GENEL ÖRTÜ PROBLEMİNİ ÇÖZEN BİR BİLGİSAYAR PROGRAMININ GELİŞTİRİLMESİ VE PROGRAMIN İKİ ÖZEL HALE UYGULANMASI

Orhan UÇAR
İ.T.Ü. F.B.E. 80626, Maslak, İstanbul
orhan_ucar@yahoo.com

Ahmet DERVİŞOĞLU
Yeditepe Ü EE Müh. Bölümü, İstanbul
adervisoglu@yeditepe.edu.tr

Anahtar sözcükler:örtü problemi, lojik tasarım, algoritma

ÖZET

Örtü problemi (covering problem) evrensel bir problemdir. Ardışıl devre tasarımında durum indirgeme ve durum kodlama, kombinezonsal devre tasarımında Boole fonksiyonlarının indirgenmesi başta olmak üzere pek çok alanda bu problemle karşılaşılmaktadır. Bu nedenlerle bu problemin kısa sürede az bellek kullanılarak çözümü önem kazanmaktadır.

Bu çalışmada geliştirilmiş olan yöntem ve bu yönteme dayalı bilgisayar programı (UCPS) ile, tamamen keyfi olarak oluşturulan 1000x1000 mertebesindeki bir minimal örtü roblemi saniyeler mertebesinde bir sürede çözülebilmektedir. Program, Boole fonksiyonlarının VE-VEYA devreleri ile minimal gerçekleştirilmesine ilişkin çeşitli bençmarklarda denenmiş ve çözümler kısa sürede belirlenmiştir. Geliştirilen program ile elde edilen sonuçlar, University of California Berkeley' de geliştirilen ESPRESSO programı ile elde edilen sonuçlarla Tablo 1'de karşılaştırılmıştır. UCPS, kısmen belirli ardışıl makinelerde durum indirgeme problemine de uygulanmış ve bençmarklarda elde edilen sonuçlar, diğer programlarla Tablo 2'de karşılaştırılmıştır.

1. GİRİŞ

Örtü problemi, bir Boole fonksiyonunun indirgenmesi ise, satırlar asal bileşenlere, sütunlar da mintermlere karşı düşer. Ardışıl makinalarda durum indirgeme halinde, satırlar uyumlu durumlara ilişkin bloklara, sütunlar ise durumlara karşı düşer. Matristeki satır ve sütun sayısının büyük olması halinde minimal bir örtüyü makul bir makine süresinde belirlemek oldukça zordur.

Bir boyutlu örtü problemi (unate covering problem) bir matris ile ifade edilebilir. $C[i,j]_{mxn}$ örtü matrisinin her satırı bir örten nesneye, her sütunu da bir örtülen elemana karşı düşer. Matrisin i. satırı B_i nesnesine, j. sütunu da v_j elemanına karşı düşsün. Matrisin c_{ij} elemanı 1 ise B_i nesnesi v_j elemanını örtüyor, c_{ij} elemanı 0 ise B_i nesnesi v_j elemanını örtmüyor demektir. Minimal örtü problemi şöyle tanımlanır: Minimum sayıda nesneyi belirle öyle ki tüm elemanlar örtülsün. Aşağıda 8 nesneli ve 23 elemanlı bir örtü problemine ilişkin örtü matrisi verilmiştir.

B_1 10000100100101010001010
B_2 01101101010000110110001
B_3 10000100100110001000010
B_4 00101010111010011001101
B_5 01000111001001010110001
B_6 11010100100111010100010
B_7 00100010101010101000101
B_8 01010101011000010101000

Genel örtü problemi bir matrisle ifade edilebileceği gibi, Petrick fonksiyonu adı verilen ve toplamlar-çarpımı biçimindeki bir Boole fonksiyonu ile de ifade edilebilir. Matrisle ifade etmede her satır bir nesneye karşı düşmekte iken, Petrick fonksiyonunda her çarpan bir elemana karşı düşmektedir ve o elemanı örten nesnelerin toplamına eşittir. Kolayca görülebilir ki minimum sayıda değişkene 1 değeri verildiğinde fonksiyonun değeri 1'e eşit oluyorsa bu minimum sayıdaki nesne bir minimal örtü oluşturur. Böyle bir değişken kümesi belirmenin bir yolu, Petrick fonksiyonunu çarpımlar-toplamı biçimine dönüştürmektir; bu açınımdaki minimum sayıda çarpan içeren her terim bir minimal örtü verir.

Bu çalışmada, toplamlar çarpımı biçimindeki bir Petrick fonksiyonunu, çarpımlar-toplamı biçimine dönüştüren, dolayısıyla tüm minimal örtüleri veren bir bilgisayar programı da verilmiştir. Bu program, büyük örtü problemlerinin çözümü için elverişli değildir; fakat eğitim amaçlı örtü problemleri için yeteri kadar hızlıdır ve dallanma yönteminin anlaşılmasını kolaylaştırmaktadır. Örneğin P=(A+B+C)(D+E+F)(A+H+I)(C+D+I) fonksiyonu 81 dal içermektedir ve bu örtü probleminin minimal örtüsü, Petrick fonksiyonu açıldığında AD olarak elde edilir.

2. GELİŞTİRİLEN YÖNTEM

Minimale yakın bir örtüyü kısa sürede bulabilmek için şöyle bir yol izlenebilir: En çok sayıda elemanı örten (örtü matrisi C'nin en çok 1 içeren satırına karşı düşen) bir nesne seç. Seçilen nesne B_1 olsun. B_1 nesnesine ilişkin satırı ve bu satırın örttüğü sütunları örtü matrisi C'den sil. Elde edilen matriste yine aynı kritere göre B_2 satırını seç. B_2 satırını ve örttüğü sütunları örtü matrisi C'den sil. Bu şekilde C örtü matrisinde satır kalmayıncaya kadar devam et. Seçilen nesnelerin kümesi $B=\{B_1, B_2,...,B_k\}$ olsun. Bu kümede $B_i \leq B_j$ olan nesneler var ise B_i'yi kümeden at.

Bu şekilde elde edilen B kümesi minimale yakın bir örtü verir. Bu yönteme EBAY(En Büyük Adımlar Yöntemi) denilecektir.

Bu çalışmada EBAY yöntemi iyileştirilerek kısa sürede minimale yakın bir örtü elde edilmekte, elde edilen sonuç minimal çözüm için bir üst sınır (UB) alınarak dallanma yöntemiyle minimal örtü (MC) elde edilmektedir. Eğer UB değeri MC' deki nesne sayısına yakın ise, MC kısa sürede elde edilmektedir. Öte yandan, MC'deki nesne sayısı m_o olsun; MC için bir alt sınır (LB) belirlenebilir ve alt sınırın m_o'a yakın olması durumunda çözüme hızla ulaşılabilmektedir.

Bu çalışmada geliştirilen iyileştirilmiş EBAY yöntemi ve iyileştirilmiş dallanma yöntemi ile önce m_o'a yakın üst sınırı belirlenmekte sonra da bundan yararlanılarak minimal örtü elde elde edilmektedir.

Yukarıda verilen 8 nesneli ve 23 elemanlı örtü problemine ilişkin minimal örtü MC={B_2, B_4, B_6} olarak elde edilir. Bu problem için m_o=3 olmakta ve aynı çözüm literatürde verilen EBAY(Greedy, [3]) yöntemi ile de elde edilmektedir. Ancak EBAY yöntemi nesne ve eleman sayısı çok az olabilen örtü problemlerinde dahi minimal örtüyü elde edemeyebilir. Bu durum büyük örtü problemlerinin çözülmesini gerektiren bençmarklarda daha belirgin hale gelir. EBAY yöntemi bazı bençmarklarda minimalden oldukça uzak örtüler elde etmektedir.

Aşağıda EBAY yönteminin minimal örtüyü elde edemediği bir örtü problemi ve iyileştirilmiş EBAY yöntemi ile minimal örtünün elde edilme adımları verilmiştir.

Tek koşullu bir örtü problemi matris şeklinde ifade edildiği gibi bir küme ile de ifade edilebilir. Örneğin {123, 134, 235, 56, 67, 27} kümesi 6 nesneli ve 7 elemanlı bir örtü problemini ifade etmektedir. Bu örtü probleminde EBAY yöntemi kullanılarak sırası ile 123,56,134,67 nesneleri örtüye dahil edilmek üzere seçilir. Bu yöntem ile elde edilen minimale yakın örtüde 4 nesne bulunmakta ve birbirini içeren nesneler bulunmamaktadır. İyileştirilmiş EBAY yönteminde, EBAY yöntemi ile elde edilmiş olan her nesnenin örtüye girme nedeni yani o nesne tarafından içerilen fakat diğer nesneler tarafından içerilmeyen elemanlar hesaplanır. Minimale yakın örtüdeki iki nesneye ait örtüye girme nedenleri eğer bir nesne tarafından içeriliyor ise, bu iki nesne atılıp yerlerine yeni nesne minimal örtüye dahil edildiğinde minimale yakın örtüdeki nesne sayısı 1 azaltılmış olur. Örneğin yukarıdaki çözümde 123 ve 56 nesneleri minimale yakın örtüden atılıp yerlerine 235 nesnesi eklendiğinde minimale yakın olan 4 nesneli çözümdeki nesne sayısı 1 azaltılır ve minimal örtü elde edilir. Aynı şekilde EBAY yöntemi ile elde edilen minimale yakın örtüden 123 ve 67 nesneleri atılıp yerlerine 27 nesnesi eklendiğinde yine 3 nesneli minimal örtü elde edilir.

Geliştirilmiş olan yöntem ile eğer k tane nesnenin minimal örtüye girme nedenleri bir nesne tarafından içeriliyor ise minimale yakın örtüdeki nesne sayısı k-1 azaltılır.

EBAY yöntemi, max1024 isimli bençmarkta, 298 nesneli minimale yakın bir örtü elde etmektedir. Minimal örtüde ise 259 nesne bulunmaktadır. Yani EBAY yöntemi bu bençmarkta 39 nesne fazlası ile (%15) minimale yakın örtüyü elde edebilmektedir. İyileştirilmiş EBAY yönteminde ise 298'li çözüm iyileştirilerek 270 nesneli çözüm elde edilmiştir. Bu durumda iyileştirilmiş EBAY yöntemi ile %11'lik bir iyileştirme sağlanmış, minimal örtüden %4 fazla nesneli çözüm elde edilmiştir. max512 isimli bençmarkta ise EBAY yöntemi ile 153 nesneli minimale yakın bir örtü elde edilmektedir. Minimal örtüde ise 133 nesne bulunmaktadır. Yani EBAY yöntemi bu bençmarkta 20 nesne fazlası ile (%15) minimale yakın örtüyü elde edebilmiştir. İyileştirilmiş EBAY yönteminde ise 153'lü çözüm iyileştirilerek 137 nesneli çözüm elde edilmiştir. Bu durumda iyileştirilmiş EBAY yöntemi ile %12'lik bir iyileştirme sağlanmış, minimal örtüden %3 fazla nesneli çözüm elde edilmiştir. Diğer pek çok bençmarkta ise (5xp1, sqr6, m2, gibi) iyileştirilmiş EBAY yöntemi ile tam bir iyileştirme sağlanmış ve minimal örtü elde edilmiştir. Minimal örtünün elde edilemediği bençmarklarda iyileştirilmiş dallanma yöntemi ile minimal örtü elde edilmiştir.

Eğitim amaçlı uygulamalarda, örtü probleminin birden çok çözümü varsa, tümünün elde edilmesi istenmekte, ayrıca programın görsel ve kullanıcı dostu olması arzu edilmektedir. Bu özelliklerin gerçekleştirilmesi halinde program büyük bençmarklarda yavaşlamaktadır. Eğitim amaçlı problemler büyük olmadığından, bir örtü probleminin tüm çözümleri saniyeler mertebesindeki bir makine süresinde belirlenebilmektedir. Bu nedenle geliştirilen yönteme ilişkin bilgisayar programının iki versiyonu hazırlanmıştır.

3. SONUÇ

Bu çalışmada geliştirilmiş olan iyileştirilmiş EBAY, iyileştirilmiş dallanma yöntemleri ve bu yöntemlere dayalı bilgisayar programı ile tamamen keyfi olarak oluşturulan 1000x1000 mertebesindeki bir minimal örtü roblemi saniyeler mertebesinde bir sürede çözülebilmektedir. Program, Boole fonksiyonlarının VE-VEYA devreleri ile minimal gerçekleştirilmesine ilişkin çeşitli bençmarklarda denenmiş ve çözümler kısa sürede belirlenmiştir. Geliştirilen program ile elde edilen sonuçlar, University of California Berkeley'de geliştirilen ESPRESSO programı ile elde edilen sonuçlarla Tablo 1'de karşılaştırılmıştır. Tablodan görüldüğü gibi UCPS algoritmasını kullanan MORP programı pek çok bençmarkta optimal, geriye kalan bençmarklarda ise optimale oldukça yakın sonuç

vermiştir. Oysa ESPRESSO programı genelde optimal çözümü değil, optimale yakın bir çözüm vermektedir.

UCPS, kısmen belirli ardışıl makinelerde durum indirgeme problemine de uygulanarak, UCPS algoritmasını kullanan SRC(State Reduction and Covering) programı geliştirilmiştir. SRC ile elde edilen sonuçlar, Puri[2] ve Rho[1] tarafından geliştirilmiş olan programların verdiği sonuçlarla Tablo 2'de karşılaştırılmıştır. Bu tablodan da görüldüğü gibi SRC programı ile diğer yöntemlere göre çok hızlı indirgeme yapılabilmektedir. Ayrıca diğer yöntemlerin indirgeyemediği bençmarklar da çok kısa sürede indirgenmiştir.

UCPS algoritmalarında kullanılan teknikler durum kodlama problemine de uygulanmıştır. Tracey minimum geçiş zamanlı durum kodlama algoritması büyük örtü problemlerinin çözümünü gerektirmektedir. Tracey kodlama yöntemini temel alan yeni bir durum kodlama yöntemi, UCPS algoritması kullanılarak geliştirilmiştir. Bu konudaki çalışmalar devam etmektedir. Geliştirilen program literatürdeki bençmarklarda denenmiş, kritik yarışsız kodlama işlemi hızlı bir şekilde gerçekleştirilmiştir.

KAYNAKLAR

[1] Rho J.K., Hachtel G.D., Somenzi F. and Jacoby R.M., "Exact and Heuristic Algorithms for the Minimization of Incompletely Specified State Machines", IEEE Transactions on CAD of Integrated Circuits and Systems, v.13, n.2, February 1994, pp.167-176.

[2] Puri R. and Gu J., "An Efficient Algorithm to Search for Minimal Closed Covers in Sequential Machines", IEEE Transactions on CAD of Integrated Circuits and Systems, v.12, n.6, June 1993, pp.737-745.

[3] Brayton R.K., G.D. Hachtel, C.T. McMullen, A.L. Sangiovanni-Vincentelli, Logic Minimization Algorithms for VLSI Synthesis (Kluwer Academic Publishers, Dordrecht, 1984).

Tablo 1 MORP Programı test sonuçları

No	Bençmark	nxm	NM	NPI	MC	MORP 9.1 (optimal)	MORP 9.1 (greedy)	Espresso (Heuristic)	Espresso (Optimal)
1	5xp1	7x10	784	390	63	63 / 4.7 s.	63 / 1.6 s.	65 / 0 s.	63 / 1 s.
2	9sym	9x1	696	1680	84	84 / 31 s.	85 / 7.4 s.	87 / 0 s.	-
3	addm4	9x8	1310	1122	189	-	190 / 2.3 s.	200 / 1 s.	-
6	apex4	9x19	2970	2336	428	428 / 1095 s.	430 / 18 s.	436 / 9 s.	-
7	apla	10x12	157	201	25	25 / 94 s.	25 / 94 s.	25 / 0 s.	25 / 1 s.
10	ex5	8x63	7620	2532	65	-	72,73/145,68	74 / 2 s.	-
11	f51m	8x8	1024	561	76	76 / 1.26 s.	76 / 1.26 s.	77 / 1 s.	76 / 2 s.
12	lin.rom	7x36	2306	1087	128	128 / 446 s.	145 / 18 s.	128 / 6 s.	-
13	luc	8x27	2246	190	26	26 / 1.4 s.	26 / 1.4 s.	26 / 1 s.	26 / 1 s.
14	m1	6x12	218	59	19	19 / 0 s.	19 / 0 s.	19 / 0 s.	19 / 1 s.
15	m2	8x16	831	243	47	47 / 0.5 s.	47 / 0.3 s.	47 / 0 s.	47 / 1 s.
16	m3	8x16	1105	344	62	62 / 5.7 s.	65 / 0.6 s.	65 / 1 s.	62 / s.
17	m4	8x16	2134	670	101	104 / 1032 s.	107 / 3.2 s.	107 / 1 s.	Error
18	max46	9x1	62	49	46	46 / 0 s.	46 / 0 s.	46 / 0 s.	46 / 0 s.
19	max128	7x24	1616	469	78	78* / 16.5 s.	85 / 1.37 s.	82 / 1 s.	78 / 2 s.
20	max512	9x6	1616	535	133	135 / 2123 s.	136 / 4s.	142 / 2 s.	133 / 7 s.
21	max1024	10x6	3232	1278	259	269 / 410 s.	270 / 60 s.	274 / 7 s.	-
22	mlp4	8x8	678	606	121	121* / 20 s.	125 / 0.71 s.	128 / 1 s.	Error
23	newtpla1	10x2	14	6	4	4 / 0 s.	4 / 0 s.	4 / 0 s.	4 / 0 s.
24	newtpla2	10x4	608	23	9	9 / 3 s.	9 / 0 s.	9 / 0 s.	9 / 0 s.
25	pope	6x48	1614	593	59	59 / 32 s.	62 / 1.6 s.	62 / 1 s.	59 / 5 s.
26	prom1	9x40	8306	9326	472	472 / 250 s.	472 / 250 s.	472 / 13 s.	-
27	rd53	5x3	48	51	31	31 / 0.1 s.	31 / 0.1 s.	31 / 0 s.	31 / 0 s.
28	rd73	7x3	428	211	127	127 / 0.8 s.	127 / 0.8 s.	127 / 1 s.	127 / 1 s.
29	rd84	8x4	411	633	255	255 / 0.33 s.	255 / 0.33 s.	255 / 0 s.	255 / 1 s.
33	sym10	10x1	837	3150	210	210 / 66 s.	210 / 43 s.	210 / 2 s.	-
35	z5xp1	7x10	576	390	63	63 / 4.78 s.	63 / 0.3 s.	63 / 1 s.	63 / 1 s.
36	z9sym	9x1	420	1680	84	86 / 2093 s.	88 / 5 s.	85 / 0 s.	-

n: giriş sayısı, m:fonksiyon sayısı, NM: Minterm sayısı, NPI: Asal Bileşen sayısı,
MC: Minimal örtüdeki nesne sayısı

Tablo 2 SRC programı test sonuçları

					SRC		Rho [1]		Puri [2]	
No	**Bençmark**	**n**	**i**	**o**	**n_r**	**süre(s.)**	**n_r**	**süre(s.)**	**n_r**	**süre(s.)**
1	bbara	10	4	2	7	0.00	7	0.15	7	0.00
2	bbsse	16	7	7	13	0.00	13	0.23	13	0.02
3	bbtas	6	2	2	6	0.00	6	0.05	6	0.00
4	beecount	7	3	4	4	0.00	4	0.07	4	0.01
5	cse	16	7	7	16	0.00	-	-	16	0.01
6	donfile	24	2	1	1	0.00	-	-	1	0.00
7	ex1	20	9	19	18	0.00	18	0.37	18	0.01
8	ex2	19	2	2	5	10	-	time out	-	-
9	ex3	10	2	2	4	0.00	not found	-	4	2.51
10	ex4	14	6	9	14	0.00	-	-	14	0.01
11	ex5	9	2	2	3	0.00	3	5.17	3	0.23
12	ex6	8	5	8	8	0.00	-	-	8	0.01
13	ex7	10	2	2	3	0.00	3	12.03	3	0.26
14	keyb	19	7	2	19	0.00	-	-	19	0.02
15	lion	4	2	1	4	0.00	-	-	4	0.01
16	lion9	9	2	1	4	0.00	4	0.00	4	0.01
17	mark1	15	5	16	12	0.00	not found	-	12	0.02
18	modulo12	12	1	1	1	0.00	-	-	1	0.01
19	opus	10	5	6	9	0.00	9	0.05	9	0.01
20	planet	48	7	19	48	0.00	-	-	48	0.11
21	planet1	48	7	19	48	0.00	-	-	48	0.11
22	s1	20	8	6	20	0.00	-	-	20	0.02
23	s1a	20	8	6	1	0.00	-	-	1	0.01
24	s8	5	4	1	1	0.00	-	-	1	0.01
25	sand	32	11	9	32	0.00	-	-	32	0.02
26	scf	121	27	56	97	0.20	97	6.77	97	0.99
27	shiftreg	8	1	1	8	0.00	-	-	8	0.02
28	sse	16	7	7	13	0.00	13	0.20	13	0.02
29	styr	30	9	10	30	0.00	-	-	30	0.04
30	tav	4	4	4	4	0.00	-	-	4	0.01
31	tbk	32	6	3	16	0.71	16	21.60	16	1.64
32	train4	4	2	1	4	0.00	-	-	4	0.01
33	train11	11	2	1	4	0.00	4	0.38	4	0.01

n: durum sayısı i: giriş sayısı o: çıkış sayısı n_r: indirgenen makinenin durum sayısı