SEZGİSEL YÖNTEMLER

2019

Fiduccia-Mattheyses (FM) Algoritması
Grafik bölümleme

ÖMER M. OĞUZOĞLU

Pamukkale Üniversitesi
Mühendislik Fakültesi | Bilgisayar Mühendisliği | 2018-2019
omer1991mustafa@gmail.com

Fiduccia-Mattheyses (FM)

Grafik bölümleme algoritması

PAMUKKALE ÜNİVERSİTESİ

SEZGİSEL YÖNTEMLER VE UYGULAMALARI ARA SINAV PROJE RAPORU – BAHAR 2019

ÖMER MUSTAFA OĞUZOĞLU
13253813

Danışman : Dr. Öğr. Ü. Kenan KARAGÜL

Pamukkale Üniversitesi – Bilgisayar Mühendisliği 2018 Bahar Dönemi- IENG 479 Sezgisel Yöntemler ve Uygulamaları dersine hazırlanmıştır.

İçindkiler

1.Giriş

Bilgisayar bilimlerinde sezgisel yöntemler bir problem çözme tekniğidir. Sonucun kesin doğru olduğunu önemsememekle beraber kabul edilir sonuçlar verir, ayrıca zaman açısından maksimum verimliliği hedefler.
Sezgisel algoritmalar makul bir süre içerisinde bir çözüm vereceklerini garanti ederler, fakat en iyi sonucu vereceklerini garanti etmezler.
Sezgisel algoritmalar, büyük optimizasyon problemler için kısa süre içerisinde optimuma yakın çözüm üretirler, bu algoritmalar genel olarak biyoloji tabanlı, fizik tabanlı, sürü tabanlı, sosyal tabanlı, müzik tabanlı ve kimya tabanlı olmak üzere altı farklı grupta değerlendirilmektedir. Sürü zekâsı tabanlı optimizasyon algoritmaları kuş, balık, kedi ve arı gibi canlı sürülerinin hareketlerinin incelenmesiyle geliştirilmiştir.[1]

Sezgisel optimizasyon yöntemlerine örnek olarak:

- Optikten Esinlenen Optimizasyon (Optic Inspired Optimization)(OIO)
- Genetik Algoritma (Genetic Algorithm)(GA)
- Karınca Kolonisi Optimizasyonu (Ant Colony Optimization)(ACO)
- Parçacık Sürü Optimizasyonu (Particle Swarm Optimization)(PSO)
- Yapay Arı Kolonisi (Artificial Bee Colony)(ABC)
- Benzetim Tavlama (Simulated Annealing)(SA)
- Su Döngüsü Optimizasyon Algoritması
- Krill Sürü Optimizasyon Algoritması
- Bakteri Yiyecek Arama Davranışı
- Yarasa Algoritması
- Yapay Alg Algoritması
- Virüs Koloni Arama Algoritması
- Ağaç-Tohum Algoritması(Tree-Seed Algorithm)(TSA)

2.Grafik bölümleme:

Matamatikte bir G=(V,E) grafiğini (V=köşe noktası, E=kenar) belirli özellikli küçük parçalara bölmek mümkün. Grafik bölümlemenin en önemli uygulamalarından bilimsel hesaplama, VLSI devre tasarımı ve çok işlemcili sistemlerde görev planlamasıdır. En yaygın Grafik bölümleme algoritmalarından birisi Kernighan-Lin (KL) Algoritması ve diğeri Fiduccia-Mattheyses (FM) algoritmasıdır. [2] Bu raporda KL algoritmasını özet geçtikten sonra FM algoritması detaylı şekilde incelenecek.

GENETİK ALGORİTMA

NP-Hard problemleri çözmek için kullanılır, Karmaşık çok boyutlu arama uzayında en iyinin hayatta kalması ilkesine göre bütünsel en iyi çözümü arar, Genetik algoritmaların temel ilkeleri ilk kez Michigan Üniversitesi'nde John Holland tarafından ortaya atılmıştır.

VLSI

Çok geniş ölçekli integer-rasyon (Very Large Scale Integration) binlerce transistörün tek bir yonga üzerinde birleştirilmesi ile Tümleşik Devrelerin oluşturulması işlemidir

2.1 Grafik bölümleme algoritmalarının genel kullanımı:

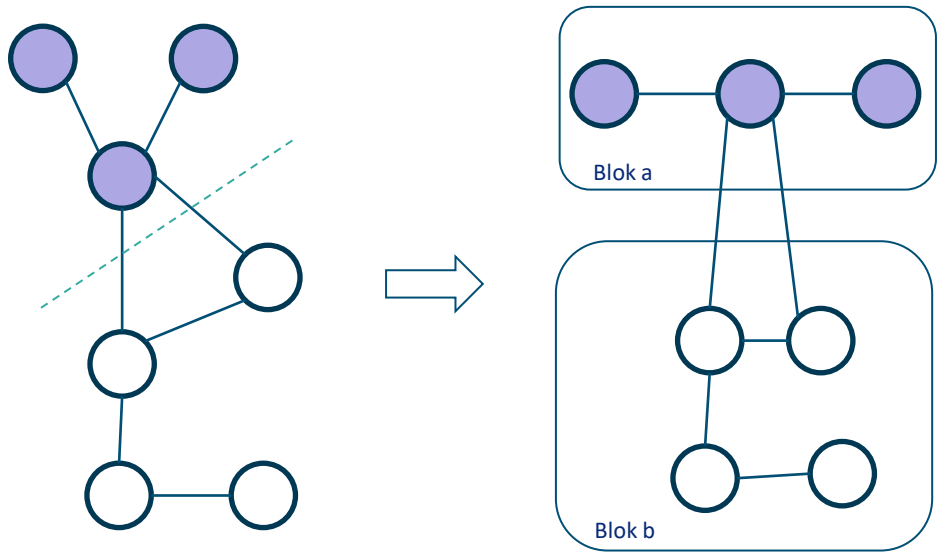
- Divide-and-conquer algoritması
- Devre düzeni ve tasarımları
- Parelel hesaplama
- Biyoinformatik

2.2 Grafik bölümleme algoritmalarının büyük grafiklerdeki kullanımı:

- internette arama
- toplulukları tanımlama
- hedefleri izleme
- spam linkleriyle mücadele
- devre bölümleme (VLSI)

2.3 Netlist ve Sistem Bölümleme

Son zamanlarda modern entegre devrelerinin tasarım karmaşıklığı eşi görülmemiş bir büyüklüğe ulaştı, Ful-Çip düzeni yapımı, FPGA tabanlı devreler ve benzeri modeller gittikçe daha da zorlaşmaktadır. Genel strateji bu tasarımları daha küçük ve basit parçalara bölümlemek, bunların her biri paralellik ve bağımsızlık derecesi ile işlenebilir. Çip tasarımı için kullanılabilen Divide-and-conquer stratejisi manuel bölümleme için kullanılır fakat bu yöntem Büyük ağlar (Netlist) için pek mümkün değil. Manuel çözüm yerine sezgisel yöntem kullanmak daha uygun olacaktır. [4]



2.4 Kernighan-Lin (KL) Algoritması:

Kernighan-Lin, köşe noktası ve kenarları olan bir grafiği ayrı alt kümelere böler, örneğin bir elektrik şebekesini grafik şeklinde temsil edebiliriz bu yüzden Kernighan-Lin algoritmasını kullanmak mümkün olacaktır. Kernighan-Lin deterministik bir algoritmadır, Dolaysıyla algoritmayı her kullandığımızda aynı sonucu elde ederiz. Aynı sonuç, aynı ağlar ve düğümler olmasa da, bölümler arasındaki geçişlerin maliyeti aynı olacaktır. [3]

3. Fiduccia-Mattheyses (FM) algoritması

Sezgisel parçalama olan bu algoritma, 1982 yılında C.M. Fiduccia ve R.M. Mattheyses tarafından paylaşıldı. Ağırlığı olan bir G(V,E) grafiği verildiğinde hedefimiz bu grafikteki tüm köşe noktalarından ayrık bölümler oluşturmak olacak, böylece amaç tüm kesilmiş ağların (nets) bölme boyutu kısıtları da dikkate alınarak ağırlık maliyetlerini en aza indirgemek olur. FM algoritması KL algoritmasına benzer bir fikri taşır, KL algoritması her seferde iki düğüm taşırken FM algoritması bir düğüm taşır.

Düğümlerin kazancı hesaplanır ve kazancı maksimum olan düğüm karşı bölgeye geçer. Eğer tüm düğmeler tek bir bölüme geçecek olursa o son işlemde denge ihlali oluşur bu yüzden o düğmeyi baz olarak seçmekten vazgeçilir ve bir diğer maksimum kazancı olan düğme (köşe noktası) seçilir ve taşınır. İşlemler aşağıda belirtilen örneklerde daha detaylı açıklanacaktır.[4]

3.1 zaman karmaşıklığı:

- Taşımak için en iyi düğmeyi bulma zamanı sabit.
- Toplam zaman karmaşıklığı O(n), n düğmelerin toplam sayısı.

3.2 KL algoritmasından farklı yanları:

- Her taşımada çift düğme yerine tek düğme taşır.
- Düğmelerin toplam sayısı çift olmak zorunda değil.

3.3 özellikleri:

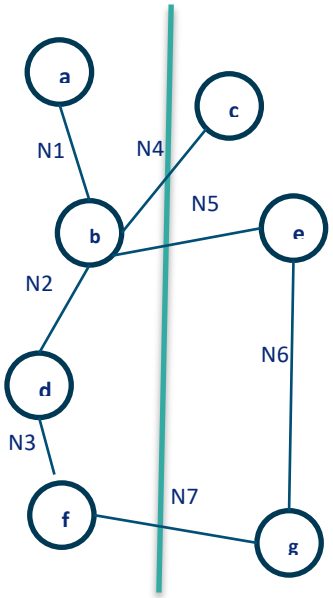
- Net-cut (ağ kesimleri) maliyetini azaltmayı hedefler.
- Ağırlığı olan grafikler de çok iyi çalışır.
- Her defasında Sadece bir köşe noktası taşınır.
- Dengesiz bölümlerle baş edebilir.
- Bölüm alanına taşınacak köşe noktası için özel bir veri yapısı kullanılır.

3.4 Aşamalar:

- Tüm hücrelerin kazancını hesapla, **$\Delta g(c) = FS(c) - TE(c)$**
- Baz hücreyi seç (Δg maksimum olduğu halde o köşe baz olur) ve bu hücreyi taşı.
- Baz hücreyi kilitle (diğer iterasyonlara dahil edilmez).
- Yeni oluşumdaki kazançları güncelle ve yeni baz hücreyi seç.
- En iyi dizilimi seç.
- Gerçekleştirilmiş taşınmaları sabitle.

3.5 Uygulama:

Örnek: kazançları hesaplama



a için: kesişim yok o halde FS(a)=0, n1 bağlı ama kesilmedi O halde TE(a)=1, Δg(a) = -1

b için: n4 ve n5 ağları kesildi o halde FS(b)=2, n1 ve n2 b ye bağlı ve kesilmedi o halde TE(b)=2, Δg(b) = 0

c için: FS(c)=1 , TE(c)=0, Δg(c) = 1

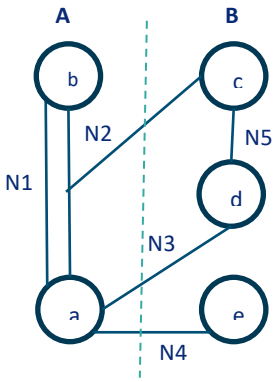
d için: FS(d)=0 , TE(d)=2, Δg(d) = -2

e için: FS(e)=1 , TE(e)=1, Δg(e) = 0

f için: FS(f)=1 , TE(f)=1, Δg(f) = 0

g için: FS(g)=1 , TE(g)=1, Δg(g) = 0

Örnek: FM algoritması (adım-adım)



0. iterasyon

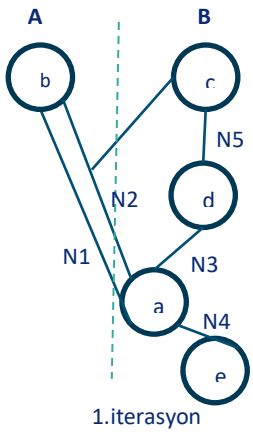
Verilenler:

1. Ağırlıklı hücreler a-e, 2. oran faktörü r= 0.3755, 3. Ağları N1-N5 ve 4. Baş bölüm (sağ taraf)

alan(a)=2, alan(b)=4, alan(c)=1, alan(d)=4, alan(e)=5

Soru: FM algoritmasının ilk geçişini gerçekleştir.

Çözüm:



1.iterasyon

Denge kriterini hesaplamak:

$$r \cdot alan(V) - alan_{max}(V) \le alan(A) \le r \cdot alan(V) + alan_{max}(V)$$

$r \cdot alan(V) - alan_{max}(V) = 0.375 \times 16 - 5 = 1$

$r \cdot alan(V) + alan_{max}(V) = 0.375 \times 16 + 5 = 11$

aralık: $1 \le alan(A) \le 11$

1. İterasyon:

her bir hücrenin kazancını hesaplamak: (0. İterasyondaki şekle göre kazanç hesabı)

a için : N3 ve N4 kesilmiş o halde: FS(a)=2, N2 kesilmemiş: TE(a)=1, $\Delta g(a) = 1$

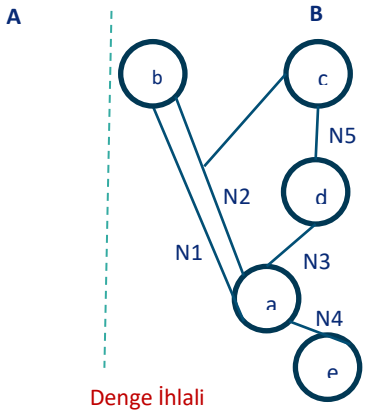
Aynı şekilde diğerleri de:

b: FS(b)=0 , TE(b)=1, $\Delta g(b) = -1$

c: FS(c)=1 , TE(c)=1, $\Delta g(c) = 0$

d: FS(d)=1 , TE(d)=1, $\Delta g(d) = 0$

e: FS(e)=1 , TE(e)=0, $\Delta g(e) = 1$



Denge İhlali

baz hücreyi seçmek:

Δg'si maksimum olan seçeriz, burada mümkün seçimler a ve e'dir.

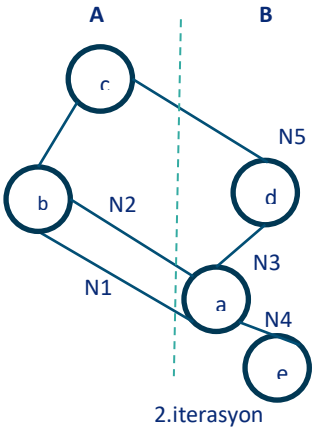
a'yı seçtikten sonraki denge kriteri: alan(A)=alan(b)=4

e'yı seçtikten sonraki denge kriteri: alan(A)= alan(a)+alan(b)+alan(e)=11

baz hücremizi a olarak seçelim:

Δg değerlerini güncelledikten sonra:

bölümler: $A_1=\{b\}$, $B_1=\{c,d,a,e\}$/ daha önce seçilmiş {a} diğer iterasyonlara dahil edilmeyecek.



2.iterasyon

2. İterasyon:

Kazanç:

(1. İterasyondaki şekle göre)

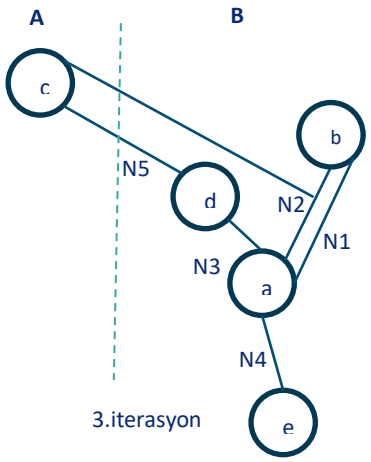
b: FS(b) = 2, TE(b) = 0, $\Delta g1(b) = 2$

c: FS(c) = 0, TE(c) = 1, $\Delta g1(c) = -1$

d: FS(d) = 0, TE(d) = 2, $\Delta g1(d) = -2$

e: FS(e) = 0, TE(e) = 1, $\Delta g1(e) = -1$

Bu kez Baz hücremiz b'dir çünkü $\Delta g2(b) = 2$ maksimum olandır ve



3.iterasyon

Alan (A)=0 **<u>denge kriteri ihlali O halde bir diğer maksimum kazanca bakalım</u>**

Δg2(c) = -1 ve alan(A)=5 olur, Δg2(e) = -1 ve alan(A)=9 olur bunlardan birini seçebiliriz

c'yi baz olarak seçelim:

bölümler: A_2={b,c}, B_2={a,d,e} / daha önce seçilmiş {a,c}

3. İterasyon:

Kazanç:

(2. İterasyondaki şekle göre)

b: FS(b) = 2, TE(b) = 1, Δg3(b) = 1

d: FS(d) = 1, TE(d) = 1, Δg3(d) = 0

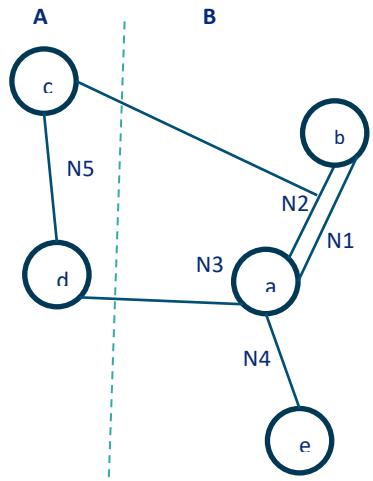
e: FS(e) = 0, TE(e) = 1, Δg3(e) = -1

b hücresi maksimum o halde:

Δg3(b) = 1 alan(A)=1 olur, denge kriterine uygun

Baz hücremiz b'dir :

Bölümler: A_3={c}, B_3={a,b,d,e}/ {a,c,b}



4.iterasyon

4. İterasyon:

Kazanç:

(3. İterasyondaki şekle göre)

d: FS(d) = 1, TE(d) = 1, Δg4(d) = 0

e: FS(e) = 0, TE(e) = 1, Δg4(e) = -1

d hücresi maksimum o halde:

Δg4(d) = 0, alan(A)=5 denge kriterine uygun

Baz hücremiz d:

Bölümler: A_4={c,d}, B_4={a,b,e} /{a,c,b,d}

5. İterasyon:

Kazanç:

(4. İterasyondaki şekle göre)

e: FS(e) = 0, TE(e) = 1, Δg5(e) = -1

Δg5(e) = -1, Alan(A)=10, denge kriteri uygun

Baz hücremiz e'dir:

Bölümler: A_4={c,d,e}, B_4={a,b}/ **tüm hücreler dahil edilmiştir**.

En iyi dizlimi seçmek: seçilmiş bazlara göre

G_1= Δg1(a) = 1

G_2= Δg1(a)+ Δg2(c) =0

G_3=Δg1(a)+ Δg2(c) +Δg3(b) = 1

G_4=Δg1(a)+ Δg2(c) +Δg3(b)+ Δg4(d) = 1

G_5=Δg1(a)+ Δg2(c) +Δg3(b)+ Δg4(d)+ Δg5(e)=0

Maksimum pozitif kümülatif kazanç: $Gm = \sum_{i=1}^{m} \Delta gi = 1$ **1,3 ve 4. İterasyonda**

en iyi denge oranı M=4,çünkü en küçük aana sahip (alan(A)=5)1
birinci geçişin sonucu: 4. İterasyondaki grafik gibidir Bölümler: A_4={c,d}, B_4={a,b,e}.
Diğer geçişler aynı yöntemle bulunur.

3.6 Fiduccia-Mattheyses (FM) algoritması sözde kodu:[4]

```
Fiduccia-Mattheyses Algorithm
Input: graph G(V,E), ratio factor r
Output: partitioned graph G(V,E)
1.   (lb,ub) = BALANCE_CRITERION(G,r)   // compute balance criterion
2.   (A,B) = PARTITION(G)               // initial partition
3.   Gm = ∞
4.   while (Gm > 0)
5.     i = 1
6.     order = Ø
7.     foreach (cell c ∈ V)             // for each cell, compute the
8.       Δg[i][c] = FS(c) – TE(c)       //   gain for current iteration,
9.       status[c] = FREE               //   and set each cell as free
10.    while (!IS_FIXED(V))             // while there are free cells, find
11.      cell = MAX_GAIN(Δg[i],lb,ub)   //   the cell with maximum gain
12.      ADD(order,(cell,Δg[i]))        // keep track of cells moved
13.      critical_nets = CRITICAL_NETS(cell) // critical nets connected to cell
14.      if (cell ∈ A)                  // if cell belongs to partition A,
15.        TRY_MOVE(cell,A,B)           //   move cell from A to B
16.      else                           // otherwise, if cell belongs to B,
17.        TRY_MOVE(cell,B,A)           //   move cell from B to A
18.      status[cell] = FIXED           // mark cell as fixed
19.      foreach (net net ∈ critical_nets) // update gains for critical cells
20.        foreach (cell c ∈ net, c ≠ cell)
21.          if (status[c] == FREE)
22.            UPDATE_GAIN(Δg[i][c])
23.      i = i + 1
24.    (Gm, m) = BEST_MOVES(order)     // move sequence c1 ... cm that
                                       //   maximizes Gm
25.    if (Gm > 0)
26.      CONFIRM_MOVES(order,m)        // execute move sequence
```

4. kaynakça

1. http://mm.iit.uni-miskolc.hu/data/texts/BOOKS/Artificial_Intelligence2/node23.html "Multimedia Maniacs Artificial Intelligence"
2. Andreev, Konstantin; Räcke, Harald (2004). *Balanced Graph Partitioning*. *Proceedings of the Sixteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures*. Barcelona, Spain. pp. 120–124
3. Sait, S.M., and Youssef, H.: "VLSI Physical Design Automation", McGraw-Hill Book Company, 1995.
4. Andrew B. Kahng • Jens Lienig Igor L. Markov • Jin Hu "VLSI Physical Design From Graph Partitioning to Timing Closure", Springer Science+Business Media B.V. 2011.