{i,j}(x_i, x_j)/p_j(x_j). \tag{2}$$

We extended the concept of UMDA and BMDA for alphabet encoding, so that $x_i \in \{0, \ldots, r_i - 1\}$ and $x_j \in \{0, \ldots, r_j - 1\}$ [4]. For each pair of positions $i, j$ the count of each combination of values can be summarized into a contingency table. Variable dependencies are discovered by Pearson's chi-square statistics we used in [4] the following form of equation:

$$X_{i,j}^2 = N\left(\sum_{k=0}^{r_i-1}\sum_{l=0}^{r_j-1}\frac{n_{i,j}^2(k,l)}{n_i(k)\,n_j(l)} - 1\right). \tag{3}$$

Variables are considered to be independent if the result does not meet a certain threshold. For example, binary variables are independent for 95% if $X_{i,j}^2 < 3.84$. The dependency information is used to build up the acyclic dependency graph-probabilistic graphical model, which can be taken as a set of trees. Each new individual $X = (x_0, x_1, \ldots, x_{n-1})$ is generated according to the distribution

$$p(X) = \prod_{i=0}^{n-1} p_i(x_i | x_{m(i)}), \tag{4}$$

where $m(i)$ is any number between zero and $n-1$ or nothing (in the case of root nodes of the tree). The root nodes correspond to the positions where the values are generated using the univariate marginal distribution; the values of the positions connected to already generated positions in the graph are subsequently generated using the conditional probability. An example of the tree dependency graph is shown in Fig. 1b.

## 2.3 Models with multivariate dependencies

The complex model using the Bayesian network to encode the structure of a problem was implemented in the Bayesian Optimization Algorithm BOA [5], [6], [7], [8]. It is an analogy of BMDA dependency graph, but the higher order variable dependencies can be covered too. For each variable $X_i$ a set of parent variables $\Pi_{X_i}$ is defined which it depends on, so the distribution of individuals is expressed by the conditional probabilities:

$$p(X) = \prod_{i=0}^{n-1} p\left(X_i | \Pi_{X_i}\right). \tag{5}$$

Generally, the existence of the directed edge from $X_j$ to $X_i$ in the network implies the belonging of variable $X_j$ to set $\Pi_{X_i}$. To reduce the space of networks, the number of incoming edges into each node is limited to $k$. The key step is thus the estimation of probability $p(X)$ via finding the Bayesian network with the maximum score measure. As is known, the probability of Bayesian network $B$ given data $D$ is done by Bayes theorems [9],[10]:

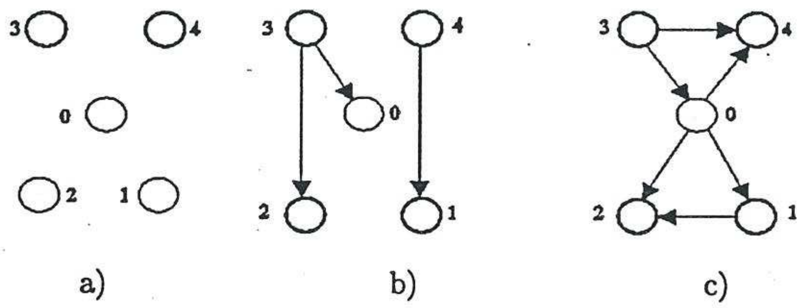
$$p(B|D) = p(D|B)p(B)/p(D) \tag{6}$$

where

| | |
|---|---|
| $p(B/D)$ | is posterior probability of $B$ given $D$ |
| $p(B)$ | is the prior probability of the Bayesian network specified by an expert |
| $p(D/B)$ | is the probability of data $D$ given $B$ |

The Bayesian Dirichlet (BD) metric is used to calculate $p(D|B)$ as a measure of the network quality [12], [6]. To construct a good network quickly, the greedy algorithm is used in such a way that in each step the best edge is added according to the BD metric.

After the network construction new instances are generated using the univariate and conditional probability. An example of the Bayesian network is in Fig. 1c.



*a)* $p(X) = p(X_3)p(X_0)p(X_2)p(X_4)p(X_1)$,
*b)* $p(X) = p(X_3)p(X_0/X_3)p(X_2/X_3)p(X_4)p(X_1/X_4)$
*c)* $p(X) = p(X_3)p(X_0/X_3)p(X_4/X_3, X_0)p(X_1/X_0)p(X_2/X_1, X_0)$

**Fig. 1** *Graphical models and joint probability distribution for a) UMDA, b) BMDA, c) BOA*

## 3. Presentation of Bayesian statistics used in the BOA

The Bayesian network models the n – dimensional probability by product of conditional probabilities. The conditional probability can be stated for the current population using the Bayesian statistics. Let us consider the fragment of the Bayesian network where node/variable $X_7$ depends on the variables $X_2, X_5$. From the current population shown in Tab. I the number of particular combinations of $X_2, X_5$ for each value of $X_7$ can be found. For example there exists one occurrence of the combination $X_2X_5 = 11$ for $X_7 = 1$ and the equal one for $X_7 = 0$. Consequently we get the conditional probability for $X_7 = 1$ :

$p(X_7|X_2 = 1, X_5 = 1) = 0.50$

Similarly, the probability for other combinations of entry variables $X_2, X_5$ is stated:

$p(X_7|X_2 = 1, X_5 = 0) = 1.00$

$p(X_7|X_2 = 0, X_5 = 0) = 0.25$

$p(X_7|X_2 = 0, X_5 = 1) = 0.00$ ... no occurrence of $(X_2, X_5, X_7) = (0, 1, 1)$

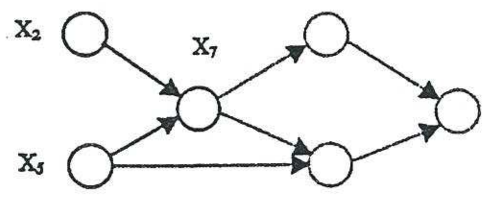


**Fig. 2** *Example of a fragment of the Bayesian network.*

These particular conditional probabilities are used for generating new values for binary variable $X_7$ having $X_2$ and $X_5$ in the frame of offspring generation using sampling procedure.

| | | $X_2$ | | $X_5$ | | $X_7$ | |
|---|---|---|---|---|---|---|---|
| S1 | ... | 1 | ... | 0 | ... | 1 | ... |
| S2 | ... | 0 | ... | 0 | ... | 0 | ... |
| S3 | ... | 1 | ... | 0 | ... | 1 | ... |
| S4 | ... | 1 | ... | 1 | ... | 1 | ... |
| S5 | ... | 0 | ... | 0 | ... | 0 | ... |
| S6 | ... | 0 | ... | 1 | ... | 0 | ... |
| S7 | ... | 0 | ... | 0 | ... | 1 | ... |
| S8 | ... | 1 | ... | 1 | ... | 0 | ... |
| S9 | ... | 0 | ... | 0 | ... | 0 | ... |
| S10 | ... | 0 | ... | 1 | ... | 0 | ... |

$(X_2X_5)$ $X_7$

| $\pi_{X_i}$ | $x_i$ | $m(x_i,\pi_{X_i})$ | $m(\pi_{X_i})$ |
|---|---|---|---|
| 00 | 0 | 3 | } 4 |
| | 1 | 1 | |
| 01 | 0 | 2 | } 2 |
| | 1 | 0 (none) | |
| 10 | 0 | 0 (none) | } 2 |
| | 1 | 2 | |
| 11 | 0 | 1 | } 2 |
| | 1 | 1 | |

**Tab. I** *Presentation of the Bayesian statistics on the population with ten strings for a fragment of the elementary Bayesian network shown in Fig. 2.*

Knowing this particular conditional probability for $X_7$ the n-dimensional probability can be expressed as

$$p(X) = p(X_0|\ldots)^* p(X_1|\ldots)\ldots^* p(X_7|X_2,X_5)\ldots. \tag{7}$$

The quality of the Bayesian network is expressed by the BD metrics:

$$\begin{aligned} p(D,B|\xi) &= p(B|\xi)\prod_{i=0}^{n-1}\prod_{\pi_{X_i}}\frac{m'(\pi_{X_i})\,!}{(m'(\pi_{X_i})+m(\pi_{X_i}))\,!} \\ &\prod_{x_i}\frac{(m'(x_i,\pi_{X_i})+m(x_i,\pi_{X_i}))\,!}{m'(x_i,\pi_{X_i})\,!} \end{aligned} \tag{8}$$

where the meaning of items $\pi_{X_i}$, $m(x_i,\pi_{X_i})$, $m(\pi_{X_i})$ flows from Tab I. The prime version of these terms express prior knowledge of network topology.

## 3.1 Complexity and prior information in the original BOA

The computational complexity of the network construction and scoring the metric calculation is done by term $O(k.n^3) + O(k.2^k.n^2.N)$ [11]. The complexity of the new population generation is negligible. We have proposed two approaches to decrease the time complexity. In [12] we solved the problem of graph bisectioning. Our goal was to incorporate the problem knowledge into the whole process of the Bayesian network construction. First we applied the prior probability $p(B)$ of the Bayesian network expressed by term $p(B|\xi) = c\kappa^{\delta}$. The essence of this approach lies in the penalization of the edges of the Bayesian network having no match in the graph to be decomposed. Variable $\delta$ is the number of edges in the final Bayesian network having no match in the hypergraph to be bisected; $c$ and $k$ are normalization constants. Second, we used the concept of cluster injection into the initial population detected on the hypergraph structure that seems to be really a promising tool for enhancement of the population genotype. Both of these phenomena used lead to the meaningful reduction of population size and to better convergence.

## 4. Multiobjective optimization

Practical problems are often characterized by several, often competing, objectives. While in the case of single-objective optimization the optimal solution is simply distinguishable, this is not true for multiobjective optimization. The standard approach to solve this difficulty lies in finding all possible trade-offs among the multiple competing objectives. These solutions are optimal, nondominated, in that there are no other solutions superior in all objectives. These so called Pareto optimal solutions lie on the Pareto optimal front. A general multiobjective optimization/maximization problem (MOP) can be described as a vector function $f$ that maps a tuple of $n$ parameters to a tuple of $m$ objectives [13]:

$$\max y = f(x) = (f_1(x), f_2(x), \ldots, f_m(x)) \tag{9}$$

$$\begin{aligned} &\text{subject to} \quad h(x) = (h_1(x), h_2(x), \ldots, h_k(x)) <= 0 \\ &\text{subject to} \quad x = (x_1, x_2, \ldots, x_n) \in X \\ &\qquad\qquad\quad\; y = (y_1, y_2, \ldots y_m) \in Y, \end{aligned}$$

where $x$ is called a decision vector, $X$ is the parameter space, $y$ is the objective vector, $Y$ is the objective space and the constraint vector $h(x) <= 0$ determines the set of feasible solutions/set $X_f$.

The set of solutions of MOP includes all decision vectors for which the corresponding objective vectors cannot be improved in any dimension without degradation in another one – these vectors are called the Pareto optimal set. The idea of Pareto optimality is based on the Pareto dominance.

For any two decision vectors $u$, $v$ it holds

| | | |
|---|---|---|
| $u \succ v$ | (*u dominates v*) | *iff* $f(u) > f(v)$, |
| $u \succ= v$ | (*u weakly dominates v*) | *iff* $f(u) >= f(v)$, |
| $u \sim v$ | (*u is indifferent to v*) | *iff u,v are not comparable.* |

Decision vector $u$ dominates decision vector $v(u \succ v)$ iff $f_i(u) \geq f_i(v)$ for $i = 1, 2, \ldots, m$ with $f_i(u) > f_i(v)$ for at least one $i$. Vector $u$ is called Pareto optimal if there is no vector $\boldsymbol{v}$ which dominates vector $\boldsymbol{u}$ in parameter space $X$. In the objective space the set of nondominated solutions lies on a surface known as Pareto optimal front. The goal of the optimization is to find a representative sampling of solutions along the Pareto optimal front. An example of the concept of the Pareto dominance and the Pareto-optimal front are presented in a graphical form in Fig. 3.

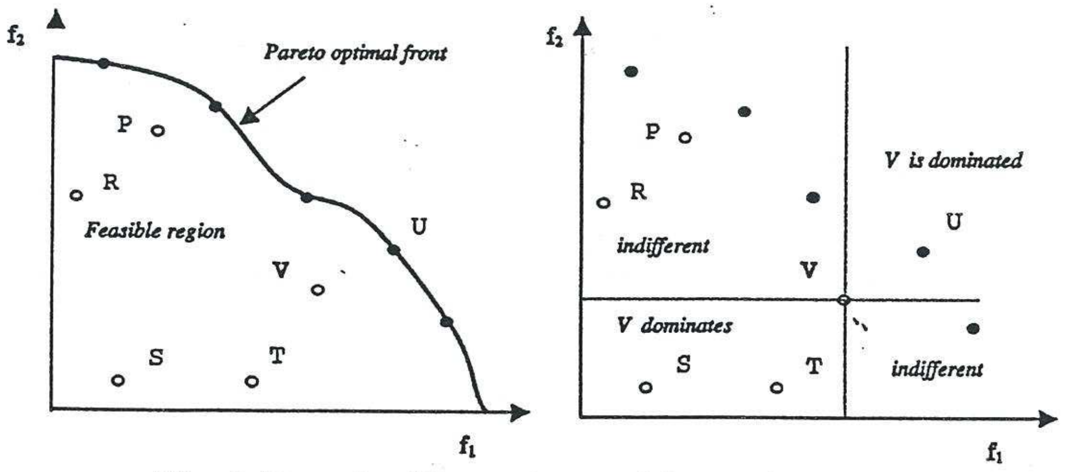


**Fig. 3** *Example of Pareto front and Pareto dominance.*

It can be stated that solutions $V$ dominates solutions $S$ and $T$, solution $V$ is dominated by $U$, solution $U$ is nondominated and Pareto optimal, solution $S$ is dominated by $V$ and $T$.

## 5. Optimization methods

### 5.1 Pareto methods

There are many papers that present various approaches to find the Pareto optimal front almost based on the classical evolutionary algorithms. All of them must solve the problem of adequate solutions evaluation and population diversity. Pareto-based fitness assignment was developed using the concept of dominance in order to determine the reproduction probability of each solution. It is evident that unlike the case of single optimization, fitness is related to the whole population.

The multiobjective optimization is a typical multimodal search for finding multiple different solutions in a single run. To reach this goal various niching techniques are used. Well known is fitness sharing – the more individuals are located in the neighborhood (defined by niche radius) of a solution the more is the fitness decreased. Less frequently nonniching techniques are used, e.g. such as restricted mating (only similar parents are mated) and crowding (offspring replace similar parents). We will shortly mention the main representatives of the Pareto optimization algorithms: The Niched Pareto Genetic Algorithm (NPGA) combines tournament selection and the concept of Pareto dominance [14]. A wide review of the basic approaches and the specification of original Pareto evolutionary algorithms include dissertations [15], [13] where the last one describes the original Strength Pareto Evolutionary Algorithm (SPEA). An interesting approach using, nondominated sorting in genetic algorithm (NSGA), is published in [16]. An interesting extension of the SPEA algorithm resulting in PESA algorithm is described in [17].

Pareto optimal methods have more preferences than disadvantages. Belonging to the advantages is the fact that Pareto approaches take all objectives into consideration simultaneously - every point/solution of the Pareto front is a good solution – and maintains the diversity of solutions. Naturally we can list two main disadvantages – this approach is computationally expensive and not very intuitive if the number of objectives is large. All these algorithms mentioned above progress towards the Pareto optimal set with a good distribution of solutions but none of them guarantees convergence to a true Pareto optimal set. The promising approach of archive-based Pareto optimization algorithms is published in [18] where the concept of $\varepsilon$-approximate Pareto set and $\varepsilon$-Pareto set is introduced. A class of algorithms is suggested with guaranteeing convergence to a diverse set of $\varepsilon$-Pareto optimal solutions.

## 5.2 Non-Pareto methods

There are many methods for the multi-criteria optimization, mostly based on the scalarization of the objective function or other non-Pareto approaches. In this way the MOP problem can be easily transformed into a more simpler SOP problem.

### Weighted - sum approach

The well known aggregation method is based on the weighted sum approach. This method transforms the objective function vector into a higher scalar function using the weighted sum of particular objectives. Let us note that in the single objective optimization, the feasible set is completely ordered according to the single objective function. For two solutions $\boldsymbol{u}, \boldsymbol{v} \varepsilon \boldsymbol{X}_f$ *either* $f(u) >= f(v)$ or $f(v) >= f(u)$. In case of multiobjective optimization, the feasible set is only partially ordered. In case of maximization we get:

$$maximize\, y = f(x) = w_1 f_1(x) + w_2 f_2(x) + \cdots + w_m f_m(x). \tag{10}$$

This equation can be modified to the following form which can be understood as a sector equation of line with slope $-w_1/w_2$ *and intercept* $y/w_2$ [13] (see Fig. 4.) :

$$f_2(x) = (-w_1/w_2) f_1(x) + y/w_2 \tag{11}$$

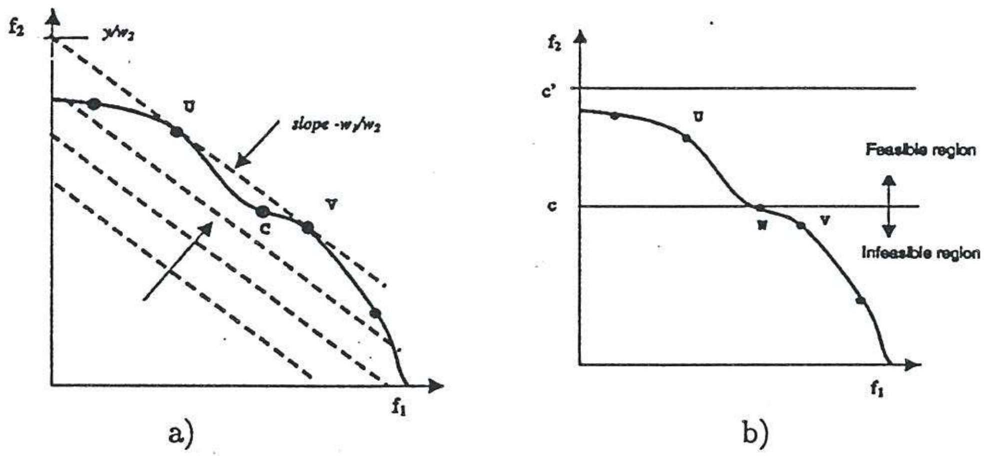


**Fig.** 4 *Example of an exploration of the search space by a) weighted sum approach, b) constrained methods.*

The set of parallel lines represents the successive search in the objective space. This approach is able to find Pareto optimal solutions but the search process is very sensitive to weight coefficients and this technique is not able to reach solutions in a local non-convex region, see point $C$.

## Constraint method

This technique is quite simple: only one function is optimized, the others are transformed into constraints. Then the penalty function can be used to fulfil the constraints. For the case of bi-objective optimization we get

$$y = f_1(x) \tag{12}$$

$$h(x) = f_2(x) >= c,$$

where $c$ is chosen repeatedly (see Fig 4.b) but remark that the $c'$ is not an available choice of bound. Another principle is used in the priority (lexicographic) method - the objective with the highest priority is minimized first, then the next with lower priority, etc. The problem appears how to state the importance of objectives.

## Fuzzy-control approach

Fuzzy controllers are often used in the control system and generally in soft computing. In [19] there is an interesting approach for scalarization of a bi-objective problem. Each generation the centre of the current population is detected and according to this knowledge the fuzzy controller decides what transformation of the cost components into a one-dimensional fitness function is taken.

Let us note that the attraction of the SOP methods described above is supported by many useful and well-studied heuristic methods like dynamic programming, branch and bound method, random search algorithm, stochastic local search algorithms and simulated annealing. In common they require several runs to obtain approximation of the Pareto-optimal set.

## 6. Pareto optimal BOA

In our Pareto BOA algorithm we replaced the original fitness assignment and replacement step of standard BOA by the Pareto niching technique utilizing a new strength criterion for the evaluation process [13]. The following specification describes the whole reproduction process of our algorithm. Let us note that although we have solved bi-objective optimization; our algorithm is able to solve $m$-objective optimization problems. The flowchart of our Pareto BOA algorithm is in Fig. 5.

It follows from the flowchart that the algorithm is archive-oriented. In an external archive the nondominated solutions during all the optimization history are archived and enter regularly into the selection and replacement phase. The most important part of our Pareto algorithm [20], [21] is the procedure for detection of nondominated (current Pareto front) and dominated solutions and sophisticated fitness calculation. The procedure for current nondominated and dominated set detection is described in the following steps:

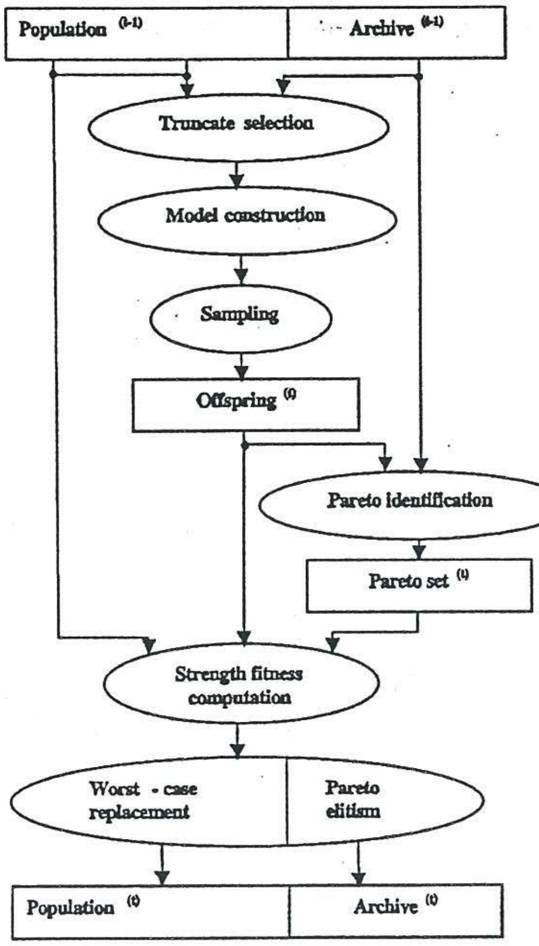


Fig. 5 *The flowchart of the Pareto BOA algorithm*

1. For each individual $X$ in population $D$ compute the vector of the objective functions

$$\overline{f}(X) = (f_1(X), f_2(X), \ldots, f_m(X)) \tag{13}$$

2. Detect the subset of nondominated solutions

$$\overline{D} = \left\{ X_j | X_j \in D \wedge \overline{\exists} X_i \in D : X_i \succ X_j \right\} \tag{14}$$

3. For each nondominated solution $X_j$ from $D$ compute its strength value as

$$s(X_j) = \frac{|\{X_i | X_i \in D \wedge \; X_j \succ X_i\}|}{|D| + 1} \tag{15}$$

4. The fitness for nondominated solutions is equal to the reverse of the strength value

$$f'(X_j) = 1/s(X_j) \tag{16}$$

5. For each dominated solution $X_i$ from $D$ determine the fitness as

$$f'(X_i) = 1/\left(1 + \sum_{X_j} s(X_j)\right), \tag{17}$$

where $X_j \in \overline{D} \wedge X_j \succ X_i$.

# 7. Problem specification

## 7.1 Bi-objective Onemax/Xor binary problem

We start to test our algorithm by an artificial problem with an easily determined Pareto optimal front. We call this problem a Onemax/Xor problem (unitation versus pairs in [16]). It is defined on the binary string. Onemax function $f_2$ is simply stated by the number of ones in the $n$ bit string $X$, e.g. $f_2(X) = f_2(0110) = 4$. $X$ or function denoted as $f_1$ is specified by the number of pairs of adjacent complementary bits, either 10 or 01, thus $f_1(X) = f_1(0110) = 1 + 0 + 1 = 2$. The goal is to maximize both functions.

## 7.2 Multiple 0/1 knapsack problem

Generally, the 0/1 knapsack problem consists of a set of items, weight and profit associated with each item, and an upper bound of the capacity of the knapsack. The task is to find a subset of items which maximizes the sum of the profits in the subset, yet all selected items fit into the knapsack so as the total weight does not exceed the given capacity. This single objective problem can be extended to a multiobjective multiple problem by allowing more than one knapsack. Formally, the multiobjective 0/1 knapsack problem is defined in the following way: Given a set of $n$ items and a set of $m$ knapsacks, with the following parameters:

| | |
|---|---|
| $p_{i,j}$ | profit of item $j$ according to knapsack $i$ |
| $w_{i,j}$ | weight of item $j$ according to knapsack $i$ |
| $c_i$ | capacity of knapsack $i$, |

find vector $x = (x_1, x_2, \ldots; x_n) \in \{0,1\}^n$, such that $x_j = 1$ iff item $j$ is selected and

$$f(x) = (f_1(x), f_2(x), \ldots, f_m(x)) \text{ is the maximum, where} \tag{18}$$

$$f_i(x) = \sum_{j=1}^{n} p_{i,j} * x_j \tag{19}$$

and for which the constraint is fulfilled

$$\forall i \in \{1, 2 \ldots, m\} : \sum_{j=1}^{n} w_{i,j} * x_j \leq c_i. \tag{20}$$

The complexity of the problem solved depends on the values of the knapsack capacity. According to [13] we used the knapsack capacities stated by the equation:

$$c_i = 0.5 \sum_{j=1}^{n} w_{i,j}. \tag{21}$$

The encoding of the solution into chromosome is realized by a binary string of length $n$. To satisfy the constraints (21) it is necessary to use repair mechanism on the generated offspring to be feasible one.

## 7.3 Hypergraph bisectioning

The bisectioning problem can be defined as follows: Let us assume a hypergraph $H = (V, E)$, with $n = |V|$ nodes and $e = |E|$ edges. We look for such a bisection $(V1, V2)$ of $V$ that minimizes the number of hyperedges that have nodes in a different set $V1, V2$ and balance $b$ of the partition sizes. The set of external hyperedges can be labelled as $E_{cut}(V1, V2)$. The cost function is the number of external hyperedges, shortly called cut size.

Each solution of the bisection is encoded as a binary string $X = (x_0, x_2, \ldots, x_{n-1})$. The variable $x_i$ represents the partition number, the index specifies the node in the hypergraph. For the case of simple graph $G(V, E, R)$ bisectioning we have derived on the binary string $X = (x_0, x_1, \ldots, x_{n-1})$ the following two functions to be minimized [18]:

$$f_1 = E_{cut}(V1, V2) = \sum_{\substack{i=0 \\ j>i}}^{n-1} r_{ij}(x_i + x_j - 2x_i x_j) \tag{22}$$

$$f_2 = b = |\sum_{i=0}^{n-1} x_i - \sum_{i=0}^{n-1}(1 - x_i)| \tag{23}$$

where the coefficient $r_{ij} = 1$ in case the net/edge of graph G exists between node $i$ and $j$, else $r_{ij} = 0$. Function $f_1$ represents the cut value of the bisection and function $f_2$ expresses the balance/difference of the partition sizes. We have tested two approaches for solving this 2-objective optimization problems: the weighted sum approach and the Pareto optimal method. The first approach transforms the original vector-valued objective function into a scalar-valued objective function. The objective function of solution $X$ is computed as a weighted sum of all objective functions:

$$f(X) = w_1 f_1(X) + w_2 f_2(X), \tag{24}$$

where $w_1, w_2$ are weight coefficients. The sensitivity of the optimization process to these values is well known. We have tested two sets of these coefficients. In the WSO1 variant we have chosen $w_1 = 0.5, w_2 = 0.5$, in the WSO2 couple of $w_1 = 0.005, w_2 = 0.995$ was used. This values were found experimentally.

## 8. Experimental results

### 8.1 Test benchmarks

We used three types of benchmarks:

1. The 64 bit Onemax/Xor function with the known Pareto set including 32 indifferent solutions.

2. Two knapsack benchmarks specified by 100 (Kn100) and 250 items (Kn250) published on the web site
[http://www.tik.ee.ethz.ch/~zitzler/testdata.html#fileformat].
We have compared our results with the results obtained by two evolutionary algorithms SPEA [13] and NSGA [16]. These two algorithms represent the well working evolutionary multiobjective algorithms.

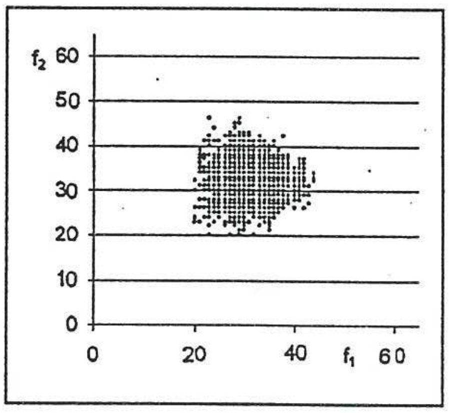
3. Two hypergraphs IC67, IC116 representing real circuits. The global optimum is not known. The structure of the circuits can be characterized as a random logic. Hypergraph IC67 consists of 67 nodes and 134 edges/nets, the IC116 consists of 116 nodes and 329 edges/nets.

## 8.2 Experiments and results

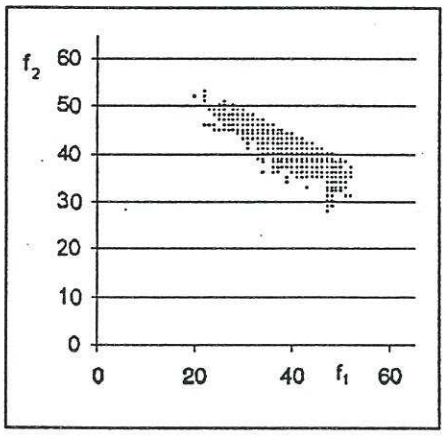
All experiments were performed on the Sun Enterprise 450 machine (4 CPUs, 4 GB RAM), in the future we consider utilizing the cluster of Sun Ultra 5 workstations.

**The Onemax/Xor binary problem**

In Fig. 6 the population distributions in the 1st generation and 10th generation are shown.



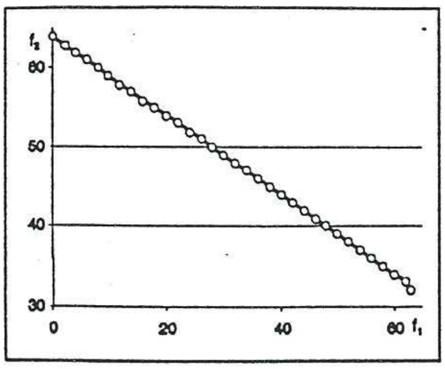
a)

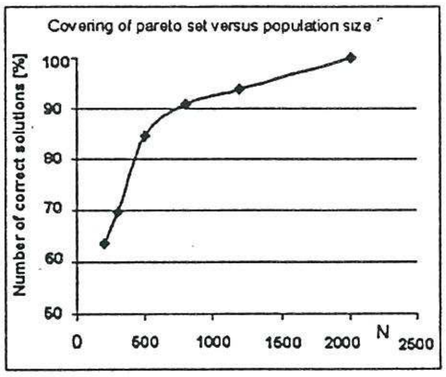


b)

**Fig. 6** *Population distributions for Onemax/Xor benchmark plotted in the a) 1st generation, b) 10th generation.*

In Fig. 7a we plot the final Pareto front (N=2000) after 100 generations. Our algorithm was succesful completely in finding the known Pareto set; in the NPGA algorithm [14] the Pareto set was not found entirely. In Fig. 7b the relation between the population size and covering the known Pareto set are shown. It is evident that for the 64 bit-string problem the population size N=2000 is sufficient. The computation time is about 3 minutes. Let us note that for a smaller population size up to N=300 we get solutions from the Pareto front but not completely all.





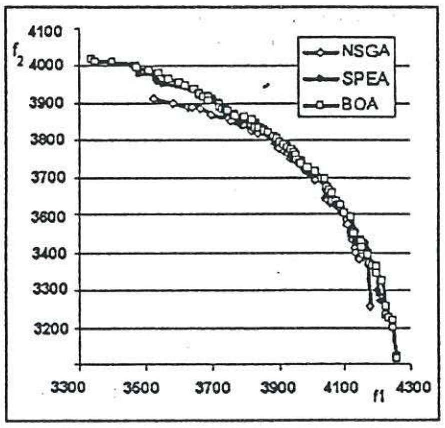
a) b)

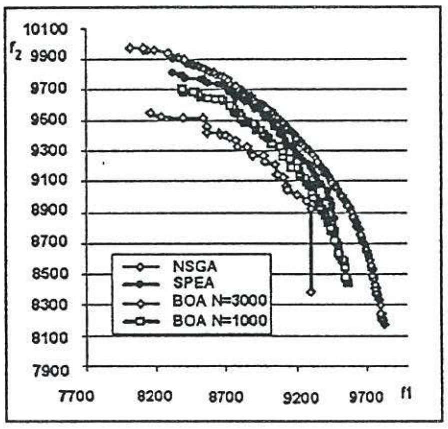
**Fig. 7a** *Pareto optimal front for the Onemax/Xor problem,*
**Fig. 7b** *Covering of the known Pareto set versus population size.*

## Knapsack problem

In Fig. 8a there is a comparison of the final Pareto front produced by our Pareto BOA algorithm and by the SPEA [13] and NSGA [16] algorithms for the case of Kn100 benchmark and in Fig. 8b for the case of Kn250 benchmark. We performed 5 independent runs and constructed the final Pareto front from 5 particular Pareto fronts. We used the following setting for our algorithm: for Kn100 we set the population size $N$ to 2000, for the case of Kn250 the population size $N$ equals to 3000 and alternatively to 1000. The number of generations used is 150. The computation time is presented in Tab II. In the context of the algorithm comparison an important question arises: What measure should be used to express the quality of the results so that the various evolutionary algorithms can be compared in a meaningful way. We preferred the topology/shape of the Pareto fronts in our comparison.

From a it is evident that for Kn100 the Pareto solutions produced by our Pareto BOA in the middle part of the Pareto front are slightly better than the Pareto solutions produced by SPEA and NSGA. What is more important – our Pareto BOA produces more solutions in the Pareto front margins.





a) b)

**Fig. 8** *Comparison of final Pareto fronts for a) Kn100, N=2000, b) Kn250, N=3000 and N=1000.*

In Fig. 8b we see that for Kn250 the difference between the Pareto fronts is more expressive – our Pareto BOA for N=3000 outperforms the SPEA and NSGA. In case of limited size N=1000 the Pareto BOA is slightly worse than SPEA, but the Pareto front is longer.

| Problem size n | Population size N, Number of generations | Computational time |
|---|---|---|
| Kn100 | N=1000, 150 gen. | 2 min 40 s |
| Kn100 | N=2000, 150 gen. | 5 min |
| Kn250 | N=1000, 150 gen. | 25 min |
| Kn250 | N=3000, 150 gen. | 45 min |

**Tab. II** *Computational time for knapsack problems*

**Hypergraph bisectioning**

The five independent runs of each algorithm were performed and the final Pareto front from the final populations is shown. For better visualization of the fronts from each run the points are connected by lines.

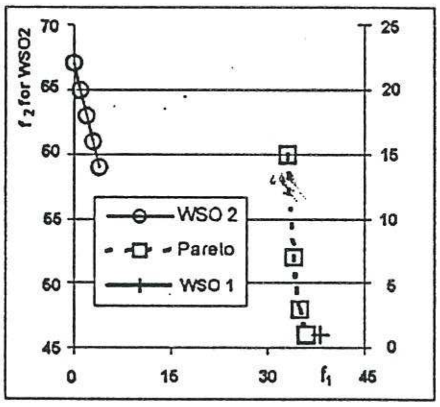


**Fig. 9a** *Bisection of IC67, N=2500.*

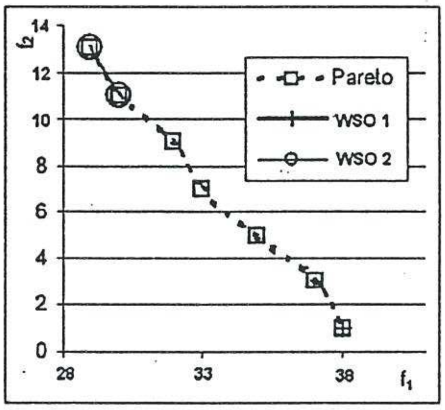


**Fig. 9b** *Bisection of IC116, N=4000.*

From Fig. 9 it is evident that the Pareto algorithm usually produces a better Pareto set with good quality solutions, whereas WSO2 produces only solutions with a low cut size but a high balance and WSO1 produces solutions with a low balance and a higher cut value – only the margins of the Pareto front are covered.

## 9. Parallel Pareto BOA

In [11], [22] we proposed a Distributed Bayesian Optimization Algorithm. It uses a cluster of workstations as a computing platform to speed up the evolution process. Let's note that in the distributed environment the whole population $D$ is split into several parts, each part $D_k$ being generated and evaluated by a different processor. This approach can be extended to the Pareto BOA. We propose the following modification of the procedure for Pareto detection and fitness assignment:

First, each processor will compute the vector of objective functions for all individuals from part $D_k$ of population $D$. Then, each processor detects its local set of nondominated solutions $\bar{D}_k$ as

$$\overline{D}_k = \{X_j | X_j \in D_k \wedge \overline{\exists} X_i \in D_k : X_i \succ X_j\} \tag{25}$$

and the master processor creates the global nondominated set $\bar{D}$ from the union of local nondominated sets $\bar{D}_k$ as

$$\overline{D} = \left\{ X_j | X_j \in \bigcup_k \overline{D}_k \wedge \overline{\exists} X_i \in \bigcup_k \overline{D}_k : X_i \succ X_j \right\}. \tag{26}$$

The strength values for nondominated solutions from $\bar{D}$ can be obtained as a sum of local strength values computed in parallel by all processors:

$$s(X_j) = \frac{\sum_k |\{X_i | X_i \in D_k \wedge \ X_j \succ X_i\}|}{|D| + 1} \tag{27}$$

After that all nondominated solutions and their strength values are known, so each processor is able to compute the Pareto fitness for all individuals from its part of population according to equations (16) and (17).

## 10. Conclusions

We have implemented the multi-objective Pareto BOA algorithm as a modification of the original single-objective BOA algorithm [6], [7], [8] using the concept of a strength criterion applied in the SPEA algorithm [13] for the Pareto oriented fitness. Let us note that the SPEA is a modern multiobjective optimization algorithm which outperforms a wide range of classical methods on many problems.

We have tested the performance of our algorithm on three types of benchmarks. In the case of artificial Onemax/Xor benchmark, we got the known Pareto optimal set completely. In the case of multiple 0/1 knapsack problems Kn100 and Kn250, we got a better result than in [13], [16]. The Pareto solutions produced by our algorithm are uniformly distributed along the Pareto front which is more global than the Pareto fronts obtained by the NSGA and SPEA algorithms.

We also implemented the weighted sum method for the case of hypergraph bisectioning. It is evident that the Pareto BOA algorithm produces a Pareto set with greater cardinality and better solution distribution than both variants of the weighted sum methods. Both algorithms WSO1 and WSO2 were very sensitive to weight coefficients - in accordance with the theory.

But many problems remain to be solved, namely the relatively larger computational complexity. The next possible improvement lies also in a more sophisticated niching technique, modification of replacement phase of the algorithm.

To reduce the computational complexity we proposed the idea of parallelization of the Pareto BOA including the decomposition and detection of the Pareto front. This approach is an extension of the Distributed Bayesian Optimization Algorithm [22], [11] based on the parallelization of the Bayesian network construction. From this point of view the future work will be oriented on the implementation of the Parallel Pareto BOA algorithm for multiobjective optimization problems.

Acknowledgement:

This research has been carried out under the financial support of the Czech Ministry of Education – FRVŠ No. 0171/2001 "The parallelization of Bayesian Optimization Algorithm" and the Research intention No. CEZ: J22/98: 262200012 – "Research in information and control systems".

## References

[1] Ehrgott M., Gandibleux X.: An Annotated Bibliography of Multiobjective Combinatorial Optimization, Report in Wirtschaftsmathematik No. 62/2000, pp. 1-61.

[2] Pelikán M., Muehlenbein H.: Marginal Distributions in Evolutionary Algorithms. Proceedings of the Mendel'98. Brno University of Technology, 1998, pp. 124-130, ISBN 80-214-1199-6.

[3] Pelikán M., Muehlenbein H.: The bivariate Marginal Distribution Algorithms. Advances of Soft Computing-Engineering Design and Manufacturing. London: Springer Verlag, pp. 521-535.

[4] Schwarz J., Očenášek, J.: Experimental Study: Hypergraph Partitioning Based on the Simple and Advanced Genetic Algorithm BMDA and BOA, Proceedings of the Mendel'99 Conference, Brno University of Technology, 1999, pp. 124-130, ISBN 80-214-1131-7.

[5] Baluja S, Davies S.: Using Optimal Dependency-Trees for Combinatorial Optimization: Learning The Structure of the Search Space. Proc. 1997 International Conference on Machine Learning, pp. 30-38.

[6] Pelikán M.: A Simple Implementation of Bayesian Optimization Algorithm in C++(Version 1.0). Illigal Report 99011, February 1999, pp. 1-16.

[7] Pelikán M., Goldberg D. E., & Cantú-Paz E.: Linkage Problem, Distribution Estimation, and Bayesian Networks. IlliGal Report No. 98013, November 1998, pp. 1-25.

[8] Pelikán M., Goldberg D. E., & Lobo F.: A Survey of Optimization by Building and Using Probabilistic Model, Illigal Report 99018, September 1999, pp. 1-12.

[9] Gottvald A.: Bayesian Evolutionary Optimization. Proceedings of the Mendel'99 Conference, Brno University of Technology, Faculty of Mechanical Engineering, Brno, 1999, pp. 30-35, ISBN 80-214-1131-7.

[10] Etxeberria R., Larranaga, P.: Global Optimization Using Bayesian Networks. II. Symposium on Artificial Intelligence. CIMAF99. Special Session on Distributions and Evolutionary Optimization, 1999, 7 pages.

[11] Očenášek J., Schwarz J.: The Parallel Bayesian Optimization Algorithm, Proceedings of the European Symposium on Computational Inteligence, Physica-Verlag, Košice, Slovak Republic, 2000, pp. 61-67, ISBN 3-7908-1322-2, ISSN 1615-3871.

[12] Schwarz J., Očenášek J.: The knowledge-based evolutionary algorithm KBOA for hypergraph bisectioning. Proceedings of the Fourth Joint Conference on Knowledge-based Software Engineering Brno, Czech Republic, 2000, pp. 51-58, ISBN 1 58603 060 4 (IOS Press).

[13] Zitzler E.: Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications PhD thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, November 1999.

[14] Horn J., Nafpliotis N., Goldberg D.E.: A Niched Pareto Genetic Algorithm for Multiobjective Optimization, In Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, 1, pp. 82-87, Piscataway, New Jersey, June 1994. IEEE Service Center.

[15] Coello C. A.: An Empirical Study of Evolutionary Techniques for Multiobjective Optimization in Engineering Design. PhD thesis, Department of Computer Science, Tulane University, New Orleans, LA, April 1996.

[16] Srinivas N., Deb K.: Multiobjective Optimization using Nondominated Sorting in Genetic Algorithm. Evolutionary Computation, Vol. 2, No. 3, 1994, pp. 221-248.

[17] Corne D. W., et al.: The Pareto Envelope-based Selection Algorithm for Multiobjective Optimization. Proceedings of the Sixth International Conference on Parallel Problem Solving from Nature (PPSN VI), pages 839-848, Berlin, September 2000. Springer.

[18] Laumanns M., et al.: On the Convergence and Diversity-Preservation Properties of Multi-Objective Evolutionary Algorithm. Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, May 2001.

[19] Cvetkovicz D., Parmee, I.: Evolutionary Design and Multi-objective Optimization. EU-FIT'98, September 7-10, 1998, ELITE Foundation, Aachen, Germany, pp. 397-401.

[20] Schwarz J., Očenášek J.: Evolutionary Multi-objective Bayesian Optimization Algorithm: Experimental Study, Proceedings of the 35th Spring International Conference MOSIS'01, Vol. 1, MARQ Ostrava, Hradec nad Moravicí, 2001, pp. 101-108, ISBN 80-85988-57-7.

[21] Schwarz J., Očenášek J.: Pareto Bayesian Optimization Algorithm for the Multiobjective 0/1 Knapsack Problem, Proceedings of the 7th International Mendel Conference on Soft Computing, Brno University of Technology, Faculty of Mechanical Engineering, Brno, 2001, pp. 131-136, ISBN 80-214-1894-X.

[22] Očenášek J., Schwarz J.: The Distributed Bayesian Optimization Algorithm for Combinatorial Optimization, EUROGEN 2001 - Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems, Athens, Greece, September 19-21$^{st}$, 2001, accepted paper.

**Schwarz Josef, Očenášek Jiří:**

**Ratio cut hypergraph partitioning using BDD based MBOA optimization algorithm.**

A-8

Bernd Straube
Erik Jan Marinissen
Zdeněk Kotásek
Ondřej Novák
Jan Hlavička
Richard Růžička (Eds.)

# IEEE Design and Diagnostics of Electronic Circuits and Systems

Fifth International Workshop IEEE DDECS 2002
Brno, Czech Republic, April 17–19, 2002
Proceedings

Organized by
Brno University of Technology
Faculty of Information Technology
Brno, Czech Republic

Co-sponsored by
IEEE Computer Society
Test Technology Technical Council

# RATIO CUT HYPERGRAPH PARTITIONING USING BDD BASED MBOA OPTIMIZATION ALGORITHM

Josef Schwarz
Brno University of Technology
Faculty of Information Technology
Bozetechova 2, 612 66 Brno, CZ
schwarz@fit.vutbr.cz

Jiri Ocenasek
Brno University of Technology
Faculty of Information Technology
Bozetechova 2, 612 66 Brno, CZ
ocenasek@fit.vutbr.cz

**Abstract.** *This paper deals with the k-way ratio cut hypergraph partitioning utilizing the Mixed discrete continuous variant of the Bayesian Optimization Algorithm (mBOA). We have tested our algorithm on three partitioning taxonomies: recursive minimum ratio cut, multi-way minimum ratio cut and recursive minimum cut bisection. We have also derived a new approach for modeling of Boolean functions using binary decision diagrams (BDDs) which are primarily used as a probabilistic model of the mBOA algorithm.*

## 1 Introduction

Partitioning problem is investigated in many research papers. A general survey of partitioning methodologies is presented in [1] and a comparison of several partitioning approaches was presented in [2] and [3]. Their common conclusion pointed out that the ratio cut partitioning gives the best results. The frequently mentioned reference involving the ratio cut criterion is that by Wei and Cheng [4]. They used move-based heuristic algorithm where graph node movement from one partition to the other is controlled by current ratio cut.

Recently, a generalization of the ratio cut implementation was published in [5] where the pin count estimated by Rent's rule was used. Problems of recursive balanced bisection are solved in [6]. A new enhancement of min-cut partitioning (useful for placement) was published in [7]. It validates the multilevel partitioning paradigm for hypergraphs with efficient implementation using benchmarks published in [8].

A separate class of optimizers involves genetic algorithms (GAs). An interesting study on graph partitioning using hard theoretical benchmarks was published in [9]. The newest research based on memetic algorithms and analysis of the fitness landscape was presented in [10]. Estimation distribution algorithms (EDAs) represent a new class of efficient optimizers. Unlike the standard GAs the crossover and mutation operators are replaced by probability estimation and sampling techniques. In other words, statistics about the search space is explicitly maintained by creating probabilistic models of the good solutions found. We have focused especially on the Bayesian Optimization Algorithm (BOA). Our first experience with these techniques was presented in [11] and [12] where minimum-cut bisection was tested and the ability to find global optima was presented. The multi-objective partitioning problem was published in [13] and [14]. The application of recursive partitioning for placement problem was presented in [15].

During our research we have implemented and tested a new Mixed discrete continuous variant of the Bayesian Optimization Algorithm (mBOA) to test its ability for multi-way partitioning. It follows the basic theory and implementation of the BOA initially published in [16], [17] and [18]. We used the concept of binary decision diagrams (BDDs) [19] as graphical probabilistic model approach, including our reformulated standard formulae of the Bayes - Dirichlet metric [20]. In addition, we extended the concept of BDDs to be able to process discrete and continuous domain, that is necessary for the direct (parallel) multi-way hypergraph partitioning.

The remainder of our paper is organized as follows. The specification of partitioning problem is done in the next section. Probabilistic model and mBOA algorithm are described in the third and fourth sections. An extra application of binary decision diagram for Boolean function modeling is presented in the fifth section. The experimental results for hypergraph partitioning are presented in the sixth section.

## 2 Problem specification

Hypergraph partitioning is a well known problem of graph theory. It means dividing hypergraph into disjoints subhypergraphs/modules each containing a subset of nodes. The partitioning is done using various criterions. Generally, the criterion is to minimize the number of hyperedges that have nodes in different partitions or alternatively the minimization of pin count is used.

The particular bi-partitioning problem can be defined as follows: Let us assume a hypergraph $H=(V,E)$, with $n=|V|$ nodes and $m=|E|$ edges. The goal is to find such a partition $(V1,V2)$ of $V$ that minimizes the number of hyperedges that have nodes in different set $V1$, $V2$ (see (1)) under the defined constraint of the partition size. It can be also expressed by the balance/unbalance of the partition size (see (2)). The set of external hyperedges can be labeled as $E_{cut}\ (V1,V2)$ and the following cost function is defined ($\cdot$ symbol represents the multiplication operation):

$$C1\ (V_1,V_2) = |E_{cut}\ (V_1,V_2)| = |\ \{e\in E\ |\ e\cap V_1 \neq \emptyset,\ e\cap V_2 \neq \emptyset\}\ | \qquad (1)$$

with constraint

$$C2(V_1,V_2) = |\,|V_1|-|V_2|\,| <= \alpha V,\ \ \alpha\in <0,1) \qquad (2)$$

Another form of balance used in ratio cut metric is expressed in product form:

$$C3(V_1,V_2) = |V_1|\cdot|V_2| = |V_1|\cdot|V-V_1| \qquad (3)$$

Therefore the ratio cut partitioning that can be specified by the criterion (4) was introduced:

$$RC = C1\ /\ (|V1|\cdot|V2|) = C1/C3 \qquad (4)$$

The ratio cut formulation allows the tradeoff between nets cut and the balance value during the partitioning. The numerator represents the minimum-cut criterion while denominator favours near-bisection. Our goal is to test the performance of the newly designed advanced mBOA algorithm for three main partition taxonomies: recursive minimum ratio cut, multi-way (parallel) minimum ratio cut and recursive minimum cut bisection. We used mainly hard artificial benchmarks with known global optimum and high nonlinearity/epistasis of instances.

## 3 Solution encoding in evolutionary algorithms

Using the population based evolution algorithm the solution of bi-partitioning is represented by binary string:

$X=(X_0, X_1,..,X_{n-1})$ is a string/solution of length $n$ with $X_i$ as a variable,
$x=(x_0, x_1,...,x_{n-1})$ is a string/solution with $x_i\in\{0,1\}$ as a possible instantiation of variable $X_i$.

In case of direct *k*-way partitioning the encoding of solution uses an alphabetic string. Each variable can acquire *k* distinct alleles. For the simplest case of 2 - way partitioning/bisection of a simple graph *G(V,E,W)* we derived on the binary string $X=(x_0, x_1, \dots, x_{n-1})$ the following quadratic cost function:

$$C1 = Ecut\,(V1,V2) = \sum_{\substack{i=0 \\ j>i}}^{n-1} w_{ij}(x_i + x_j - 2x_i x_j) \tag{5}$$

with the balance value $C2 = \left| \sum_{i=0}^{n-1} x_i - \sum_{i=0}^{n-1}(1-x_i) \right|$, (6)

where the coefficient $w_{ij}=1$ in case that net/edge exists between node *i* and *j*, else $w_{ij}=0$. The balance *C3* can be expressed by term:

$$C3 = \sum_{i=0}^{n-1} x_i \bullet \sum_{i=0}^{n-1}(1-x_i) \tag{7}$$

## 4 Standard Bayesian optimization algorithm (BOA)

The principle of BOA algorithms that work on the basis of probabilistic models can be specified on the following framework:

```
Generate initial population of individuals of size M (randomly);
While termination criteria is false do
begin
  Select parent population of N individuals according to a selection method;
  Estimate the probability distribution of the selected parents;
  Generate new offspring according to the estimated probabilistic model;
  Replace some individuals in current population with generated offspring;
end
```

### 4.1 Bayesian network (BN)

The original Bayesian Optimization Algorithm [17] operates on the population of strings/chromozomes of *n* binary variables/genes. The Bayesian Network is learned in each generation how to encode the structure of promising solutions. The following step includes a sampling process to discover the promising areas of the search space. In BN for each variable $X_i$ a set of parent variables $\Pi_{X_i}$ is defined which it depends on, so the distribution of individuals is expressed by the conditional probabilities:

$$p(X) = \prod_{i=0}^{n-1} p\left(X_i \mid \Pi_{X_i}\right) \tag{8}$$

Generally, the existence of a directed edge from $X_j$ to $X_i$ in the network implies the belonging of the variable $X_j$ to the set $\Pi_{X_i}$.

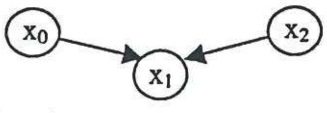


Figure 1: Example of BN with 3 nodes, where $X_0$ and $X_2$ are independent and $X_1$ depends on $X_0$ and $X_2$

## 4.2 Binary decision diagram (BDD)

Additional accuracy and efficiency can be achieved by utilizing decision trees or diagrams in Bayesian network, see [19]. For the BN in Fig. 1, the corresponding contingency table for $X_1$ would have 4 rows, one row for each possible instance of the substring $(X_0,X_2)$. Now let us suppose that in some case $p(X_1|10) = p(X_1|11)$ and $p(X_1|00) \neq p(X_1|01)$. It is evident, that if $X_0=1$ the value of $X_1$ does not depend on $X_2$. This allows to reduce the number of table rows (* is a don't-care symbol).

Table 1. Simplified contingency table by don't-care symbols.

| $X_0$ | $X_2$ | consequent $p(X_1)$ |
|---|---|---|
| 0 | 0 | $p(X_1|00)$ |
| 0 | 1 | $p(X_1|01)$ |
| 1 | * | $p(X_1|1*)$ |

This situation can be expressed by a decision tree (see Fig. 2). Each variable, which determines the $X_1$ value corresponds to one or more split nodes in the tree. Each row in the Tab.1 corresponds to one leaf of the tree and each leaf determines $p(X_1)$ among the individuals fulfilling the split conditions on the path from the root:

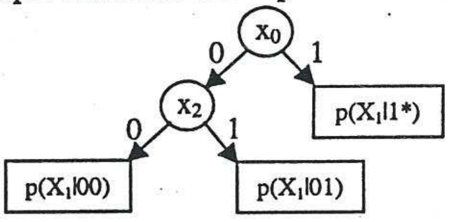


Figure 2: Binary decision tree for the determination of $X_1$

Next advantage of decision trees lies in low complexity of their building – the step of adding new split node is easy to evaluate by the metric (9) – it splits only one row in the contingency table. From the Bayes-Dirichlet metrics (BD) we derived the incremental equation for adding one new binary split:

$$Gain(X_i,X_j)=\frac{\sum_{r\in\{0,1\}}\sum_{s\in\{0,1\}}\Gamma(m_{r,s}+1)\cdot\Gamma\left(\sum_{r\in\{0,1\}}\sum_{s\in\{0,1\}}(m_{r,s}+1)\right)}{\sum_{r\in\{0,1\}}\Gamma\left(\sum_{s\in\{0,1\}}(m_{r,s}+1)\right)\cdot\sum_{r\in\{0,1\}}\Gamma\left(\sum_{s\in\{0,1\}}(m_{r,s}+1)\right)}, \tag{9}$$

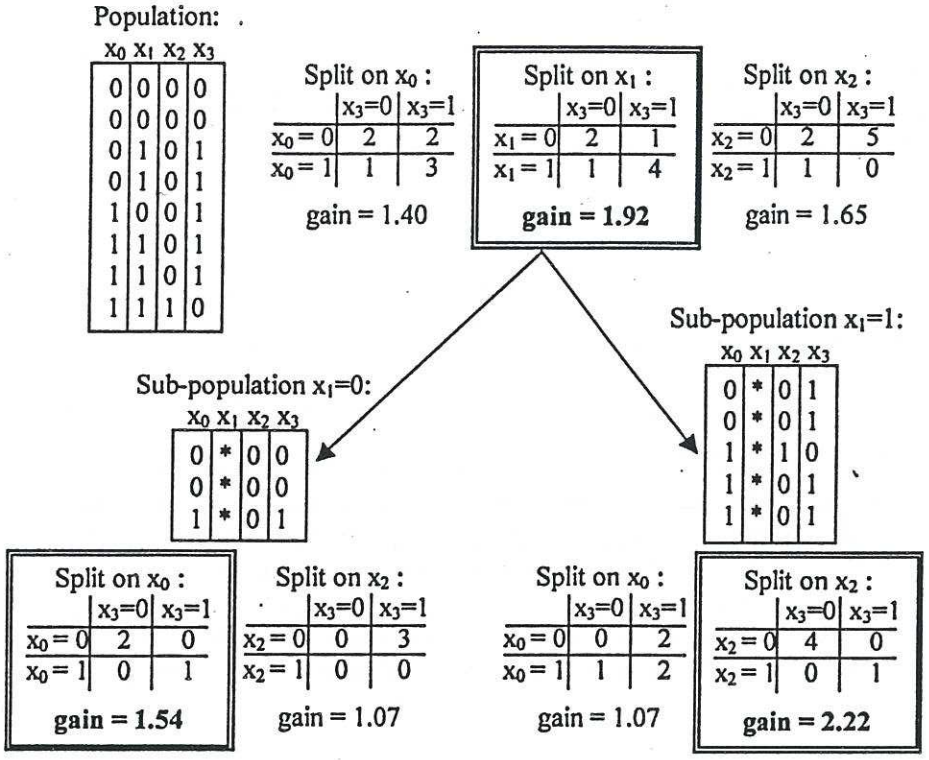
where $X_i$ is the child variable, $X_j$ is the parent variable - possible split, and $m_{r,s}$ is the number of individuals having $X_j=r$ and $X_i=s$. Note that the splitting is performed recursively, so $m_{r,s}$ is determined only from the subpopulation being split.

Table 2. Notation of symbols $m_{r,s}$

| | $X_i = 0$ | $X_i = 1$ |
|---|---|---|
| $X_j = 0$ | $m_{0,0}$ | $m_{0,1}$ |
| $X_j = 1$ | $m_{1,0}$ | $m_{1,1}$ |

Moreover, the gain of each split operation can be penalized by the model complexity (e.g. the depth of the tree), which removes the need of the limitation of the number of parents.

The algorithm for building the binary decision tree from the population can be illustrated in the following figure:



Population:

| $x_0$ | $x_1$ | $x_2$ | $x_3$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

Split on $x_0$ :

| | $x_3$=0 | $x_3$=1 |
|---|---|---|
| $x_0$ = 0 | 2 | 2 |
| $x_0$ = 1 | 1 | 3 |

gain = 1.40

Split on $x_1$ :

| | $x_3$=0 | $x_3$=1 |
|---|---|---|
| $x_1$ = 0 | 2 | 1 |
| $x_1$ = 1 | 1 | 4 |

**gain = 1.92**

Split on $x_2$ :

| | $x_3$=0 | $x_3$=1 |
|---|---|---|
| $x_2$ = 0 | 2 | 5 |
| $x_2$ = 1 | 1 | 0 |

gain = 1.65

Sub-population $x_1$=0:

| $x_0$ | $x_1$ | $x_2$ | $x_3$ |
|---|---|---|---|
| 0 | * | 0 | 0 |
| 0 | * | 0 | 0 |
| 1 | * | 0 | 1 |

Sub-population $x_1$=1:

| $x_0$ | $x_1$ | $x_2$ | $x_3$ |
|---|---|---|---|
| 0 | * | 0 | 1 |
| 0 | * | 0 | 1 |
| 1 | * | 1 | 0 |
| 1 | * | 0 | 1 |
| 1 | * | 0 | 1 |

Split on $x_0$ :

| | $x_3$=0 | $x_3$=1 |
|---|---|---|
| $x_0$ = 0 | 2 | 0 |
| $x_0$ = 1 | 0 | 1 |

**gain = 1.54**

Split on $x_2$ :

| | $x_3$=0 | $x_3$=1 |
|---|---|---|
| $x_2$ = 0 | 0 | 3 |
| $x_2$ = 1 | 0 | 0 |

gain = 1.07

Split on $x_0$ :

| | $x_3$=0 | $x_3$=1 |
|---|---|---|
| $x_0$ = 0 | 0 | 2 |
| $x_0$ = 1 | 1 | 2 |

gain = 1.07

Split on $x_2$ :

| | $x_3$=0 | $x_3$=1 |
|---|---|---|
| $x_2$ = 0 | 4 | 0 |
| $x_2$ = 1 | 0 | 1 |

**gain = 2.22**

Figure 3: Building binary decision tree for $x_3$

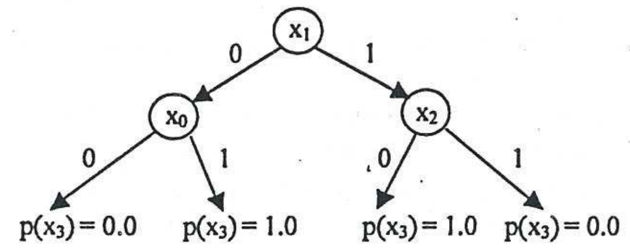


Figure 4: Final binary decision tree for $x_3$

### 4.3 BDD for mixed continuous discrete mBOA algorithm

The idea of the utilizing BDD in BOA algorithm was mentioned for the first time in [19]. In our Mixed Bayesian Optimization Algorithm (mBOA) we extended the idea of decision diagram to continuous and integer domains. Our mBOA is the only one EDA which is able to solve problems with mixed real-discrete parameters (alleles) without conversion to binary representation. In Fig. 5 an example with continuous child variable $X_i$ and continuous parent variable $X_j$ is shown. Our algorithm tries to find a $X_j$ and $X_i$ boundaries such that the numbers of individuals in each quadrant maximize (9). In the case of integer domain (see Fig. 6) we use hill-climbing algorithm to split the set of possible $X_j$ values into left and right subset. The variable $s$ in (9) goes through all possible $X_i$ values instead of only two values *{0,1}*.

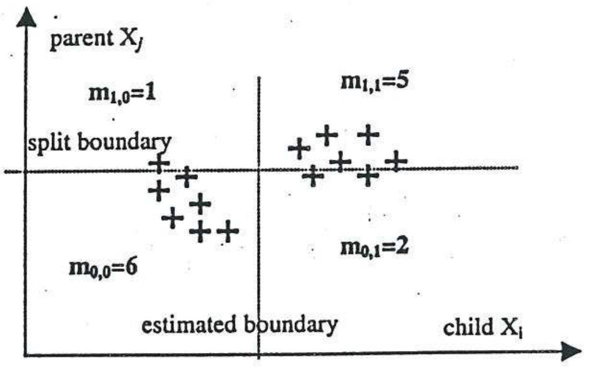


Figure 5: An example of real domain split

|  | $X_i = 0$ | $X_i = 1$ | $X_i = 2$ | $X_i = 3$ |
|---|---|---|---|---|
| $X_j \in \{0,2\}$ | $m_{0,0}$ | $m_{0,1}$ | $m_{0,2}$ | $m_{0,3}$ |
| $X_j \in \{1,3\}$ | $m_{1,0}$ | $m_{1,1}$ | $m_{1,2}$ | $m_{1,3}$ |

Figure 6: An example of integer domain split

### 4.4 Probabilistic model sampling

As a result of probabilistic model construction we obtain a set of decision trees, one tree for each variable. This set of trees is used for generation of the offspring (new population) during the Probabilistic Logic Sampling (PLS). During this PLS process the variables (whose parents are already determined) are generated by traversing their decision trees. This is repeated for each offspring until all its variables are generated.

In our example in Fig. 1 the variables $X_0$ and $X_2$ are generated as first. Then the concrete value of $p(X_1)$ is determined according to concrete values of $X_0$ and $X_2$ - using decision tree from Fig. 2.

### 4.5 Parallel BDD construction

The probabilistic model construction is the most time consuming task in mBOA. We are currently working on the distributed mBOA using Message Passing Interface (MPI). The goal is to utilize more processors when searching for a good model. Our consolation is that the BD metric is separable and can be written as a product of $n$ factors, where $i$-th factor expresses the quality of decision tree for variable $X_i$. It is possible to use up to $n$ processors, each processor has its own local copy of parent population and it builds tree for different variable. The addition of splits/parents to the trees is parallel, so we need an additional mechanism to keep the mutual dependencies acyclic. In [21] we proposed the concept of restricted set of parents in BN. We are going to extend this concept for BDD in mBOA. In each generation, variables will be ordered in advance, according to a random permutation vector $q = (q_0, q_1, ... q_{n-1})$. Each decision tree of variable $X_i$ may contain only such parental splits $X_j$ having $q_j < q_i$. This approach ensures linear scalability, because no communication overhead is required. In addition, scalable methods for overlapping the communication latency during generation, evaluation and broadcasting of new population among the processes will be implemented using the farmer-workers architecture.

## 5 BDD diagram as a model of Boolean function

Binary Decision Diagrams are commonly used for representation of Boolean functions because of their efficiency in terms of time and space. The BDDs are capable to improve many conventional algorithms significantly. Besides Boolean function, BDDs can be also used for representation of other types of discrete functions, such as multi-valued functions, cube sets and arithmetic formulas [22]. BDDs or oriented BDDs (OBDDs) can be constructed from Binary Decision Tree (BDT) using two basic reduction rules: 1) reduction of nodes with unique ancestor nodes and 2) sharing all equivalent sub-graphs. The previous two-phase approach suffers from the necessity of a specification of the node ordering in the first phase. The node ordering is often provided by genetic algorithms [23]. In another approaches the ordering is found dynamically during the process of BDD building. We suggest a new approach to build BDDs for a Boolean function (see Fig. 7 and Fig. 8). This approach using BD metric is similar to the previous one presented in Fig. 4 in case that we interpret the variable $x_3$ as a Boolean function. The function must be represented by truth table. But this can be considered as a bottleneck of the method. The advantage of suggested approach lies in the implicit ordering of BDD nodes and lower BDT complexity.

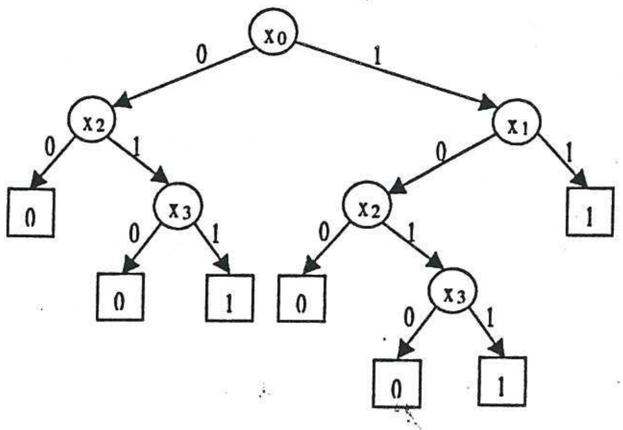


Figure 7: BDT-model of Boolean function $F=x_0x_1+x_2x_3$

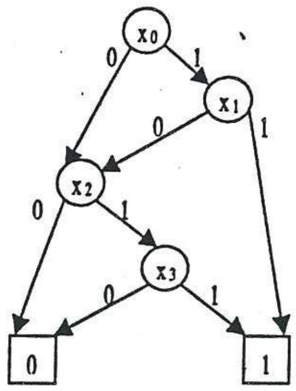


Figure 8: BDD after removing duplicate subgraphs

## 6 Experimental results

### 6.1 Test benchmarks

The complexity of the partitioning problem is determined by the type and complexity of the instances – benchmark graphs. We used four types of benchmarks - three artificial ones (like in the excellent experimental studies [9], [10] and in [11]) with known global optimum (see Fig. 9) and one benchmark consisting of two real circuits (random logic) from benchmark package [8]:

1. Regular graphs *Grid_n* with square grid structure, where the notion *n* specifies the number of nodes. Graphs *GridH_n* have a 2-edge asymmetrical horizontal bottle-neck and *GridHV_n* have an extra 2-edge asymmetrical vertical bottle-neck. As an example a quadrisection of asymetric graph *GridH_100* is represented in Fig. 9.
2. Random geometric graph *U_n.d* with *n* vertices placed in the unit square. The coordinates of vertices are chosen randomly with uniform distribution. An edge exists between two vertices if their Euclidean distance is $l$ or less, where the expected vertex degree is specified by $d = n\pi l^2$. We have chosen $n=120$, $d=5$.
3. Caterpillar graphs *CAT_n.k*, with *k* articulations, *(n-k)/k* legs for each articulation and *n* nodes.
4. Real circuits labeled by *IC_n*. The hypergraph IC_67 consists of 67 nodes and 138 edges/nets, the IC_116 consists of 116 nodes and 329 edges/nets.

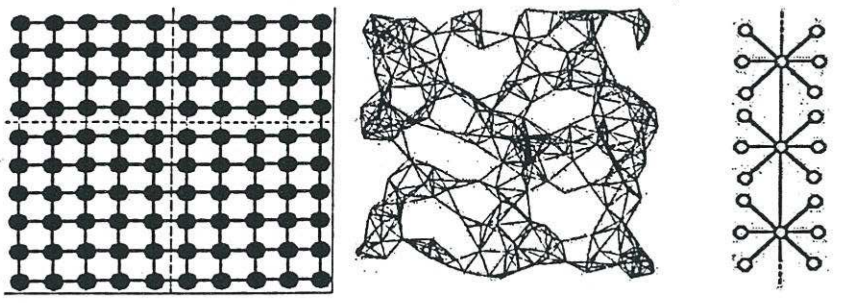


Figure 9: Graph structure of *GridH_100, U_120.5* and a segment of the *CAT_21.3 graph.*

## 6.2 Summary of experimental results

Experimental results are summarized in Tab. 3 and partly in Fig. 10. Our algorithm is capable to solve the top-down *k*-way hypergraph partitioning but for the better preview we present results for *4*-way partitioning only. We have arranged 3 types of experiments:

- Ratio cut partitioning by recursive bi-partitioning RRC
- Ratio cut partitioning by non recursive multi-way partitioning MRC
- Partitioning by recursive bisection RB

The ratio cut value is presented in the inverse mode, because this mode for fitness representation is used. The 1/RC values printed in bold represent the global optimum.

Table 3. Partitioning results for four types of graphs and three algorithms. The population size was set to N=40*n for RRC and RB, N=100*n for MRC, computation time is expressed in seconds.

| | | | RRC | | MRC | | RB | |
|---|---|---|---|---|---|---|---|---|
| Benchmark | n | 1/RC optimum | 1/RC cut size | Time Eval. | 1/RC cut size | Time Eval. | 1/RC cut size | Time Eval. |
| Grid_64 | 64 | **3855.1** | **3855.1**<br>16 | 179<br>35939 | 3626.6<br>17 | 2574<br>233600 | **3855.1**<br>16 | 479<br>80316 |
| GridH_64 | 64 | **5236.4** | **5236.4**<br>10 | 136<br>31864 | **5236.4**<br>10 | 2392<br>214400 | 3640.9<br>17 | 479<br>87819 |
| GridHV_64 | 64 | **10125.0** | **10125.0**<br>4 | 134<br>31830 | **10125.0**<br>4 | 1979<br>168600 | 4369,1<br>14 | 483<br>81253 |
| Grid_100 | 100 | **18601.2** | **18601.2**<br>20 | 1041<br>78000 | 17857.1<br>20 | 15258<br>495000 | **18601.2**<br>20 | 3406<br>219000 |
| GridH_100 | 100 | **27692.3** | **27692.3**<br>12 | 867<br>75400 | 26584.6<br>12 | 12617<br>385000 | 18601.2<br>20 | 3405<br>221000 |
| GridHV_100 | 100 | **66355.2** | **66355.2**<br>4 | 850<br>67440 | **66355.2**<br>4 | 10931<br>330000 | 22977.9<br>16 | 3491<br>220000 |
| IC_67 | 67 | - | 1006.9<br>64 | 387<br>88521 | **1165.9**<br>65 | 2647<br>219000 | 1091.8<br>71 | 932<br>167232 |
| IC_116 | 116 | - | 7253.33<br>62 | 1554<br>93344 | **8250.7**<br>80 | 17977<br>435000 | 7113.5<br>95 | 3956<br>204477 |
| U_120.5 | 120 | - | **15669.5**<br>31 | 2613<br>129047 | 14168.0<br>44 | 30111<br>594000 | 12272.7<br>65 | 5771<br>247200 |
| CAT_105.7 | 107 | - | **101250.0**<br>3 | 933<br>66750 | **101250.0**<br>3 | 12358<br>336000 | 29659.5<br>15 | 5292<br>295679 |

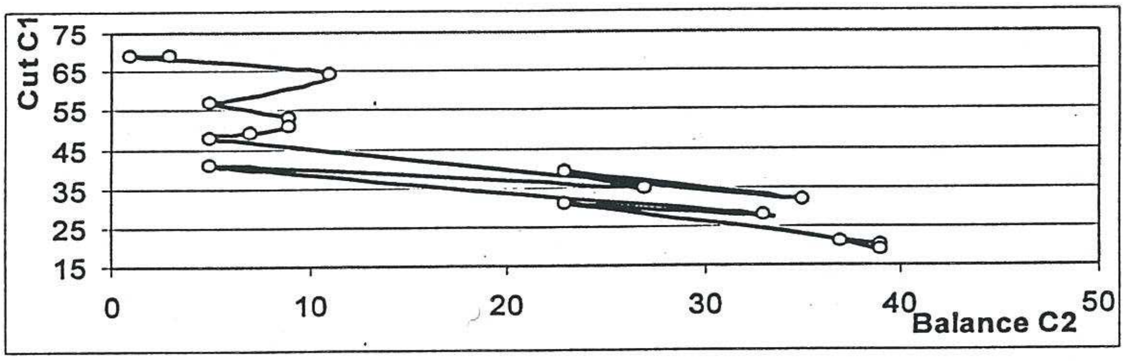


Figure 10: Relation between criterion *C1* and *C2* during the RRC minimization of the *IC67* hypergraph.

## 7 Conclusions

We have implemented a new advanced evolutionary algorithm mBOA for multi-way partitioning of hypergraphs. Its main advantage against the mostly used move-based heuristic methods lies in the ability to discover and determine the amount of epistasis in a given problem instance and to find the optimal solution. The Bayesian statistics and BDD diagrams used in probabilistic model cause high performance of mBOA. It is evident from Tab. 3 that the recursive ratio cut algorithm RRC is always able to find the known global optimum in case of artificial graphs. The non recursive multi-way ratio cut algorithm MRC provides almost comparable results to the RRC but the time complexity is very high – approximately 14 times greater than for the RRC. The recursive bisection algorithm RB is worse in more cases in comparison with the previous two algorithms. An example of the optimization process for RRC algorithm is shown in Fig. 10. The white nodes on the optimization curve represent the best solution in each generation. It can be recognized how the algorithm gradually searches for the minimum cut size value resulting in greater balance value *C2*. The future activity will be focused on the sophisticated RRC algorithm including the external pin count, Rent's rule and the massive parallelization of the mBOA algorithm. The research will be also directed towards the enhancement of the suggested approach to the modeling of Boolean functions using BDDs with BD metric.

## Acknowledgments

This research has been carried out under the financial support of the Research intention no. CEZ: J22/98: 262200012-"Research in information and control systems" (Ministry of Education, CZ) and the research grant GA 102/02/0503 "Parallel system performance prediction and tuning" (Grant Agency of Czech Republic).

## References

[1] Alpert, C. J., Kahng, A. B.: Recent Directions in Netlist Partitioning: A Survey, Integration: The VLSI Journal 19 (1995), pp. 1-81.

[2] Agrawal, B., Narendran ,N., Shivakumar, N.: Multi-Way VLSI Circuit Partitioning. Proceedings of 9th International Conference on VLSI Design, Bangalore, India, Jan' 96, pp.-17.

[3] Hagen, L., Kahng A. B., Kurdahi F.: "On the Intrinsic Rent Parameter and New Spectra-Based Methods for Wireability Estimation". IEEE Transaction on CAD 13(1), January 1994, pp. 27-37.

[4] Wei, Y.- C., Cheng, C.-K: Ratio Cut Partitioning for Hierarchical design, IEEE Trans. Computer Aided Design Integrated Circuit & System, Vol.10 (No.7), pp. 911-921, July 1991.

[5] Stroobandt, D.: Pin Count Prediction in Ratio Cut Partitioning for VLSI and ULSI. In M. A. Bayoumi, Editor, Proceedings of the IEEE International Symposium on Circuits and Systems, Pages VI/262-VI/265, May 1999.

[6] Simon, H. D., Teng, S.-H.: How Good is Recursive Bisection?, SIAM J. Scientific Computing 18 (5) (1997), pp. 1436-1445.

[7] Karypis, G., Kumar, V.: Hmetis - A Hypergraph Partitioning Package, Version 1.5.3, University of Minnesota, Department of Computer Science & ENGINEERING, Army HPC Research Center Minneapolis. http://www-users.cs.umn.edu/~karypis/metis/hmetis/download.shtml.

[8] A Benchmark Set, University of California, Los Angeles, VLSI CAD Laboratory, http://vlsicad.cs.ucla.edu/~cheese/benchmarks.html.

[9] Than Nguen Bui, Byung Ro Moon: Genetic Algorithm and Graph Partitioning. IEEE Transactions on Computers, Vol. 45, No.7, July 1996, pp. 841-855.

[10] Merz P., Freisleben, B.: Fitness Landscapes, Memetic Algorithms and Greedy Operators for Graph Bi-Partitioning. Evolutionary Computation, Vol. 8, No. 1, pp. 61-91, 2000. Preprint Available as Technical Report No. 98-01 (Informatik-Berichte).

[11] Schwarz, J., Očenášek, J.: Experimental Study: Hypergraph Partitioning Based on the Simple and Advanced Genetic Algorithm BMDA and BOA, Proceedings of the Mendel' 99 Conference, Brno University of Technology, 1999, pp. 124-130, ISBN 80-214-1131-7.

[12] Schwarz, J., Očenášek, J.: The Knowledge-based Evolutionary Algorithm KBOA for Hypergraph Bisectioning. Proceedings of the Fourth Joint Conference on Knowledge-based Software Engineering Brno, Czech Republic, 2000, pp. 51-58, ISBN 1 58603 060 4 (IOS Press).

[13] Schwarz, J., Očenášek, J.: Multiobjective Bayesian Optimization Algorithm for Combinatorial Problems: Theory and Practice, Neural Network World, Vol.11, No.5, 2001, Published by Academy of Science Czech Republic, pp.423-441, ISSN 1210-0552.

[14] Schwarz, J., Očenášek, J.: Evolutionary Multiobjective Bayesian Optimization Algorithm: Experimental Study. Proceedings of the 35th Spring International Conference MOSIS' 01, Vol. 1, MARQ Ostrava, Hradec nad Moravicí, 2001, p. 101-108, ISBN 80-85988-57-7.

[15] Schwarz, J., Očenášek, J.: Partitioning-oriented Placement Based on Advanced Genetic Algorithm BOA, Proceedings of the 6th International Mendel Conference on Soft Computing, Brno University of Technology, Faculty of Mechanical Engineering, Brno, 2000, pp . 145-150.

[16] Pelikan, M., Goldberg, D. E., & Cantú-Paz, E.: Linkage Problem, Distribution Estimation, and Bayesian Networks. IlliGal Report No. 98013, November 1998, pp. 1-25.

[17] Pelikan, M.: A Simple Implementation of Bayesian Optimization Algorithm in C++ (Version 1.0). Illigal Report 99011, February 1999, pp. 1-16.

[18] Pelikan, M., Goldberg, D. E., & Lobo, F.: A Survey of Optimization by Building and Using Probabilistic Model, Illigal Report 99018, September 1999, pp. 1-12.

[19] Pelikan, M., Goldberg, E., Sastry, K.: Bayesian Optimization Algorithm, Decision Graphs, and Occams Razor. Illigal Report No. 2000020, May 2000, pp.1-24.

[20] Heckerman, D., Geiger, D., & Chickering, M. (1994). Learning Bayesian Networks: The combination of Knowledge and Statistical Data (Technical Report MSR-TR-94-09), Redmont, WA: Microsoft Research, 1995, pp. 1-53.

[21] Očenášek, J., Schwarz, J.: The Distributed Bayesian Optimization Algorithm, Proceeedings of the Eurogen 2001 - Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems, The National Technical University of Athens, Greece, 19-21 September 2001, in print.

[22] Sasao T., Fujita, M., editors: Representations of Discrete Functions. Kluwer Academic Publisher, London 1996.

[23] Drechsler R.: Evolutionary algorithms for VLSI CAD. Kluwer Academic Publishers, London 1998, ISBN 0-7923-8168-8.

| | |
|---|---|
| **Schwarz Josef, Očenášek Jiří:**<br>**Bayes-Dirichlet BDD as a probabilistic model for logic function and evolutionary circuit decomposer.** | A-9 |

# MENDEL 2002

8th International Conference on Soft Computing

Evolutionary Computation, Genetic Programming, Fuzzy Logic, Rough Sets, Neural Networks, Fractals, Bayesian Methods

June 5–7, 2002, Brno, Czech Republic

# BAYES-DIRICHLET BDD AS A PROBABILISTIC MODEL FOR LOGIC FUNCTION AND EVOLUTIONARY CIRCUIT DECOMPOSER

Josef Schwarz
Jiří Očenášek

Brno University of Technology
Faculty of Information Technology
Department of Computer Systems
CZ - 61266 Brno, Božetěchova 2

Tel.: 420 5 41141210, fax: 41141270, e-mail: schwarz@dcse.fee.vutbr.cz,
Tel.: 420 5 41141206, fax: 41141270, e-mail: ocenasek@dcse.fee.vutbr.cz

*Abstract: This paper deals with the utilizing of Binary Decision Diagrams built on the Bayes-Dirichlet metric for representation of logic functions. In the first phase the Binary Decision Tree is built followed by a reduction process resulted in binary decision diagram (BDD). The BDDs are also used as a probabilistic model for advanced Bayesian decomposer for digital circuit partitioning. It is shown that this approach is more efficient than the utilizing of Bayesian networks (BN) and in addition the concept of BDDs parallelization is simple to implement.*

*Key Words:. Logic function, binary decision diagrams, Bayes-Dirichlet metric, digital circuit partitioning.*

## 1 Introduction

Many "decision" procedures use a branching process which consists of testing some property with the branch depending on the test outcome (typical examples are identification of objects or classification). Concrete application include also simulation or modeling of digital circuits [1]. This branching process can be represented usually by graph structure called decision diagram (DD). A decision diagram is a directed acyclic graph in which each decision node is labelled by a variable/attribute tested in this node (control variable/attribute). The edges coming out from the decision node leading to the nodes in subsequent levels correspond to the values of the control variable. The number of the edges relates to the values of the control variable. Besides decision nodes there are leaves (terminal nodes) labelled by the value of the function that is being evaluated by the diagram. Important parameter of a given DD is the size of the DD (the number of decision nodes). The construction of minimum-sized DDs belongs to NP-hard problems. The survey of optimization algorithm for binary DD (BDD) based on top-down as well as bottom-up techniques is published in [2] and [3]. Let it be noted that DD can be seen also as a reduction of decision tree built by the known Shannon's expansion. The key problem is the finding a proper ordering of the DD variables. In this paper we applied a new approach for DD construction based on the Bayes- Dirichlet metric.

## 2 Bayes- Dirichlet metric for BDD building

This technique is based on the construction of binary decision tree (BDT) using Bayes-Dirichlet metric[4] followed by a fast heuristic for reducing of BDT to BDD. We specify its main feature on the case of classification and prediction. BDT is expressed by:

- a leaf (terminal) node - indicates the value of the target attribute (class) of examples
- a decision node (split) - specifies a test to be carried out on a single attribute-value, with one branch and sub-tree for each possible outcome of the test.

A binary decision tree can be used to classify an example by starting at the root of the tree and moving through it towards a leaf node, which provides the classification of the instance. Most algorithms that have been developed for learning decision trees employ a top-down, greedy search through the space of possible decision trees. Our algorithm searches through the attributes of the training instances and extracts the attribute that best separates the given examples. If the attribute perfectly classifies the training sets then it stops, otherwise it recursively operates on the two partitioned subsets to get their "best" attribute. The algorithm uses a greedy search, that is, it picks the best attribute and never looks back to reconsider earlier choices.

The central focus of the binary decision tree growing algorithm is selecting which attribute to test at each decision node in the tree. From the Bayes-Dirichlet metrics (BD) we derived the incremental equation for adding one new binary split. This measure is used to select among the candidate attributes at each step while growing the tree:

$$Gain(X_i, X_j) = \frac{\sum_{r\in\{0,1\}}\sum_{s\in\{0,1\}} \Gamma(m_{r,s}+1) \cdot \Gamma(\sum_{r\in\{0,1\}}\sum_{s\in\{0,1\}}(m_{r,s}+1))}{\sum_{r\in\{0,1\}} \Gamma(\sum_{s\in\{0,1\}}(m_{r,s}+1)) \cdot \sum_{r\in\{0,1\}} \Gamma(\sum_{s\in\{0,1\}}(m_{r,s}+1))}, \qquad (1)$$

where $X_i$ is the target attribute, $X_j$ is the investigated split attribute, and $m_{r,s}$ is the number of individuals having $X_j=r$ and $X_i=s$. For practical purposes we use logarithm of this metric, which avoids multiplication operations. The splitting is performed recursively, so $m_{r,s}$ is determined only from the subpopulation being split. To avoid over-fitting the data, the gain of each split operation can be penalized by the model complexity (e.g. the depth of the tree).

| | $X_i = 0$ | $X_i = 1$ |
|---|---|---|
| $X_j = 0$ | $m_{0,0}$ | $m_{0,1}$ |
| $X_j = 1$ | $m_{1,0}$ | $m_{1,1}$ |

Table 1 Notation of symbols $m_{r,s}$

Note that we also deal with continuous and categorial (integer) variables. If the investigated split attribute $X_j$ is continuous, we use fast heuristics to find the optimal split point $E_j$. The training cases are split according to condition "$X_j<E_j$" into two parts, where the value of target attribute is more determined.

## 2.1 BDD for representation of Boolean function

Binary Decision Diagrams are commonly used for representation of Boolean functions because of their efficiency in terms of time and space. Besides Boolean function, BDDs can be also used for representation of other types of discrete functions, such as multi-valued functions, cube sets and arithmetic formulas [2]. BDD is a rooted acyclic graph G=(V,E) with node set V containing nonterminal/decision and terminal nodes. A decision node $x$ represents the Shannon's expansion of the Boolean function:

$$f = (x_i f_1 + \bar{x}_i f_0) \qquad (2)$$

where $i$ is the index of the decision node, $f_0$ and $f_1$ are the functions of the nodes pointed to by *0-* and *1-* edges, respectively. The terminal nodes represent the logic values *(1/0)*.
The previous approaches suffer from the necessity of a specification of the node ordering. It is often provided by genetic algorithms [3]. In another approaches the ordering is found dynamically during the process of BDD building.

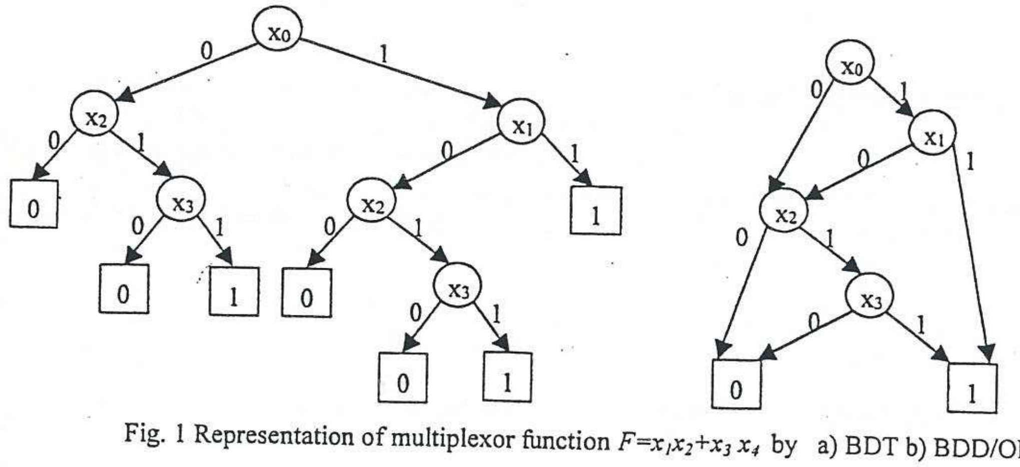


Fig. 1 Representation of multiplexor function $F=x_1x_2+x_3\,x_4$ by a) BDT b) BDD/OBDD

As mentioned above our algorithm constructs decision tree (BDT) from a set of training cases. The logic function truth table is used as the training set – the input variables are candidates for splits/decision nodes and the function value is the target attribute. Then, BDD is constructed from BDT using two basic reduction rules: 1) reduction of nodes with unique ancestor nodes and 2) sharing all equivalent sub-graphs. In Fig. 1 an example of BDT and the final OBDD for multiplexor function is shown. Our approach for building the binary decision tree from truth table is illustrated in the Fig.2 for logic function $F = x_1x_2+x_3$.

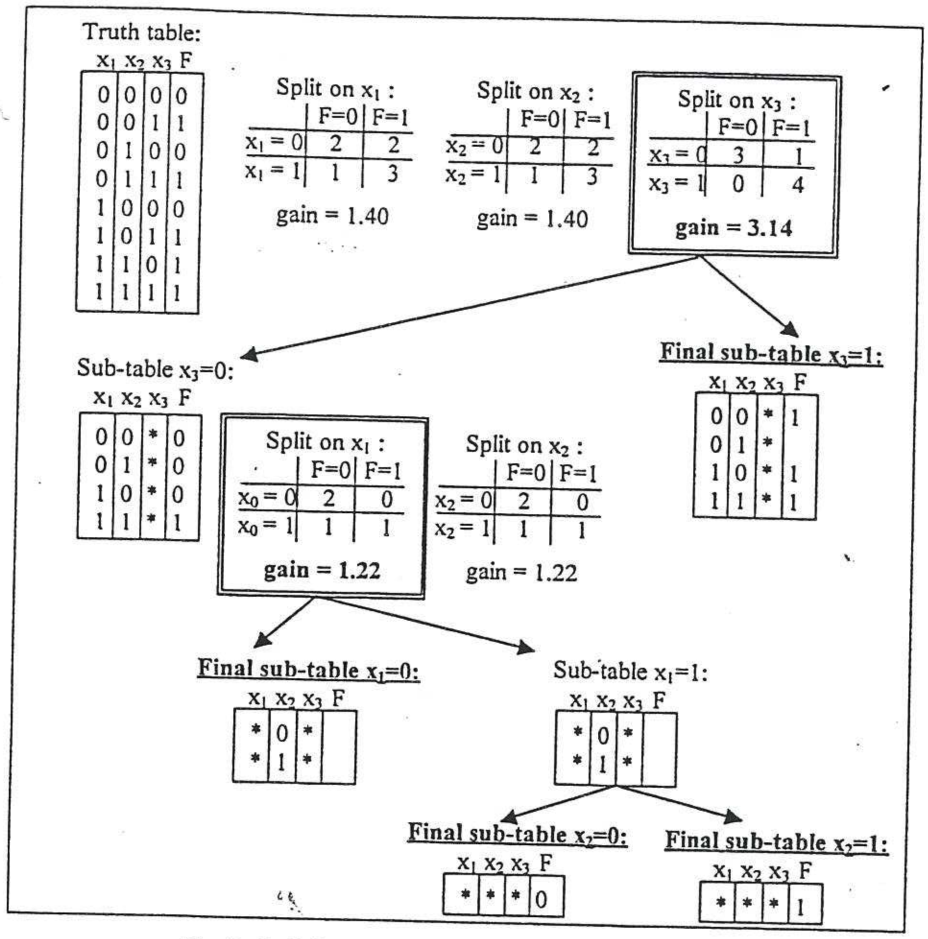
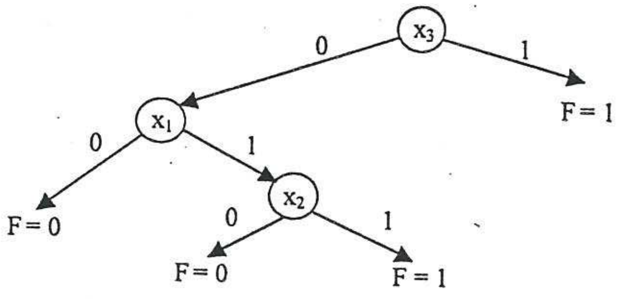


Fig. 2 Building binary decision tree for $F=x_1x_2+x_3$



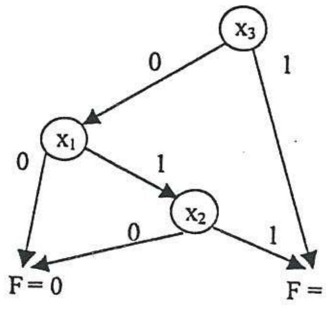


Fig. 3a Binary decision tree for $F = x_1x_2+x_3$

Fig. 3b Binary decision diagram for $F = x_1x_2+x_3$

The function must be represented by truth table. The advantage of suggested approach lies in the implicit ordering of BDD nodes and lower BDT complexity. The worst case time complexity of our BDT construction is $O(n^2 2^n)$. Time complexity of classical simple greedy BDD construction is $O(n!2^n)$ and time complexity of advanced classical algorithm based on dynamic programming is $O(n^2 3^n)$. In [1] the suboptimal bottom-up approach for every ordering of variable with time complexity $O(n^2 2^n)$ is described.

The maximal structure complexity of the BDD (the size of BDD) is given by the number of nonterminal nodes. As mentioned before the structure complexity of BDD is determined by variable ordering. In case of simple Boolean function $F_m = x_1x_2+x_3x_{4+,\dots,}+x_{2n-1}x_{2n}$ the upper bound of structure complexity is $O(2^n-1)$.

We have applied our BDT constructor on the function $F_m$ for increasing number of variables from 4 to 20 with folowing results:

| n - number of variables | 4 | 6 | 8 | 10 | 12 | 14 | 20 |
|---|---|---|---|---|---|---|---|
| m - number of BDT decision nodes | 6 | 14 | 30 | 62 | 126 | 254 | 2046 |
| Number of reduced BDD decision nodes | 4 | 6 | 8 | 10 | 12 | 14 | 20 |

Table 2 Resulting size of BDT/BDD representing the function $F_m$

From this experiment the size complexity of BDT was derived as $O(2^{n/2+1})$. The BDD (OBDD) was derived from BDT by reduction heuristic with time complexity $O(m^2 \log m)$ where $m$ is the size of BDT. The BDD size complexity is $O(n)$.

## 2.2 BDD for multi-valued Boolean function

We extended the idea of decision diagram to integer domains for multi-valued logic function (see Fig. 4). If the target variable $X_i$ is categorial, we use in (1) symbols $m_{ij}$ from the following contingency table with multiple columns (each of them representing the possible value of $X_i$) to evaluate the gain of possible split/decision nodes:

| | $X_i = 0$ | $X_i = 1$ | $X_i = 2$ | $X_i = 3$ |
|---|---|---|---|---|
| $X_j = 0$ | $m_{0,0}$ | $m_{0,1}$ | $m_{0,2}$ | $m_{0,3}$ |
| $X_j = 1$ | $m_{1,0}$ | $m_{1,1}$ | $m_{1,2}$ | $m_{1,3}$ |

Table 3 An example of $m_{ij}$ coefficients for target variable $X_i \in \{0,1,2,3\}$

| $x_0$ | $x_1$ | $x_2$ | F |
|---|---|---|---|
| 0 | 0 | 0 | 3 |
| 0 | 0 | 1 | 5 |
| 0 | 1 | 0 | 3 |
| 0 | 1 | 1 | 4 |
| 1 | 0 | 0 | 4 |
| 1 | 0 | 1 | 4 |
| 1 | 1 | 0 | 4 |
| 1 | 1 | 1 | 5 |

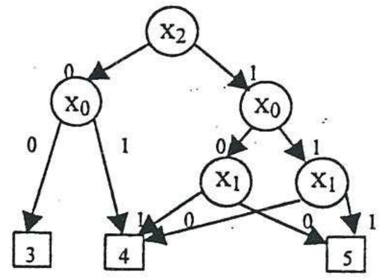


Fig. 4 An example of multi-valued Boolean function a) truth table, b) BDD

## 3 BDD based Bayesian decomposer

BDD based Bayesian decomposer was developed for general problem of system decomposition. We demonstrate its performance on the partitioning of digital circuits, see chapter 4. The decomposer is based on the Bayesian Optimization Algorithm (BOA)[5] and [6] that belongs to the probabilistic model building genetic algorithms where crossover and mutation operators are replaced by probability estimation and sampling techniques. The skeleton of common EDA algorithm (Estimation Distribution Algorithm) can be specified as follows:

```
Generate initial population of individuals of size M (randomly);
While termination criteria is false do
begin
  Select parent population of N individuals according to a selection method;
  Estimate the probability distribution of the selected parents;
  Generate new offspring according to the estimated probabilistic model;
  Replace some individuals in current population with generated offspring;
end
```

Individuals in the population are treated as vectors of instantiations of $n$ random variables $X_i$, each random variable (gene) represents one parameter of a solution

$$X = (X_0, X_1, \dots, X_{n-1})$$

Most methods for automated learning of parameter dependencies in EDAs have been adopted from the area of data mining. We proposed the mBOA algorithm based on our BDT construction. The idea of the utilizing DBT (resp. BDD) in BOA algorithm was mentioned for the first time in [7]. We extended the idea of BDT (BDD) decision trees to continuous and integer domains. This way we were allowed to extend the standard BOA algorithm to Mixed Bayesian Optimization Algorithm (mBOA). Our mBOA is the only one EDA which is able to solve problems with mixed real-discrete parameters (alleles) without conversion to binary representation.
The estimated probability model encodes the probability of whole chromosome $X$ as the product of local conditional probabilities. In mBOA the probability model is composed of $n$ decision trees. An ordering permutation of variables $(o_0, o_1, \dots, o_{n-1})$ exists such that the variables $\{X_{o_0}, X_{o_1}, \dots, X_{o_{i-1}}\}$ can serve as splits in the binary decision tree of target variable $X_{o_i}$. Each leaf determines $p(X_{o_i})$ among the individuals fulfilling the split conditions on the path from the root.

As a result of probabilistic model construction we obtain a set of decision trees, one tree for each variable. This set of trees is used for generation of the offspring (new population) during the Probabilistic Logic Sampling (PLS). During this PLS process the variables of each offspring are generated by traversing decision trees in $(o_0, o_1, \ldots, o_{n-1})$ order.

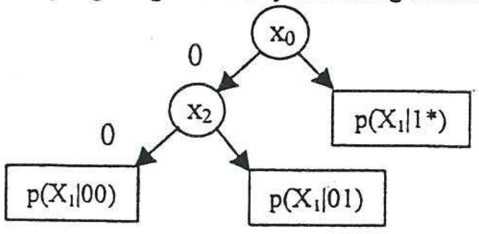


Fig. 5 An example of binary decision tree for the determination of $X_1$

In the example in Fig. 5 the concrete value of variable $X_1$ is generated according to concrete values of $X_0$ and $X_2$ using $p(X_1)$ value from corresponding leaf of decision tree.

### 3.1 Parallel construction of BDD

The probabilistic model construction is the most time consuming task in mBOA. We are currently working on the distributed mBOA using Message Passing Interface (MPI). The goal is to utilize more processors when searching for a good model. Our consolation is that the BD metric is separable and can be written as a product of $n$ factors, where $i$-th factor expresses the quality of decision tree for variable $X_i$. It is possible to use up to $n$ processors, each processor has its own local copy of parent population and it builds tree for different variable. The addition of splits to the trees is parallel, so we need an additional mechanism to keep the mutual dependencies acyclic. In [8] we proposed the concept of restricted set of parents in Bayesian network. We are going to implement this concept for BDD in mBOA. In each generation, variables will be ordered in advance, according to a random permutation vector $(o_0, o_1, .., o_{n-1})$. Each decision tree of variable $X_{oi}$ may contain only such parental splits $X_{oj}$ having $i < j$. This approach ensures linear scalability because no communication overhead is required.
The following algorithm specifies the parallel construction of BDTs and their sampling:

```
Gererate random permutation vector $(o_0, o_1, .., o_{n-1})$;
For $i := 0$ to problem_size–1 do in parallel
begin
  Model estimation:  Build BDT for target variable $X_{oi}$ (in each processor independently);
  Model sampling:    Receive from other processors values of predecessor variables $X_{o0}, X_{o1}, \ldots, X_{oi-1}$;
                     Generate $X_{oi}$ according to BDT and received variables;
                     Broadcast $X_{oi}$;
end
```

The time for BDT construction increases with $i$ (as more variables are allowed to be used as splits). In addition, scalable methods for overlapping the communication latency during generation, evaluation and broadcasting of new population among the processes will be implemented using the farmer-workers architecture. For example the communication latency during receiving $X_{o0}, X_{o1}, \ldots, X_{oi-1}$ from the other processors can be overlapped by starting the next construction of BDT from the queue of tasks.

## 4 Decomposition of large digital circuits

Decomposition/partitioning a large circuit into k subcircuits/modules is frequently solved problem that has several applications in VLSI circuit design ranging from circuit layout to logic minimization, simulation and testing. The $k$-way partitioning can be defined as follows: Let us assume a circuits $(E,S)$ with a set of circuit elements $E=\{e_1, \ldots, e_n\}$ that are connected by the set of nets $S=\{s_1, \ldots, s_m\}$, where a net is a subset of elements to be interconnected. The goal is to find such a partition of the set of elements into $k$ modules $\{M_1, M_2, \ldots, M_k\}$ so as to minimize the amount of interaction among modules under the size constraint

$$(1-\beta)\frac{|E|}{k} \le M_i \le (1+\beta)\frac{|E|}{k}, \quad i = 1,2,\ldots,k \tag{3}$$

$\beta$ is user-specified parameter to express the measure of the size imbalance of individual modules in equi-partitioning process.

To minimize the number of interactions among modules various criteria/cost functions have been used depending on the specific application context in which the partitioning is performed. Three main cost functions can be used:

- net-cut value *Ce* (cut-value, cut-size) – number of external nets
- wire-cut $W_e$ - number of external wires
- pin-count $P_c$ – number of external pins

A relation among these cost criteria exists: $P_c = C_e + W_e$. In case of two point nets it holds: $C_e = W_e$ and $P_c = 2C_e$.

The choice of a proper criterion is provided by the application context (average delay of net, via-count, test generation etc.) In Fig. 6 there are two cases of *5*-way partitioning with constant pin-cut value $P_c=15$. In the first case cut-value is very small, $C_e = 3$ but $W_e=12$ unlike the second case with $C_e = 7$ and $W_e=8$.

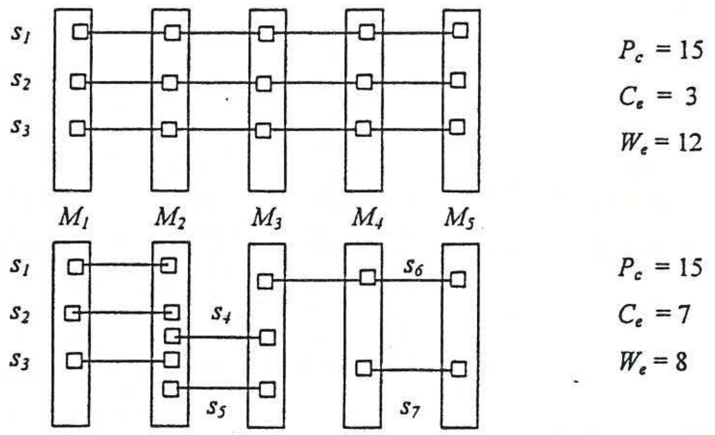


Fig. 6 Comparison of two cases of *5*-way partitioning in the layout context.

In the layout context, $W_e$ represents the number of printed wires so the first partitioning is more adequate.
In the context of the net average delay $D_a = W_e/C_e$ we get for the first partitioning $D_a=12/3=4$ and for the next one $D_a = 8/7=1,14$ so the first partition has to be preferred too.

### 4.1 Ratio-cut partitioning

To realize the multi-way partitioning the recursive minimum-cut bisection is mostly used producing modules with the evenly size or with a small size imbalance according to constraint (3). But this approach in common is not capable to identify a natural cluster in logic design which is crucial for the successful hierarchical design of the large digital circuits. Therefore a new approach was developed in [9] and [10] based on the ratio-cut metric. It is a variant of minimum–cut metric where the constraint (3) has been moved into the cost function. The ratio-cut cost function is given by

$$Rc= C/[size(M_1)*size(M_2)*\dots*size(M_k)] \tag{4}$$

where $C$ represents one of the three cost functions $C_e$, $W_e$ and $P_c$.
We have compared the standard recursive minimum-cut bisection technique (with zero imbalance of module size) with the recursive ratio-metric for each of the three cost functions.

## 5 Experimental results

### 5.1 Test benchmarks

The complexity of the partitioning problem is determined by the type and complexity of the instances – benchmark graphs. We used two benchmarks consisting of two real circuits (random logic) from benchmark package [11] with following parameters: circuit IC_67 consists of 67 elements and 138 nets, the IC_116 consists of 116 elements and 329 nets.

## 5.2 Summary of experimental results

We have arranged *8*-way partitioning experiments (ten runs for each one) for three type of cost criteria (see Fig. 7-9):

- net-cut $C_e$ (cut-value, cut-size) – number of external nets
- wire-cut $W_e$ - number of external wires
- pin-count $P_c$ – number of external pins

We have implemented and compared two algorithms – AMB based on standard recursive minimum-cut bisection technique (with zero size imbalance) and ARC based on the recursive minimum ratio-cut partitioning.

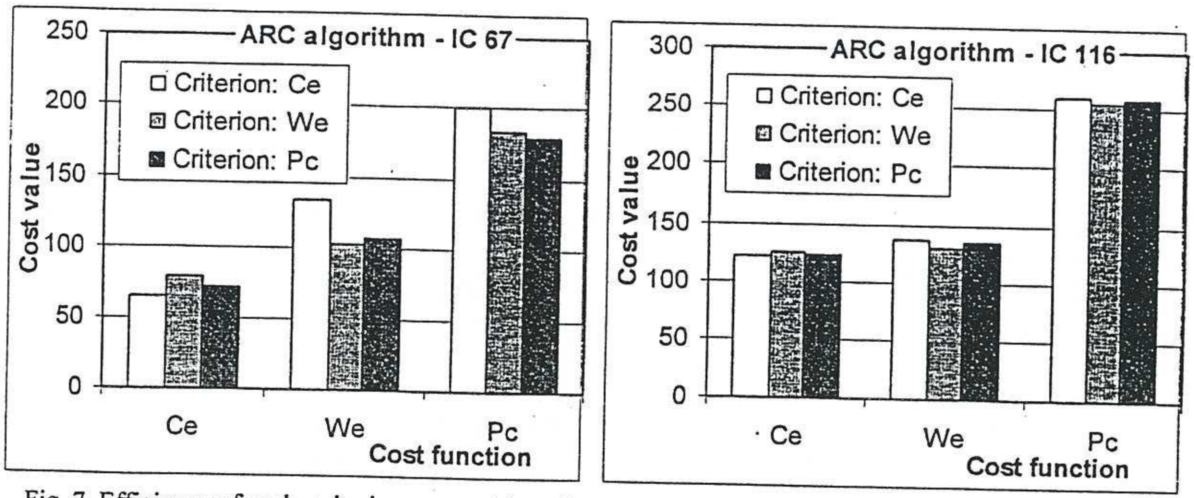


Fig. 7 Efficiency of each criteria expressed by values of other cost functions: a) for IC 67, b) for IC 116

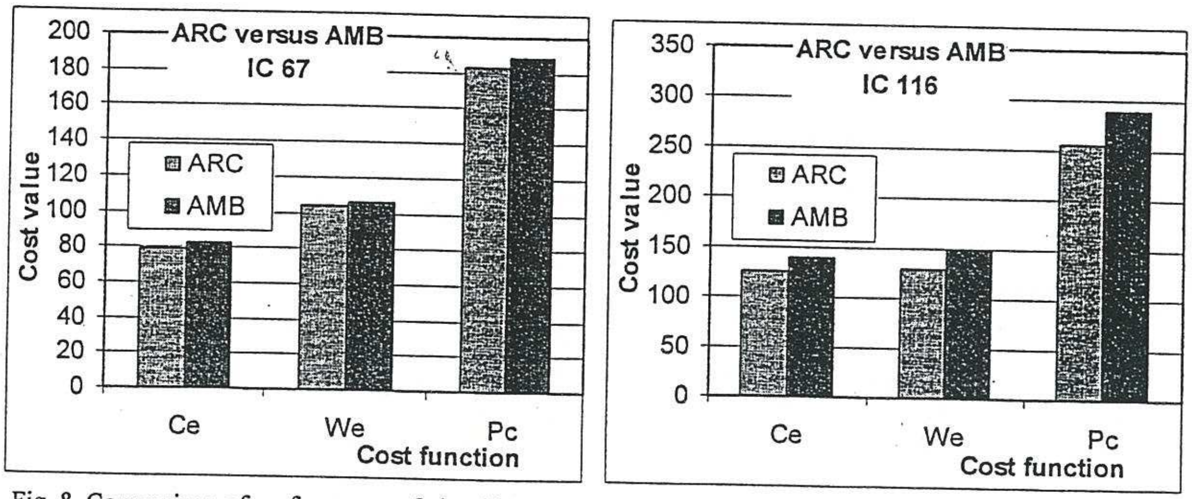


Fig. 8 Comparison of performance of algorithm ARC and AMB for $W_e$ criterion: a) for IC 67, b) for IC 116

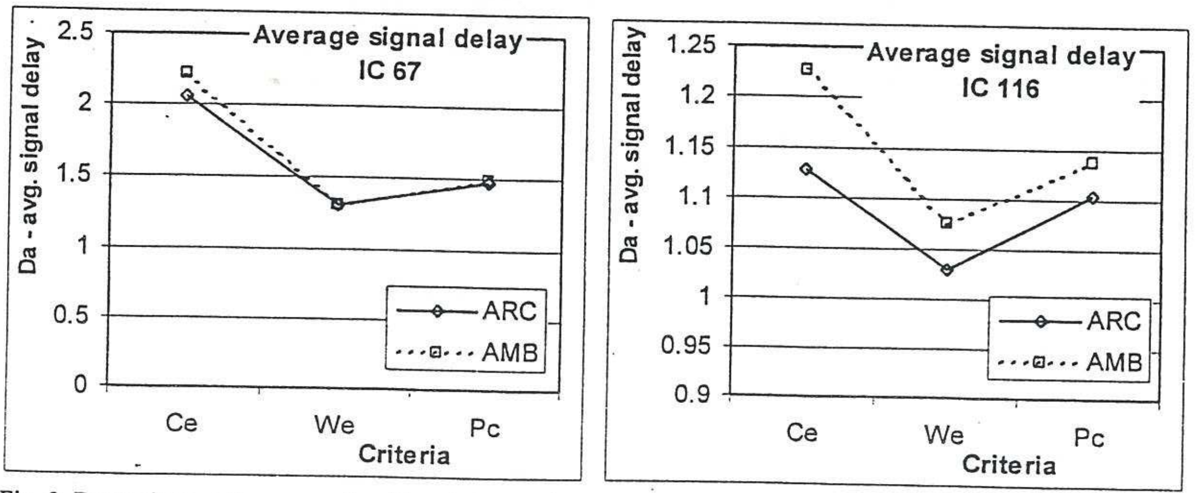


Fig. 9 Dependence of average signal/net delay on the type of optimization criteria: a) for IC 67, b) for IC 116

## 6 Conclusions

We have developed and implemented a new two phase algorithm for BDD construction which remôves the problem of variable ordering. In the first phase the minimized BDT was found using Bayes-Dirichlet metric. In the second phase fast heuristics was used based on the reduction of decision nodes with equal ancestor nodes and sharing all equivalent sub-graphs. We have tested our approach on the well known benchmark representing multiplexor- Boolean function $F_m$ up to 20 variables - the known global optimum is found in all cases. But it is desirable to test the BDD constructor on the larger number of benchmarks. We also have implemented a circuit decomposer based on BDD mBOA algorithm for multi-way partitioning of digital circuits. Its main advantage against the mostly used move-based heuristic methods lies in the ability to discover and determine the amount of epistasis in a given problem instance and to find the optimal solution. The Bayesian statistics and BDD diagrams used in probabilistic model cause its high performance. In Fig. 7 the efficiency of each criterion is shown. It is evident that criteria $W_e$ and $P_c$ performed quite well and can be used alternatively. From Fig. 8 it follows that the recursive ratio-cut algorithm ARC produces better results than the recursive min-cut bisection algorithm AMB. The difference is not too high, it is caused by the nature of applied benchmarks that probably do not include strong clusters. In Fig. 9 the average delay of nets $D_a$ is represented as a function of optimization criterion – it is evident that the criterion $C_e$ provides the best value of $D_a$.
The future activity will be focused on the sophisticated ARC algorithm including Rent's rule and the implementation of parallelization of the BDD mBOA algorithm. Another task to be solved is the deeper exploitation of the concept of OBDD construction using the Bayes-Dirichlets metric.

## Acknowledgement

*This research has been carried out under the financial support of the Research intention no. CEZ: J22/98: 262200012- "Research in information and control systems" (Ministry of Education, CZ) and the research grant GA 102/02/0503 "Parallel system performance prediction and tuning" (Grant Agency of Czech Republic).*

## References

[1] Dvořák V.: Bounds on Size of Decision Diagram. JUCS, Vol.3, 1997, pp. 2-22.

[2] Sasao T., Fujita, M., editors: Representations of Discrete Functions. Kluwer Academic Publisher, London 1996.

[3] Drechsler R.: Evolutionary algorithms for VLSI CAD. Kluwer Academic Publishers, London 1998, ISBN 0-7923-8168-8, pp. 1-183.

[4] Heckerman, D., Geiger, D., & Chickering, M. (1994). Learning Bayesian Networks: The combination of Knowledge and Statistical Data (Technical Report MSR-TR-94-09), Redmont, WA: Microsoft Research, 1995, pp. 1-53.

[5] Pelikan, M.: A Simple Implementation of Bayesian Optimization Algorithm in C++ (Version 1.0). Illigal Report 99011, February 1999, pp. 1-16.

[6] Pelikan, M., Goldberg, D. E., & Lobo, F.: A Survey of Optimization by Building and Using Probabilistic Model, Illigal Report 99018, September 1999, pp. 1-12.

[7] Pelikan, M., Goldberg, E., Sastry, K.: Bayesian Optimization Algorithm, Decision Graphs, and Occams Razor. Illigal Report No. 2000020, May 2000, pp.1-24.

[8] Očenášek, J., Schwarz, J.: The Distributed Bayesian Optimization Algorithm, Proceeedings of the Eurogen 2001 - Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems, The National Technical University of Athens, Greece, 19-21 September 2001, in print.

[9] Wei, Y.- C., Cheng, C.-K: Ratio Cut Partitioning for Hierarchical design, IEEE Trans. Computer Aided Design Integrated Circuit & System, Vol.10 (No.7), July 1991, pp. 911-921.

[10] Stroobandt, D.: Pin Count Prediction in Ratio Cut Partitioning for VLSI and ULSI. In M. A. Bayoumi, Editor, Proceedings of the IEEE International Symposium on Circuits and Systems, May 1999, Pages VI/262-VI/265.

[11] A Benchmark Set, University of California, Los Angeles, VLSI CAD Laboratory, http://vlsicad.cs.ucla.edu/~cheese/benchmarks.html.