TAM SAYI PROGRAMLAMA TABANLI BİR TUZAK KÜMESİ ARAMA TEKNİĞİ

AN INTEGER PROGRAMMING BASED TRAPPING SET SEARCH TECHNIQUE

Abdullah Sarıduman[1], *Ali Emre Pusane*[1], *Z. Caner Taşkın*[2]

[1]Elektrik-Elektronik Mühendisliği Bölümü, Boğaziçi Üniversitesi
[2]Endüstri Mühendisliği Bölümü, Boğaziçi Üniversitesi
{abdullah.sariduman, ali.pusane, caner.taskin}@boun.edu.tr

ÖZETÇE

Düşük-yoğunluklu eşlik-denetim kodların (LDPC) kod çözümünde oluşan hataların başlıca nedenlerinden biri yakın kod sözcükleridir. Yüksek işaret gürültü oranları için kod çözümünde hata tabanına neden olan yakın kod sözcükleri, tuzak kümeleri olarak adlandırılmaktadır. Özellikle küçük boyutlu tuzak kümeleri, LDPC kodlarla kodlanmış iletişim sistemlerinde önemli sorunlara neden olmaktadır. Ayrıca, tuzak kümeleri neredeyse bütün eşlik denetim eşitliklerini sağladıkları için bulunmaları zordur. Bu çalışmamızda tam sayı programlama eniyileme yöntemi kullanılarak en küçük boyutlu tuzak kümesi aranmaktadır. Bundan başka, en küçük tuzak kümelerinin içerisinde yer alan ve kod çözücüyü tuzak kümesi içerisinde salınım yaptıran en küçük boyuttaki bit grubu, geliştirdiğimiz algoritma ile belirlenebilmektedir.

ABSTRACT

Near codewords of low-density parity-check (LDPC) codes are known to be one of the main reasons of errors that occur in decoding. For high signal-to-noise ratios, these codewords cause error floors in decoding and they are called trapping sets. Especially, trapping sets with small size devastate the performance of communication systems that use LDPC codes. Unfortunately, trapping sets are difficult to find since they satisfy almost all of the parity check equations. In this paper, we develop an integer programming based optimization approach to find the smallest trapping set. Moreover, we develop an algorithm to determine the smallest bit set that belongs to the smallest trapping set and causes an oscillation in the decoder.

1. GİRİŞ

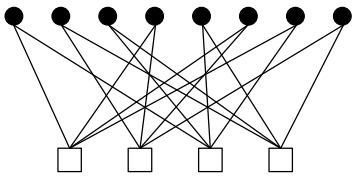
İlk defa 1962 yılında Gallager tarafından geliştirilen LDPC kodları, yüksek hata başarımları ve geniş kullanım alanları sayesinde yeniden önem kazanmış ve günümüzün en popüler hata düzelten kodlarından biri olmuştur, [1]. Fakat, sonlu-uzunluklu LDPC kodların bazı problemleri vardır. Bu problemlerden birisi de hata tabanıdır. Hata tabanı, yinelemeli kod çözücülerin yüksek işaret gürültü oranları için çerçeve hata oranı (FER) eğrisindeki ani düzleşmedir. Richardson, yarı-analitik bir teknik ile LDPC kodların hata tabanlarını hesaplamayı ve hata tabanlarına neden olan bit hata setlerini belirlemeyi başarmış ve bu setleri tuzak kümesi olarak adlandırmıştır, [2]. Ayrıca, hata tabanı bölgesindeki başarımın büyük ölçüde küçük boyutlu tuzak kümeleri tarafından belirlendiğini belirtmiştir. Bunun nedeni de küçük boyutlu tuzak kümelerinin, yani bit hata setlerinin, gerçekleşme olasılığının daha yüksek olmasıdır. Ancak, bir LDPC kodun sahip olduğu tuzak kümelerinin boyutlarını belirlemek NP-zordur, [3]. Ayrıca tuzak kümelerin alt kümesi olan bazı özel bit hata setleri de tuzak kümesi gibi davranabilmektedir. Tuzak kümesinin sahip olduğu bu özel bit hata setlerinin en küçük boyutlu olan setinin elaman sayısına, ilgili tuzak kümesinin kritik sayısı denmektedir, [4].

Çalışmamızda, tam sayı programlama eniyileme yöntemi kullanılarak en küçük boyutlu tuzak kümesi aranmaktadır. En küçük tuzak kümesi bulunduktan sonra, bu kümeye ait kritik sayı, geliştirilen algoritma ile kısa süre içerisinde bulunmaktadır. Bu verilerin ışığında, hata tabanını büyük ölçüde belirleyen tuzak kümesi, LDPC kod içerisinde yok edilerek hata tabanı daha küçük değerlere indirilebilir. [2]'de bilgisayar benzetimleriyle çalışılmış, kod çözücünün takıldığı vektörler incelenerek tuzak kümesi kavramı ortaya atılmıştır. Bizim çalışmamızda ise verilen bir kod için olası tüm tuzak kümeleri arasından en küçük boyutlu tuzak kümesini bulacak bir eniyileme yaklaşımı geliştirilmiştir. En küçük tuzak kümesini bulmak NP-zor olduğu için bu konu hakkında çok az çalışma vardır. Katışımsal bir algorima kullanan [5], tuzak kümelerinin özel bir durumu olan k-dışarı tuzak kümelerinin en küçük boyutunu bulmuştur. Tuzak kümesi arayan diğer çalışmalarda genellikle kod hakkında bazı özelliklerin bilindiği varsayılmıştır ya da geliştirilen algoritma bütün tuzak kümelerini bulamamıştır, [6], [7].

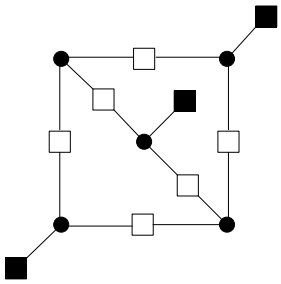
Bildirinin geri kalanı şu şekilde düzenlenmiştir: Bölüm 2'de LDPC kodlara ait tanımlamalar verilmiş ve tuzak kümesinin topolojik yapısından söz edilmiştir. Bölüm 3'te geliştirilmiş olan tam sayı programlama modeli ve en küçük tuzak kümesi ile ona ait kritik sayıyı bulan algoritma anlatılmıştır. Bölüm 4'de sayısal sonuçlar gösterilmiş ve Bölüm 5'de sonuçlar ve planlanan gelecek çalışmalar üzerinde durulmuştur.

Bu bildirideki çalışmalar Avrupa Birliği tarafından PIRG07-6A-2010-268264 nolu proje kapsamında desteklenmiştir.

978-1-4673-0056-8/12/$26.00



Şekil 1:: (8,4) LDPC kodu Tanner grafı.



Şekil 2:: (5,3) tuzak kümesi.

2. TEMEL KAVRAMLAR

(n, k) boyutlu bir LPDC kodu, Tanner graf olarak adlandırılan iki bölmeli bir graf üzerinde n bit düğümü ve $m = n - k$ denetim düğümü ile gösterilebilir. Bit düğümleri kod çözümü yapılacak olan kod sözcüğüne, denetim düğümleri ise eşlik denetim matrisinin oluşturduğu eşitliklere karşılık gelmektedir. Bir bit düğümünün bağlı olduğu bir denetim düğümü onun komşusu olarak adlandırılır ve bu tanımın tersi de geçerlidir. Bir düğümün derecesini bağlı olduğu komşuların sayısı belirler. Denetim düğümlerinin değerlerini (doğru veya hatalı) bağlı oldukları bit düğümleri belirler. Eğer bit düğümlerinden gelen mesajların toplamı çift ise denetim düğümü doğru durumundadır, tek ise denetim düğümü hatalı durumundadır. $\mathbf{H}$ matrisi Tanner grafı temsil eden iki bölmeli bitişiklik matrisidir. $\mathbf{v} = \{v_1, v_2, ..., v_n\}$ vektörü $\mathbf{vH^T} = 0 \mod (2)$ eşitliğini sağlar ise geçerli bir kod sözcüğüdür. (Bundan sonra $\mod (2)$ ifadesi kullanılmayacaktır.) Başka bir ifadeyle bütün denetim düğümleri $\mathbf{v}$ vektörü için doğru durumdaysa, $\mathbf{v}$ geçerli bir kod sözcüğüdür.

Örnek 1: (8,4) LDPC kodunun $\mathbf{H}$ matrisi için Tanner grafı Şekil 1'de gösterilmiştir. Bit düğümleri • ile, denetim düğümleri □ ile gösterilmiştir. Eşlik denetim matrisi

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

ile verilmiştir.

Kod sözcüğünün çözümünde yumuşak karar ya da kesin karar (0 ya da 1 kararı) kullanılabilir. Richardson, toplanır beyaz Gauss gürültülü (AWGN) kanallarda, yumuşak karar kullanan kod çözücülerin kesin karar kullananlarla aynı tuzak kümelerine takıldığını göstermiştir, [2]. Bundan dolayı, kolaylık sağlaması amacıyla tuzak kümeleri kesin karar ile çalışan kod çözücüler için bulunur. Sonuç olarak, bu tuzak kümeleri yumuşak karar ile çalışan kod çözüler için de geçerlidir. Yinelemeli bir kod çözücüsü olan mesaj geçirme algoritmasında her bir yinelemede, bit düğümleri ile denetim düğümleri arasındaki karşılıklı iletişim sonucunda düğümlerin değerleri değişir. Yineleme işlemi geçerli bir kod sözcüğüne ulaşılana kadar tekrarlanır. Çalışmamızda, kesin karar ile çalışan mesaj geçirme algoritmalı kod çözücüler göz önüne alınmıştır. Kesin karar verilirken bit evirme işlemi kullanılır. Bir bit düğümüne denetim düğümlerinden gelen mesajların çoğunluğu yanlış ise bu düğüm evrilerek düzeltilir. Gallager B algoritmasında, bu değişim bütün düğümler için aynı anda parelel biçimde yapılır, [1].

Richardson, tuzak kümesini kod çözünün çözemediği hatalı bit düğümlerinin kümesi olarak tanıtmıştır, [2]. Buna göre, bir (a, b) tuzak kümesi a tane bit düğümünden ve bu düğümlerin komşusu olan tek dereceli b tane denetim düğümlerinden oluşur.

Örnek 2: [8]'da verilen Tanner (155,64) kodunun Gallager B kod çözücü algoritması ile çözümünde meydana gelen (5,3) tuzak kümesi örneği Şekil 2'de verilmiştir. Bit düğümleri • ile, tek dereceli denetim düğümleri ■ ile ve çift dereceli denetim düğümleri □ ile gösterilmiştir. Bu tuzak kümesindeki bütün denetim düğümleri hatalı olduğu zaman, yineleme aşamasında her bir bit düğümü için denetim düğümlerinden gelen mesajların çoğunluğu o bit düğümünün doğru olduğunu söyleyecektir. Bunun sonucunda hiçbir bit düğümünün değeri tersine çevrilmeyecek ve kod çözücü bu noktada hataları düzeltmeden takılacaktır.

Kod çözücünün bir tuzak kümesi içerisinde tıkanması için tuzak kümesine ait bütün bit düğümlerinin hatalı olmasının gerekmediği yakın zamanda gösterilmiştir, [4]. Bir tuzak kümesinin kod çözücünün takılmasına neden olduğu asgari hatalı bit düğüm sayısına tuzak kümesinin kritik sayısı denmektedir. İki tuzak kümesi arasında kritik sayısı daha küçük olan tuzak kümesinin, daha büyük boyutlu olsa bile, FER performansını daha fazla etkilediği gösterilmiştir, [4].

Örnek 3: Şekil 2'deki tuzak kümesi için kritik sayı 2'dir. Sağ üst ve sol alt köşelerdeki bit düğümlerinin hatalı olması kod çözücünün tuzak kümesi içersinden çıkamaması ve geçerli bir kod sözcüğü bulamaması için yeterli olacaktır. İlk yinelemede, denetim düğümleri bütün bit düğümleri evrilmesini sağlayacaktır. İkinci yinelemede de aynı şekilde bütün bit düğümleri evrilecektir. Sonuç olarak, kod çözücü başlangıç durumuna gelecek ve bu yineleme işlemleri sonsuza kadar sürecektir.

3. EN KÜÇÜK BOYUTLU TUZAK KÜMESİNİN VE BU TUZAK KÜMESİNE AİT KRİTİK SAYININ BULUNMASI

Doğrusal programlama eniyileme kuramının temel tekniklerinden biridir ve birçok farklı alanda karşılaşılan eniyileme problemlerinin modellenip çözülmesi için yaygın olarak kullanılmaktadır. Doğrusal programlama yönteminde, öncelikle çözülmesi gereken problemde yer alan her karar noktası birer *karar değişkeni* olarak modellenir. Bulunmak istenen çözümün sağlaması gereken *kısıtlar*, karar değişkenlerinin doğrusal birer fonksiyonu olarak ifade edilir.

Benzer şekilde, kısıtları sağlayan her *olurlu çözümün* kalitesini ölçen, karar değişkenlerinin doğrusal bir fonksiyonu da *amaç fonksiyonu* olarak tanımlanır. Genel bir doğrusal programlama problemi aşağıdaki gibi ifade edilebilir:

$$\text{enazla } \mathbf{c}^T\mathbf{x} \tag{1}$$
$$\text{kısıtlar: } \mathbf{Ax} = \mathbf{b} \tag{2}$$
$$\mathbf{Dx} \geq \mathbf{e} \tag{3}$$
$$\mathbf{Fx} \leq \mathbf{g} \tag{4}$$
$$\mathbf{x} \geq 0. \tag{5}$$

Bu şekilde bir doğrusal programlama problemi verildiğinde tüm kısıtları sağlayan ve amaç fonksiyonunun mümkün olan en küçük değeri almasını sağlayan bir *eniyi çözüm* polinom zamanda bulunabilir, [9].

Doğrusal programlama yönteminde karar değişkenlerinin kesirli değerler alabileceği varsayılmaktadır. Öte yandan, ele aldığımız problemi de içeren bazı durumlarda bu kabul gerçekçi olmamakta ve karar değişkenlerinin mutlaka tam sayı değerleri almaları gerekmektedir. Doğrusal programlamanın, değişkenlerin alabileceği değerlerin tam sayılarla sınırlandırıldığı şekline tam sayı programlama adı verilir. Genel bir tam sayı programlama problemi aşağıdaki gibi ifade edilebilir:

$$\text{enazla } \mathbf{c}^T\mathbf{x} \tag{6}$$
$$\text{kısıtlar: } \mathbf{Ax} = \mathbf{b} \tag{7}$$
$$\mathbf{Dx} \geq \mathbf{e} \tag{8}$$
$$\mathbf{Fx} \leq \mathbf{g} \tag{9}$$
$$\mathbf{x} \in Z^+. \tag{10}$$

Doğrusal programlama problemlerinin polinom zamanda çözülebilir olmasına karşın, tam sayı programlama problemlerini çözmek NP-zordur. Buna rağmen tam sayı programlama problemleri, dal-sınır ve dal-kesi gibi algoritmalar kullanılarak pratikte oldukça büyük boyutlu problemler için çözülebilmektedir, [10].

3.1. En Küçük Boyutlu Tuzak Kümesinin Bulunması

Sıfır kod sözcüğü $\mathbf{vH^T} = 0$ eşitliğini bütün $\mathbf{H}$ matrisleri için sağladığından dolayı geçerli bir kod sözcüğüdür. Sıfır kod sözcüğünün verici tarafından gönderildiği varsayıldığında gelen vektördeki bit düğümlerin 1 değerine sahip olanlarının hepsi hatalı durumdadır. Bir tuzak kümesinin oluşması için tuzak kümesine ait olan bit düğümlerinin 1 değerine sahip olması ve her bit düğümünün komşu denetim düğümlerinin çoğunluğunun doğru durumunda olması gerekmektedir. Amacımız en küçük tuzak kümesini oluşturan gelen vektörü elde etmektir. Getireceğimiz kısıtlar yardımıyla, oluşturulan vektörün 1 değeri alan bitleri bize tuzak kümesini verecektir. Öncelikle gelen n uzunluklu vektörün her i. bit değeri için v_i karar değişkeni tanımlanır. Bu durumda, en küçük tuzak kümesini elde edebilmek için amaç fonksiyonu $\sum_{i=1}^{n} v_i$ olarak belirlenir. Kısıtlarda ise m adet denetim düğümü için ikili $\mathbf{c} = \{c_1, c_2, ..., c_m\}$ vektörü tanımlanır. Bu vektör, denetim düğümlerinin durumunu tutar. c_i değişkeni 1 değerinde ise i. denetim düğümü hatalı, 0 değerinde ise i. denetim düğümü doğru durumunda olacak şekilde c_i değişkenin değeri belirlenir. Bütün denetim düğümleri doğru durumunda olduğunda kod vektörü geçerli bir kod sözcüğü olacaktır ve dolayısıyla bir tuzak kümesi oluşmayacaktır. Bu durumu engellemek için $\sum_{i=1}^{m} c_i \geq 1$ kısıtı kullanılır. Eğer gelen vektör bir kod sözcüğü ise $\mathbf{vH^T} = 0$ eşitliği sağlanacaktır. Ancak biz, gelen vektörün en küçük tuzak kümesini oluşturmasını istiyoruz. Karar değişkenlerini $\mathbf{vH^T} + \mathbf{c} = 0$ eşitliğini sağlayacak şekilde seçersek gelen vektör bir kod sözcüğü olmayacaktır çünkü $\mathbf{c}$ vektörünün sıfır vektörüne eşit olması bir önceki kısıt ile engellenmiştir. Ayrıca her bit düğüm için komşu olduğu c_i'lerin toplamı, bu düğümün toplam komşu sayısının yarısından az olur ise bir düğümünün komşuları olan denetim düğümlerinin çoğunluğunun doğru durumunda olması şartı da sağlanmış olur. Bu durumu her bit düğümü için matematiksel olarak $\mathbf{cH_i} \leq \sum_{j=1}^{m} H_{i,j}$ ile ifade edebiliriz. ($\mathbf{H_i}$, $\mathbf{H}$ matrisinin i. sütununu gösterir.) Bu bilgiler ışığında en küçük tuzak kümesini bulan tam sayı programlama modeli aşağıdaki gibi ifade edilmiştir:

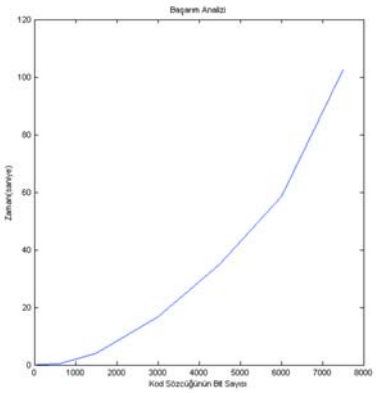
$$\text{enazla} \sum_{i=1}^{n} v_i \tag{11}$$
$$\text{kısıtlar: } \mathbf{vH}^T + \mathbf{c} = 2\mathbf{k} \tag{12}$$
$$\mathbf{cH} \leq \mathbf{y}/2 \tag{13}$$
$$\sum_{i=1}^{m} c_i \geq 1 \tag{14}$$
$$y_i = \sum_{j=1}^{m} H_{i,j}, \quad \forall i = 1, 2, ..., n \tag{15}$$
$$c_i \in \{0, 1\}, \quad k_i \in \mathbb{Z}^+, \quad \forall i = 1, 2, ..., m \tag{16}$$
$$v_i \in \{0, 1\}, \quad \forall i = 1, 2, ..., n. \tag{17}$$

Tam sayı programlamada doğrusal olmayan $\mod(2)$ için bir düzenleme bulunmamaktadır. Bu nedenle k tam sayı değişkeni tanımlanmıştır. (12) eşitliğinde tam sayı programlamanın k tam sayısını eşitliğin sol tarafına göre uygun bir tam sayı olarak seçmesi ve bu sayının 2 ile çarpılması $\mod(2)$'de 0 değerine karşılık gelmektedir.

3.2. En küçük Tuzak Kümesinin Kritik Sayısının Bulunması

Önceki bölümde sunulan tam sayı programlama modelinin çözülmesiyle en küçük tuzak kümesi bulunduktan sonra ilgili tuzak kümesine ait olan kritik sayı, Algoritma 1 ile bulunur. En küçük tuzak kümesine ait olan bit düğümleri ve denetim düğümleri kullanılarak kritik sayıya ulaşılır. Bu algoritmada işlemleri kolaylaştırmak ve hızlandırmak için bir tablo hazırlanmıştır. Bu tablo yardımıyla çözülen bir kod sözcüğünün daha önce işlenip işlenmediği kontrol edilir.

Algoritma 1'de, tuzak kümesi içerisinde yer alan bütün bit grupları sırayla üretilmekte ve kod çözücüyü tuzak kümesi içerisinde salınıma sokan ilk bit grubu aranmaktadır. Eğer bit grupları Hamming ağırlığına göre sırayla üretilirse, bulunan ilk grup en küçük boyuttaki bit grubu olacaktır. Kod tablosundan okunan vektörlerin kod çözümü sırasında oluşturdukları bütün vektörler kaydedilir. Kod tablosundan okunan vektör kod çözümü sırasında önceden kaydedilmiş bir vektöre ulaşabilirse daha fazla işleme gerek yoktur. Çünkü, daha önce bu vektör çözülmüştür. Diğer yandan kod tablosundan okunan vektör daha önceden kendi oluşturduğu bir vektöre dönüşmüş ise kod çözücü salınım yapmaya başlamış demektir.



Şekil 3:: En küçük tuzak kümelerinin tam sayı programlama ile bulunması için gereken süre.

Algoritma 1 Verilen tuzak kümesinin kritik sayısının bulunması

1) $\mathbf{H}$ matrisinin tuzak kümesine ait olan düğümleri içeren alt grafı $\mathbf{H^1}$ ve bu düğümler için gerçekleşmesi olası bütün a uzunluklu ikili vektörler (sıfır vektörü hariç) Hamming ağırlığına göre sırayla üretilir.
2) Vektörlerin oluşturduğu kod tablosundan sıradaki ilk vektör alınır ve sıra numarası kaydedilir.
3) Vektör sadece bir yineleme kod çözümüne tabi tutulur.
4) Oluşan yeni vektörün 10'luk düzlemdeki değerine karşılık gelen yerde bir sıra numarasının olup olmadığına tablodan bakılır. Sıra numarası yok ise 2. aşamada kaydedilen sıra numarası buraya yazılır. Sıra numarası var ise 6. aşamaya geçilir.
5) Üretilen yeni $\mathbf{x}$ vektörü $\mathbf{H^1x} = 0$ eşitliğini sağlıyor ise 2. aşamaya geri dönülür. Sağlamıyor ise 3. aşamaya dönülür.
6) 2. aşamada kaydedilen sıra numarası ile 4. aşamada okunan sıra numarası aynı değil ise 2. aşamaya geri dönülür, aynı ise kod tablosundan en son okunan vektörün Hamming ağırlığı kritik sayıya eşittir.

4. SAYISAL SONUÇLAR

Permütasyon matrisi kullanarak (2,2) tuzak kümesi içeren farklı boyutlarda $\mathbf{H}$ matrisleri elde edilmiştir. Bölüm 3.1'de tarif edilen tam sayı programlama modeli, CPLEX 12.3 yazılımıyla çözülerek her bir $\mathbf{H}$ matrisinin en küçük boyutlu tuzak kümesi (boyutları 2) başarıyla bulmuştur. Bölüm 3'te açıklandığı üzere, tam sayı programlama probleminin çözümü NP-zor olmakla birlikte pratikte tam sayı programlama problemleri olukça etkin şekilde çözülebilmektedir. Şekil 3'te kod sözcüğünün uzunluğuna göre programın çalışma süresinin, uygulamada görülebilecek pratik LPDC kod uzunlukları için tatmin edici olduğu gösterilmiştir. En küçük tuzak kümesini bulunması sadece kod tasarımı ve analizi sırasında yapıldığı için gerçek zamanlı çözülmesi gereken bir problem olmamasına rağmen, küçük tuzak kümelerinin hızlı bir şekilde bulunması diğer algoritmalar içerisinde kullanılabilmesine imkan tanıması açısından önem teşkil etmektedir.
En küçük tuzak kümesine ait kritik sayıyı bulmak için Tanner'in sözde-dolanır matrislerini temel alarak oluşturduğu LDPC kodları kullanılmıştır, [8]. Tanner (155,64) kodu için en küçük tuzak kümesinin boyutu ve ilgili tuzak kümesine ait olan kritik sayı 5 olarak bulunmuştur. Diğer boyutu 5 olan tuzak kümeleri araştırıldığında, kritik sayısı 2 olan bir tuzak kümesi elde edilmektedir. Şekil 2'deki tuzak kümesi, Tanner (155,64) kodu için bulunan en küçük boyutlu ve en küçük kritik sayıya sahip olan tuzak kümesidir.

5. SONUÇLAR

Bütün kod çözücü türleri için bir LDPC koda ait olan en küçük tuzak kümesi, geliştirdiğimiz tam sayı programlama modeli kullanılarak başarıyla bulunmuştur. En küçük tuzak kümesi bulunduktan sonra tuzak kümesi bir çeşit kaplama metodu ile ortadan kaldırılabilir, [4]. Oluşturulan yeni LDPC kodunun en küçük tuzak kümeleri de aynı yöntemle bulunarak ortadan kaldırabilir. Sonuç olarak, istenilen hata tabanı değerine ulaşılıncaya kadar en küçük tuzak kümeleri yok edilebilir. Ayrıca, kritik sayı hesaba katılarak bu işlem daha verimli bir şekilde uygulanabilir. En küçük kritik sayıya ait küçük boyutlu tuzak kümelerinin bulunması ve yok edilmesi, hata tabanını daha düşük değerlere çekecektir. Bundan sonraki çalışmalar da en küçük kritik sayıya ait en küçük tuzak kümelerinin bulunması yönünde olacaktır.

6. KAYNAKÇA

[1] Gallager, R., "Low-density parity-check codes," *IRE Transactions on Information Theory*, cilt 8, no. 1, ss. 21–28, Ocak 1962.
[2] Richardson, T., "Error floors of LDPC codes," *Proceedings of the annual Allerton conference on communication control and computing*, cilt 41, no. 3, Eylül 2003, ss. 1426–1435.
[3] McGregor, A. ve Milenkovic, O., "On the hardness of approximating stopping and trapping sets," *IEEE Transactions on Information Theory*, cilt 56, no. 4, ss. 1640–1650, Nisan 2010.
[4] Ivkovic, M., Chilappagari, S., ve Vasic, B., "Eliminating trapping sets in low-density parity-check codes by using Tanner graph covers," *IEEE Transactions on Information Theory*, cilt 54, no. 8, ss. 3763–3768, Ağustos 2008.
[5] Wang, C., Kulkarni, S., ve Poor, H., "Finding all small error-prone substructures in LDPC codes," *IEEE Transactions on Information Theory*, cilt 55, no. 5, ss. 1976–1999, Mayıs 2009.
[6] Karimi, M. ve Banihashemi, A., "An efficient algorithm for finding dominant trapping sets of LDPC codes," *Proc. IEEE International Symposium on Turbo Codes and Iterative Information*, Eylül 2010, ss. 444–448.
[7] Abu-Surra, S., DeClercq, D., Divsalar, D., ve Ryan, W., "Trapping set enumerators for specific LDPC codes," *Proc. Information Theory and Applications Workshop*, Şubat 2010, ss. 1–5.
[8] Tanner, R., Sridhara, D., Sridharan, A., Fuja, T., ve Costello Jr, D., "LDPC block and convolutional codes based on circulant matrices," *IEEE Transactions on Information Theory*, cilt 50, no. 12, ss. 2966–2984, 2004.
[9] Bazaraa, M., Jarvis, J., ve Sherali, H., *Linear programming and network flows*, 2nd ed. Wiley, 1990.
[10] Wolsey, L. A., *Integer Programming*. New York, NY: Wiley-Interscience, 1998.