Journal of Computer Science
https://dergipark.org.tr/en/pub/bbd

ISSN,e-ISSN: 2548-1304
Volume:7, Issue:2, pp:103-110, 20 22
https://doi.org/10.53070/bbd.1199578
Research Paper

Hiperküplerde Hamilton çevriminin tespiti için yenilikçi bir yöntem

An innovative method for the detection of Hamiltonian cycles in Hypercubes

Fatih OKUMUŞ*[1], Ahmet KARADOĞAN[1]

[1]İnönü University, Department of Software Engineering, Malatya, Turkey

(fatih.okumus@inonu.edu.tr, ahmet.karadogan@inonu.edu.tr)

Received:Nov.04,2022 Accepted:Nov.25,2022 Published:Dec.07,2022

Özetçe— Hamilton döngüsü problemi (HCP), belirli bir çizge için iyi bilinen bir NP-zor sorunudur. Çizgelerde, ilk düğümden başlayan, her düğümü bir kez ziyaret eden ve tekrar ilk düğümde duran bir yol bulmak, HCP'nin birincil amacıdır. Oluşan diyagramda sonuç olarak tam bir döngü üretilir. Düzenli yapısı ve çeşitli kenar bağlantılarından dolayı hiperküpler, paralel makineler için tercih edilen topolojilerden biridir ve aynı zamanda Hamiltonyen çizgelerdir. Düğümler arasında ideal eşleşmenin kurulması, hiperküplerde Hamilton diyagramının oluşturulmasını gerektirir. Bu makalede, hiperküp bağlı çizgesinin Hamilton çevrimini ve bu çevrimi elde etmek için en uygun yolu bulan özgün bir yöntem önerilmektedir. Önerilen yöntem öncelikli olarak kenarlar için temel kesme kümelerini elde etmeye odaklanır. Temel kesme kümeleri etkili bir kapsam ağacı tekniği olan Kmax ve Kmin ağaçları oluşturularak hesaplanır. Bu iki ağaç üzerinde yapılan temel kesme işlemi sonunda her bir kenar için kesme sayısı elde edilir. Ardından, en yüksek kesme sayısına sahip kenardaki yüksek öncelikli düğümden düşük dereceli düğüme doğru bir yol takip edilir. Aynı düğüme tekrar gelmemek kaydıyla tüm düğümler bu yöntemle dolaşılır. Sonuç olarak her düğüm arasında bir Hamilton yolu oluşturulur. Önerilen algoritma ile yapılan deneysel çalışmalarda başarılı sonuçlar üretilmiş ve tüm hiperküpler için bir Hamilton çevrimi elde etmiştir.

Anahtar Kelimeler: *Çizgeler, Hiperküp, Hamilton çevrimi, Hamilton yolu, NP-zor problemler*

Abstract— The Hamilton cycle problem (HCP) is a well-known NP-hard problem for a given graph. In graphs, finding a path that starts from the first node, visits each node once, and stops at the first node again is the primary goal of HCP. In the resulting diagram, a complete cycle is produced as a result. Because of their regular structure and various edge connections, hypercubes are one of the preferred topologies for parallel machines and are also Hamiltonian graphs. Establishing the perfect match between the nodes requires the creation of the Hamilton diagram in hypercubes. This article proposes an original method that finds the Hamiltonian cycle of the Hypercube connected graph and the optimal way to derive this cycle. The proposed method primarily focuses on obtaining the basic cut-sets for the edges. Basic cut-sets are calculated by constructing Kmax and Kmin trees, which is an efficient spanning tree technique. At the end of the basic cut-set process on these two trees, the number of cuts for each edge is obtained. Then, a path is traced from the high priority node on the edge with the highest number of cuts to the lower degree node. All nodes are navigated with this method, provided that they do not come to the same node again. As a result, a Hamiltonian path is created between each node. Experimental studies with the proposed algorithm produced successful results and obtained a Hamiltonian cycle for all Hypercubes.

Keywords : *Graphs, Hypercube, Hamiltonian cycle, Hamiltonian path, NP-hard problems*

1.Giriş

Her sonlu bağlantılı köşe-geçişli çizgenin bir Hamilton yoluna, yani tüm köşelerden geçen basit bir yola sahip olup olmadığı sorusu uzun zamandır çözüm aranan NP-zor bir problemdir (Garey and Johnson, 1990; Karp,1972). NP-zor problemler üstel karmaşıklık gerektirir ve bu sebeple verilen çizge için Hamilton yolunu oluşturmaya yönelik birçok çalışma vardır. Bu konudaki tüm çalışmalar yaklaşıklık esasına dayanmaktadır.

Hiperküpler, düzenli yapıları ve çeşitli kenar bağlantıları nedeniyle paralel makineler için tercih edilen topoloji olmuştur (Boals ve diğerleri, 1994). Bu özellikler, düğümler arası iletişim için verimli algoritmaların geliştirilmesini sağlar. Çizge teorisinde, hiperküp Q_n olarak ifade edilir; 2^n düğümü ve $n2^{n-1}$ kenarı vardır. Bu topolojide Hamilton çizgesinin oluşturulması düğümler arasındaki mükemmel eşleşmenin oluşturulmasının önünü açmaktadır.

Hamilton çizgeleri hakkında literatürde birçok farklı çalışma yapılmış olup ilk olarak 1963 yılında Ore tarafından tanıtılmıştır (Ore, 1963). Bir çizgenin geçerli Hamiltonyen yollarını bulmak için bir ışık filtresi kullanan ışık tabanlı bir çözüm önerilmiştir (Sartakhti et al., 2013). Dikdörtgen ızgara grafiklerinde bir Hamilton yolu bulmak için doğrusal zaman algoritması sunulmuştur (Keshavarz-Kohjerdi and Bagheri, 2017). Dybizbański ve Szepietowski, çift olarak ayrık hatalı kenarları olan hiperküplerde Hamilton çevrimlerini ele almıştır (Dybizbański and Szepietowski, 2021). DeBiasio ve Spanier, Hamilton çevrimini dengeli k-parçalı çizgelerde hesaplamayı incelemiştir (DeBiasio ve Spanier, 2021). Agueda vd. paralel ve döngü kenarlı kenar renkli çoklu çizgelerde uygun Hamiltonyen çevrimi incelemiştir(Agueda ve diğerleri, 2017). Herhangi bir çizge iki Hamiltonyen çevrime ayrıştırılabilirse, bu grafiğin türünün ne olacağı sorusuna cevap aramışlardır(Sivaraman ve Zaslavsky,2022). Hatalı bükümlü hiperküplerdeki Hamilton çevrimleri ve yolları için çözümler sunulmuştur (Liu ve diğerleri, 2019). Düzlemsel üçgenlerdeki Hamilton çevrimlerinin sayısı hesaplanmıştır (Liu ve diğerleri, 2022).

Bu makalenin amacı, eğer varsa, verilen grafikte Hamilton çevrimini elde etmek için yeni bir yöntem önermek, aksi takdirde bağlı çizgeler için bir kapsayan yol elde etmektir. Bu şekilde verilen çizgeler için iki özel kapsayan ağaç oluşturulur. Verilen çizgelerin özel yayılma ağaçlarını elde etmek için iki yaklaşım vardır. Bu ağaçlar Karcı'nın çalışmalarında Kmax ve Kmin olarak adlandırılmıştır (Karci,2020a; Karci,2020b; Karci,2020c; Karci and Karci, 2020; Karci et al, 2021). Kmax ağacının oluşturulması için; öncelikle kök düğüm olarak maksimum dereceye sahip düğüm seçilir. Maksimum dereceye sahip birden fazla düğüm varsa, bunlardan biri isteğe bağlı olarak seçilebilir. Seçilen düğümün komşuları ağaca kökün çocukları olarak eklenir. Eklenen her bir düğüm için ağaçta olmayan komşuları bu düğümün çocukları olarak ağaca eklenir. Açılacak düğümün seçilmesi ise öncelikli olarak maksimum dereceye göre ve eşitlik durumunda ise ağaçtaki seviyesine göre belirlenir. Kmin ağacının oluşturulma aşamaları da Kmax ağacında olduğu gibi gerçekleşir. Ancak farklı olarak kök seçiminde ve açılacak düğümün belirlenmesinde minimum dereceli düğüm önceliklidir. Her iki ağaç da temel kesme kümelerini oluşturmak için kullanılır. Her temel kesme setinde kenar oluşum sayısı, karşılık gelen kenarın verimliliğini temsil eder. Bu verimlilik değerleri kullanılarak, çalışmada önerilen yenilikçi bir yol bulma algoritması ile Hamilton çevrimi elde edilmektedir. Bu çalışma ile ilk defa bağlı çizgeler için temel kesme kümeleri kullanılarak Hamilton çevrimi elde edilmiştir.

Çalışmanın üçüncü bölümünde sunulan deneysel çalışmalar göstermiştir ki; tüm hiperküpler için önerilen algoritma bir Hamilton çevrimi ve yolu bulmaktadır. Algoritmanın çalışma süresi nesne yönelimli inşa edilmesiyle birlikte oldukça kısadır. Özellikle büyük çizge veri setlerinde hız avantajı daha net izlenebilmektedir.

2. Önerilen Yöntem

Hamilton grafiğinin düğümleri arasındaki döngüyü oluşturan yolu bulmak için 4 adımlı bir algoritma öneriyoruz. İlk adımda, maksimum düğüm derecesine sahip düğümlere öncelik veren Kmax ağacı oluşturulur. İkinci adımda, minimum dereceye sahip düğümlere öncelik verilerek Kmin ağacı oluşturulur. Üçüncü adımda, Kmax ve Kmin'e dayalı temel kesme kümeleri kullanılarak Kmax ve Kmin ağaçlarının düğümlerini ağaçtan ayırmak için kenarların kesilmesi uygulanır. Her bir kenarın temel kesme sayısı, o kenarın etki değerini verir. Son adımda bu kesme değerlerine bağlı olarak yüksek öncelikli ayrıt ve bu ayrıttaki yüksek öncelikli düğüm seçilir. Seçilen bu düğümden başlayarak minimum dereceli komşuya doğru ilerleyen bir yol izlenir. Bu yol, grafikteki tüm düğümlere ulaşan Hamilton çevrimini oluşturacaktır.

Önerilen yöntem, nesne tabanlı bir altyapı ile geliştirilmiştir. Algoritmanın iki temel sınıfı, düğümleri temsil eden GraphNode ve ayrıtları temsil eden GraphEdge'dir. GraphNode sınıfının özellikleri Tablo 1'de, GraphEdge sınıfının özellikleri ise Tablo 2'de gösterilmiştir.

Tablo 1. GraphNode sınıfındaki özellikler

Özellik	Açıklama
_id	Her düğümü tanımlayan benzersiz kod
_label	Düğümün etiketi
_kMaxParentNode	Düğümün KMax ağacındaki atası olan düğüm
_kMinParentNode	Düğümün KMin ağacındaki atası olan düğüm
_kMaxChildren	Düğümün KMax ağacında atası olduğu çocuk düğümler
_kMinChildren	Düğümün KMin ağacında atası olduğu çocuk düğümler
_degree	Düğümün derecesi
_kMaxLevel	Düğümün KMax ağacında yer aldığı seviye (jenerasyon sırası)
_kMinLevel	Düğümün KMin ağacında yer aldığı seviye
_kMaxParentNodeEdge	Düğümün KMax ağacında ata düğümle yaptığı ayrıt
_kMinParentNodeEdge	Düğümün KMin ağacında ata düğümle yaptığı ayrıt
_neighbours	Düğümün komşuları

Tablo 2. GraphEdge sınıfındaki özellikler

Özellik	Açıklama
_id	Her ayrıtı tanımlayan benzersiz kod
_label	Ayrıtın etiketi
_node1	Ayrıtın bir ucundaki düğüm (sıralama önemli değil)
_node2	Ayrıtın diğer ucundaki düğüm
_isMaxChord	Ayrıt KMax ağacında yer alıyor mu?
_isMinChord	Ayrıt KMin ağacında yer alıyor mu?
_cutCount	Ayrıta uygulanan kesme sayısı

Tablo1'de her bir düğüm sınıfı için belirlenen özellikler, Tablo 2'de ise düğümler arasında oluşturulan her bir kenar için belirlenen özellikler gösterilmektedir. Bir kenarın hangi iki düğüm arasında oluştuğu bilgisi _node1 ve _node2 isimli özellik belirteçlerinde yer almaktadır. Tablo 1 ve Tablo 2'de belirtilen bu özellikler önerilen algoritmanın tüm aşamaları için yüksek öneme sahiptir. Özellikle ağaç içerisinde bir düğümün tüm çocuklarını ziyaret etmek için bu yapıya ihtiyaç vardır. Bir düğüm için temel kesme işlemi uygulandığında aynı işlem bu düğümün tüm çocuklarına da uygulanmaktadır. Çocuk düğümler arasında farklı bir ailedeki bir düğümle bağlantısı olan düğümler için de temel kesme işlemi uygulanır.

Yöntemin ilk aşaması, Kmax ağacının oluşturulmasını içerir. Kmax ağacı, maksimum dereceye sahip düğümden başlayarak grafiğin bir ağaç olarak açılması ve açılacak sıradaki düğümün seçilmesinde maksimum dereceye sahip olmasını ve minimum seviyede (ağaçtaki yerleşim seviyesi) yer almasını önceleyerek genişleme işleminin devam etmesi fikrine dayanmaktadır. Bu ağacı oluşturmanın amacı, yüksek dereceli düğümleri ön plana çıkarmaktır.

Algoritma, düğüm listesinin bir kopyasını oluşturur ve en yüksek derecedeki düğümü kök olarak belirler. Kmax ağacını temsil eden iki boyutlu bir düğüm matrisi oluşturulur ve kök düğüm, matrisin ilk satırına yerleştirilir. Kök düğüm ana düğüm olarak belirlenir ve tüm komşuları bir alt satıra yerleştirilir ve bu düğümler içindeki en yüksek dereceli düğüm yeni ebeveyn olarak belirlenir. Böylece ağaç, ağaca girmeyen düğüm kalmayana kadar inşa edilir. En yüksek dereceli düğümün seçimi için, ağaca giren tüm düğümler bir kuyruğa yerleştirilir. Bu kuyrukta düğümler derecelerine göre öne yerleştirilir. Aynı derecede düğümler varsa, ağaçta daha düşük seviyede olan öncelik alır. Bir düğüm ağaca girdiğinde,

tüm komşularının dereceleri bir azaltılır. Bu süreç, en yüksek dereceli düğümün seçilmesinde önemli bir rol oynar. Grafiğin Kmax ağacının oluşturulmasının sözde kodu Algoritma 1'de verilmiştir.

Algoritma 1. Kmax ağacının oluşturulması

```
Function Kmax(dugumList)
    ağacı başlat
    kök ← En yüksek dereceli düğüm
    ebeveyn ← kök
    ebeveyn düğümünü kuyruğa ekle
    kök düğümünü dugumList'ten sil

    dugumList.length > 0 olduğu sürece:
        ebeveyn düğümünü kuyruktan sil
            ebeveyn._neighbours içindeki her düğüm için:
            çocuk ← düğüm
            eğer çocuk ağaç içinde yoksa:
                çocuğun komşularının düğüm derecelerini 1 azalt
                çocuk düğümü kuyruğa ekle
                çocuk düğümü dugumList'ten sil

        ebeveyn ← Kuyruktaki en yüksek dereceli düğüm (ağaç seviyesi düşük olan öncelikli)

    ağacı döndür
```

Algoritmanın bir sonraki aşaması Kmin ağacını oluşturmaktır. Bu ağacın yapımında düğümün açılabilmesi için minimum derecede olması gerekmektedir. Esasen, tüm adımlar Kmax ağacının oluşturulmasına oldukça benzer. Ancak ana düğüm belirlenirken en yüksek dereceli düğüm yerine en düşük dereceli düğüm seçimi uygulanır. Kmin ağacı oluşturmanın amacı, düşük dereceli düğümleri ön plana çıkarmaktır. Bir çizge için Kmin ağacını oluşturan programın sözde kodu Algoritma 2'de verilmiştir.

Algoritma 2. Kmin ağacının oluşturulması

```
Function Kmin(dugumList)
    ağacı başlat
    kök ← En düşük dereceli düğüm
    ebeveyn ← kök
    ebeveyn düğümünü kuyruğa ekle
    kök düğümünü dugumList'ten sil

    dugumList.length > 0 olduğu sürece:
        ebeveyn düğümünü kuyruktan sil

        ebeveyn._neighbours içindeki her düğüm için:
            çocuk ← düğüm
            eğer çocuk ağaç içinde yoksa:
                çocuğun komşularının düğüm derecelerini 1 azalt
                çocuk düğümü kuyruğa ekle
                çocuk düğümü dugumList'ten sil

        ebeveyn ← Kuyruktaki en düşük dereceli düğüm (ağaç seviyesi düşük olan öncelikli)
```

```
ağacı döndür
```

Kmin ve Kmax ağaçları oluşturulduktan sonra, temel kesme kümelerinin inşası gerçekleştirilir ve tüm düğümleri ağaçlardan çıkarmak için kesme işlemi uygulanır. Bir düğümün ağaçtan ayırma işleminde, düğümü ve ata düğümü birleştiren ayrıt kesilir, buna ek olarak düğümün hiyerarşik sıradaki tüm alt öğelerinin kiriş bağlantıları da kesilir. Ancak bu kirişin diğer ucundaki düğüm, ağaçtan ayrılan düğümün çocukları arasında yer almamalıdır. Her kesme işleminde ayrıtın kesme sayısı bir artırılır. Kesme kümesini oluşturan yöntemin sözde kodu Algoritma 3'te verilmiştir.

Algoritma 3. Temel kesme kümesinin oluşturulması

```
Function KesmeKümesi(ağaç)
    mevcutDüğüm ← null

    ağaçtaki her bir satır için:
        satırdaki her bir düğüm için:
            düğümList ← ÇocuklarıBul(ağaç, düğüm)

            düğümList'teki her bir çocuk için:
                çocuğun her bir komşusu için:
                    eğer komşu düğümList içinde değilse:
                        kenarın kesme sayısını 1 artır
```

Bir düğümün ağaçtan koparılması esnasında çocuk düğümlerin kirişlerinin tespit edilmesi için düğümün tüm çocuklarına ulaşmak gerekmektedir. Düğümün tüm çocuklarına ulaşmak için, özyinelemeli bir yöntem yerine, sınırsız sayıda düğüme ulaşabilen $O(n^2)$ karmaşıklığında bir yöntem önerilmiştir. Yöntem, iki boyutlu matrisinin satırları ve sütunları arasındaki düğümün alt öğelerini bulmaya odaklanır. Düğümün bütün çocuklarını bulan yöntemi Algoritma 4'te verilmiştir.

Algoritma 4. Düğümün tüm çocuklarının bulunması

```
Function ÇocuklarıBul(ağaç, düğüm)
    düğümList ← [ ]
    idList ← [ ]
    seviye ← 0
    i ← 2

    i < ağaç.uzunluğu olduğu sürece:
        satır = ağaç[i]

        satırdaki her bir çocuk için:

            eğer idList içerisinde çocuk._parentNode._id varsa:
                çocuğun id'sini idList içine at
                çocuk düğümü düğümList'e ekle

    düğümList'i döndür
```

Önerilen yöntemin son aşaması, tüm düğümlerden geçerek bir Hamilton çevriminin oluşturulduğu yolu bulmaktır. Oluşturulacak yolun ilk düğümü, en fazla kesme sayısına sahip kenarın, diğer bir deyişle etkinlik değeri maksimum olan ayrıtın, önem derecesi en yüksek düğümüdür. Birinci düğüm seçildikten sonra aynı ayrıtın diğer ucundaki düğüm ikinci olarak belirlenir. Bu aşamadan sonra seçilecek bir sonraki düğüm için komşu düğümler arasında en düşük dereceye sahip olan seçilir ve bu seçim

ulaşılmayan düğüm kalmayana kadar devam eder. Aynı dereceye sahip birden fazla düşük dereceli komşu varsa bu durumda komşu kenarlar kontrol edilir ve ayrıtlarının toplam kesme sayısı en yüksek olan düğüm seçilir. Kesme sayılarında da eşitlik varsa bunlardan herhangi biri seçilir ve devam edilir. Yola eklenen her düğüm listeden kaldırılır, bu nedenle bir düğüm en fazla bir kez ziyaret edilir. Ağaca son düğüm eklendiğinde, ilk düğüm arasında bir kenar varsa, Hamilton döngüsü oluşacaktır. Hamilton yolu bulma yöntemi Algoritma 5'te gösterilmiştir.

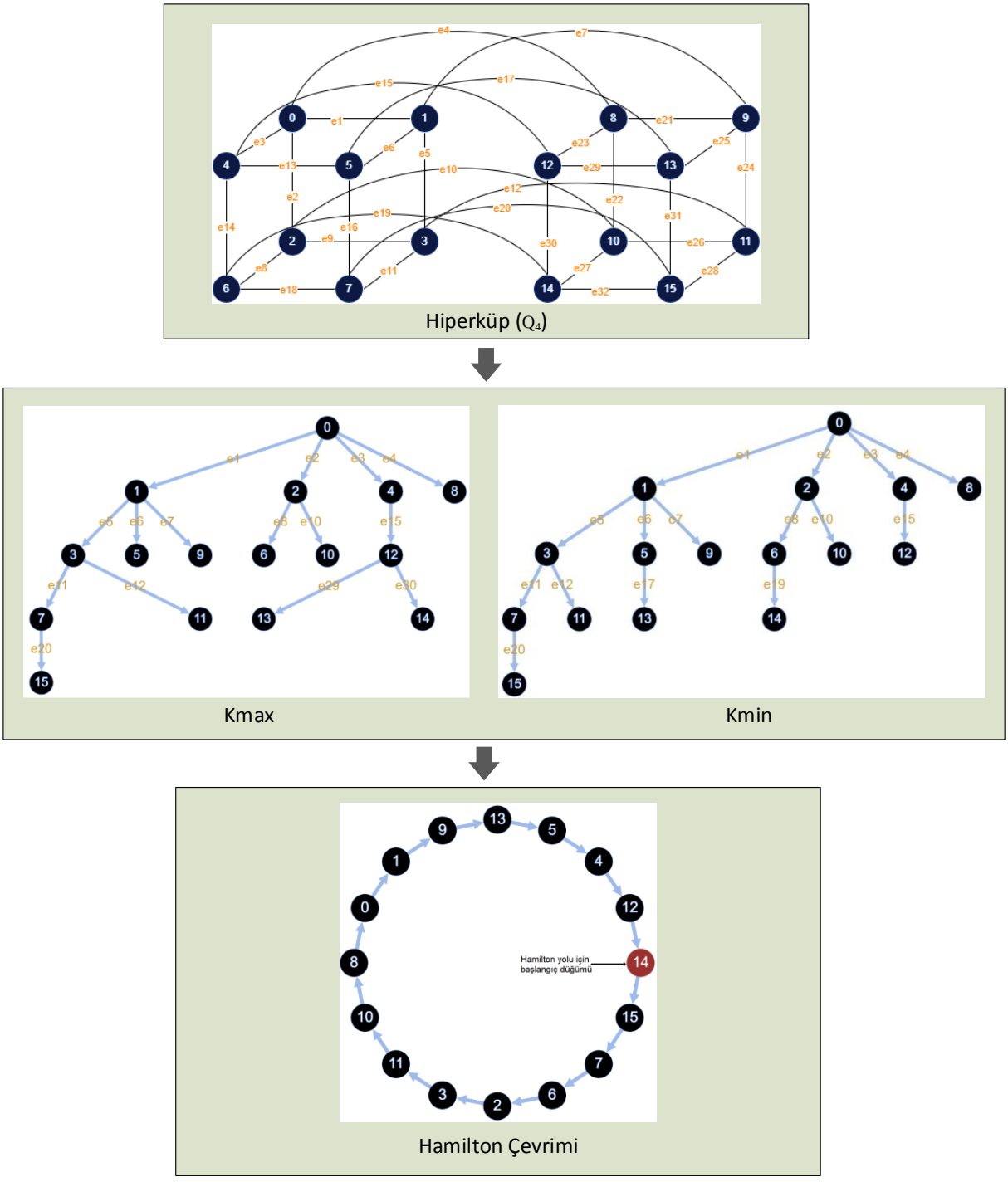
Algoritma 5. Hamilton çevriminin ve yolunun bulunması

```
Function ÇevrimiBul()
    hamilton =[ ]
    öncelikliKenar ← Kesme sayısı en fazla olan kenar
    düğüm1 ← oncelikliKenar'ın yüksek öncelikli düğümü
    düğüm2 ← oncelikliKenar'ın düşük öncelikli düğümü
    düğüm1'i hamilton'a ekle
    düğüm2'yi hamilton'a ekle
    mevcutDüğüm ← düğüm2

    mevcutDüğüm'üm komşuları varsa ve bu komşulardan hamilton'a eklenmeyen varsa devam et:
        mevcutDüğüm ← hamilton'a eklenmeyen en düşük dereceli ve düşük öncelikli komşu
        mevcutDüğüm'ü hamilton'a ekle
        çocuğun komşularının derecesini 1 azalt

    hamilton'u döndür
```

3. Deneysel Sonuçlar

Önerilen algoritmanın güvenilirliğini ölçmek için Q_4 hiperküp örneğine uygulanmıştır. Şekil 1'de, 16 düğümü ve 32 kenarı olan hiperküp Q_4 için önerilen algoritmanın çalışma adımları gösterilmektedir. Hiperküp bir Hamilton çizgesidir ve önerilen algoritma, Şekil 1'de gösterildiği gibi Hamilton döngüsünü ve yolunu tespit etmiştir. Verilen örnek için algoritma, en etkili kenarın 15. ve 16. düğümler arasında olduğunu belirlemiştir. Ayrıca 15. düğüm Hamilton yolunun ilk düğümü olmuştur.

Verilen tüm örneklerde (Q_2, Q_3, Q_4) hiperküpler için Hamilton çevrimi algoritma tarafından tespit edilmiştir. Algoritmanın üstel karmaşıklığı $O(n^2)$ olması sayesinde büyük çizge veri setlerinde dahi oldukça kısa sürede Hamilton çevrimi tespit edilmiştir.



Şekil 1. Hiperküp için oluşan uygulama adımlarının görsel çıktıları

4. Sonuç

Hiperküplerde olduğu gibi herhangi bir bağlı çizge için Hamilton yolu veya Hamilton döngüsü bulma problemi NP-tam bir problemdir. Literatürde Hamiltonyen yolunu/çevrimini bulma probleminin çözümüne yönelik birçok çalışma bulunmaktadır. Ancak bunların hiçbiri, kapsayan ağaçları ve temel kesme kümelerini kullanmamıştır. Bu yazıda, çizgeler için iki özel kapsama ağacı (Kmax ve Kmin) kullanarak temel kesme kümeleri oluşturulmuş, bu temel kesme kümeleri referans alınarak Hamilton yolunu/çevrimini bulmak için yenilikçi bir metot önerilmiştir. Önerilen metot, temel kesme işlemlerinin ardından oluşan kenar verimliliklerine odaklanmaktadır. Kenar verimlilik sayıları kullanılarak Hamilton

yolu için düğümlerin hangi sırayla geçilmesi gerektiğini belirleyen bir yöntem uygulanmıştır. Tüm düğümler bir kez ziyaret edildiğinde son adım tamamlanmış ve böylece Hamilton yolu oluşturulmuştur.

Önerilen yöntem tamamen yinelemeli algoritmalar içerir, bu nedenle yöntemin zaman karmaşıklığının bir polinom işleviyle sınırlandırılması garanti edilmiştir. Algoritma üzerinde yapılacak geliştirmeler sonunda Hiperküp gibi diğer ağırlıklandırılmamış çizgelere de uygulanabilecektir. Ayrıca ağırlıklandırılmış çizgeler için de algoritma revize edilecektir.

Kaynaklar

M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*. USA: W. H. Freeman & Co., 1990.

Karp, R. M. *Reducibility Among Combinatorial Problems*, In Complexity of Computer Computations (Ed. R. E. Miller and J. W. Thatcher). New York: Plenum Press, pp. 85-103, 1972.

Boals, A.J., Gupta, A.K., Sherwani, N.A., *Incomplete hypercubes: Algorithms and embeddings*, The Journal of Supercomputing Vol:8, pp:263-294, 1994.

O. Ore, "Hamilton-connected graphs," *J Math Pure Appl*, vol. 42, pp. 21–27, 1963.

J. S. Sartakhti, S. Jalili, and A. G. Rudi, "A new light-based solution to the Hamiltonian path problem," *Future Gener. Comput. Syst.*, vol. 29, no. 2, pp. 520–527, Feb. 2013, doi: 10.1016/j.future.2012.07.008.

F. Keshavarz-Kohjerdi and A. Bagheri, "A linear-time algorithm for finding Hamiltonian (s,t)-paths in even-sized rectangular grid graphs with a rectangular hole," *Theor. Comput. Sci.*, vol. 690, pp. 26–58, Aug. 2017, doi: 10.1016/j.tcs.2017.05.031.

J. Dybizbański and A. Szepietowski, "Hamiltonian cycles and paths in hypercubes with disjoint faulty edges," *Inf. Process. Lett.*, vol. 172, p. 106157, Dec. 2021, doi: 10.1016/j.ipl.2021.106157.

DeBiasio, L., Martin, R.R., Molla, T.,*Powers of Hamiltonian cycles in multipartite graph*s, Discrete Mathematics, Vol:345, 112747, 2022.

Agueda, R., Borozan, V., Diaz, R., Manoussakis, Y., Montero, L.,*Proper Hamiltonian cycles in edge colored multigraphs*, Discrete Mathematics, Vol:340, pp:1897-1902, 2017.

Sivaraman, V., Zaslavsky, T., *Two Hamiltonian cycles*, Discrete Mathematics, Vol:345, 112797, 2022.

Liu, H., Hu, X., Gao, S.,*Hamiltonian cycles and paths in faulty twisted hypercubes*, Discrete Applied Mathemaitics, Vol:257, pp:243-249, 2019.

Liu, X., Wang, Z., Yu, X., *Counting Hamiltonian cycles in planar triangulations*, Journal of Combinatorial Theory, series B, Vol:155, pp:256-277, 2022.

A. Karci, "New Algorithms for Minimum Dominating Set in Any Graphs," 2020.

A. Karci, "Finding Innovative and Efficient Solutions to NP-Hard and NP-Complete Problems in Graph Theory," *Comput. Sci.*, vol. 5, no. 2, pp. 137–143, 2020.

A. Karci, "Efficient Algorithms for Determining the Maximum Independent Sets in Graphs," *Comput. Sci.*, vol. 5, no. 2, pp. 144–149, 2020.

A. Karci and Ş. Karci, "Determination of effective nodes in graphs," *Int. J. Adv. Comput. Eng. Netw.*, vol. ISSN, no. 8, pp. 2321–2063, 2020.

Ş. KARCI, A. ARI, and A. KARCİ, "Pençesiz çizgelerde maksimum-yakın bağımsız küme ve üst sınırları için yeni algoritma," *Gazi Üniversitesi Mühendis.-Mimar. Fakültesi Derg.*, Oct. 2021, doi: 10.17341/gazimmfd.902093.