ICONSAD'21

*ICONSAD'21 22-25 December 2021*

# Parçacık Sürü Optimizasyonu Kullanılarak Etkili Bir Lineer Geri Beslemeli Kaydırma Yazmaç Yapısı

## An Efficient Linear Feedback Shift Register Structure Using Particle Swarm Optimization

Eyüp Eröz [*,1], Erkan Tanyıldızı [2], Fatih Özkaynak [2] ( [*] : *sorumlu yazar* )

[*]: eeroz@firat.edu.tr, ORCID: 0000-0003-2670-0606

1Karakoçan Meslek Yüksek Okulu, Fırat Üniversitesi, Elazığ, Türkiye

2Yazılım Mühendisliği Bölümü, Fırat Üniversitesi, Elazığ, Türkiye

*(eeroz@firat.edu.tr, etanyildizi@firat.edu.tr, ozkaynak@firat.edu.tr)*

**Özet** Rasgele bir dizi üretmek birçok uygulamada önemli bir gereksinimdir. Rastgele görünümlü diziler elde edebilmek için en yaygın kullanılan yöntemlerden biri doğrusal geri beslemeli kaydırma yazmaç yapılarıdır. Ancak bu yapılar ile uzun rasgele sayı dizileri elde etmek zor bir görevdir. Çünkü üretilecek dizinin uzunluğu tasarımda kullanılan flip-flop sayısı ile bağlantılıdır. Belirli bir değerden sonra tasarım sürecinde seçilecek konfigürasyonlara karar vermek NP hesaplama karmaşıklığına sahiptir. Bu çalışmada bu hesapsal karmaşıklık problemi bir sezgisel yaklaşım olan parçacık sürü optimizasyon algoritması ile çözülmeye çalışılmıştır. Elde eilen sonuçların başarısı çeşitli istatistiksel testler ile doğrulanmıştır. Başarılı çıktıların ileride birçok pratik uygulamada başarılı olarak kullanılabileceğine inanılmaktadır.

***Anahtar Kelimeler:*** *rasgelelik, doğrusal geri beslemeli kaydırma yazmaç, optimizasyon*

**Abstract** Generating a random sequence is an important requirement in many applications. Linear feedback shift register structures are one of the most widely used methods for generating random-looking arrays. However, obtaining long strings using these structures is a difficult task. Because the length of the string to be generated is related to the number of flip-flops used in the design. Deciding on the configurations to be selected in the design process after a certain value has NP computational complexity. In this study, this computational complexity problem is tried to be solved by particle swarm optimization algorithm, which is a heuristic approach. The success of the obtained results has been confirmed by various statistical tests. It is believed that successful outcomes can be used successfully in many practical applications in the future.

***Keywords:*** Randomness, lineer feedback shift register, optimization

## I. GİRİŞ

Rastgelelik, matematikten tıbba, ekonomiden mühendisliğe, simülasyondan oyunlara birçok araştırmacının çıktılarının başarısını etkileyen önemli bir gereksinim olmuştur. Bu nedenle rastgelelik gereksinimlerinin en verimli şekilde karşılanması temel bir araştırma sorusu olmuştur. Uzun yıllar boyunca birçok farklı Rastgele Sayı Üreticisi (RSÜ) tasarımı önerilmiştir [1, 2]. Bu tasarımlar arasında dikkat çeken çalışmalardan biri Doğrusal Geri Beslemeli Kaydırma Yazmaç (Lineer Feedback Shift Register / LFSR) yapıları olmuştur [3]. LFSR yapısı, rastgelelik gereksinimleri için temel kabul edilen istatistiksel gereksinimleri sağlayabildiği için

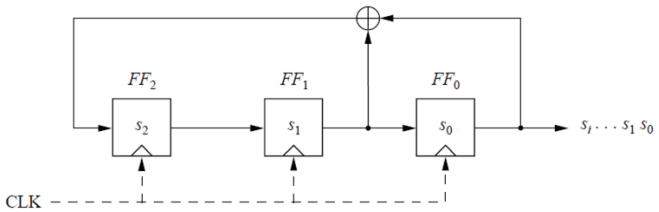
birçok pratik uygulamada ilk tercih edilen oluşturuculardan biri olmuştur. Ancak, bu tasarım önerisinin önemli sorunlarından biri, uzun bit dizileri üretmek için hangi konfigürasyonun kullanılacağını belirlerken ortaya çıkar. Çünkü bit dizisinin uzunluğu, konfigürasyonda kullanılan flip-flop sayısı ile ilgilidir. Flip-flop sayısı arttıkça geri besleme hangi çıktıların kullanılacağı NP (Nondeterministic Polynomial) bir problem haline gelir [5, 6].

Bu çalışmada, NP probleminin çözümü için parçacık sürü optimizasyonu (PSO) algoritmasına [7, 8] dayalı bir yaklaşım önerilmiştir. Çalışmada 1, 2, 3, 4, 5, 6, 7 ve 8 flip/flop içeren tüm LFSR konfigürasyonları için maksimum bit uzunluk konfigürasyon parametreleri PSO kullanılarak belirlenmiştir. Elde edilen çıktıların başarısı, istatistiksel ki-kare testi [9, 10] kullanılarak tüm güven değerleri için doğrulanmıştır.

Çalışmanın yapısı dört bölüm üzerine kurulmuştur. İkinci bölümde LFSR yapıları hakkında genel bilgiler verilmiştir. Üçüncü bölümde, geliştirilen programın arayüzü tanıtılmakta ve istatistiksel test sonuçları paylaşılmaktadır. Son bölümde ise sonuçlar yorumlanmış ve geleceğe yönelik olası planlar verilmiştir.

## II. Doğrusal Geri Beslemeli Kaydırma Yazmacı

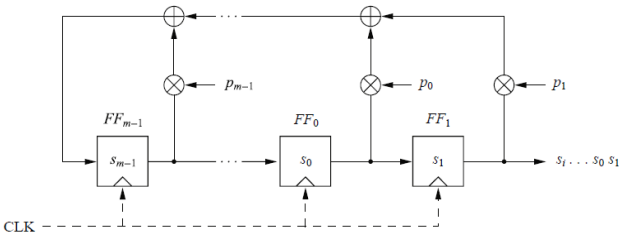
En kullanışlı RNG olan LFSR, flip-flop'lardan (1-bit depolama elemanı) ve geri besleme yolundan oluşur. flip-flop sayısı bize LFSR'nin derecesini verir. Başka bir deyişle, m flip-flop ile oluşan bir LFSR'nin derecesi m'dir [11, 12]. Geri besleme yolu, son flip-flop'un girişidir; belirli flip-flopların XOR toplamının sonucu olarak hesaplar. Şekil 1, basit bir 3 sıralı LFSR yapısını göstermektedir.




**Şekil 1.** Basit bir LFSR örneği

Şekil 1'deki konfigürasyon için 100 başlangıç tohum değerleri kullanılırsa, elde edilecek rastgele bit dizisinin çıkışı 00101110010111010111... olacaktır. Örnek tasarımda 3 flip-flop kullanıldığından, maksimum bit uzunluğu ulaşılabilecek dize $2^3$-1=7 olacaktır. 8. değerden sonra dizi kendini tekrar edecektir [3].

m dereceli bir LFSR'nin genel formu Şekil 2'de gösterilmektedir. Bu, tümü XOR işlemiyle birleştirilmiş m flip-flop ve m olası geri besleme konumunu göstermektedir. Bir geri besleme yolunun aktif olup olmadığı, geri besleme katsayısı $p_0$, $p_1$,…,$p_{(m-1)}$ ile tanımlanır.




**Şekil 2.** LFSR genel gösterimi

- $p_i$=1 (kapalı anahtar) ise geri besleme aktiftir.
- $p_i$=0 (açık anahtar) ise, karşılık gelen flip-flop çıkışı geri besleme için kullanılmaz.

## III. Geliştirilen Yöntem

Geliştirilen programın mümkün olduğunca basit bir arayüze sahip olması istenmiştir. Optimizasyon algoritması sonucunda elde edilen veri seti herkese açık olarak paylaşılır [16]. Araştırmacılar bu dosyaya kişisel bilgisayarlarından indirerek kolayca ulaşabilirler. Veri kümesinin her bir sütunu, 3 ila 8 arasındaki farklı LFSR dereceleri için maksimum uzunluk rastgele dizisini üreten konfigürasyona karşılık gelir. Şekil 3'te veri kümesinin genel görünümü paylaşılmaktadır. Her sınıf için konfigürasyonlar iki aşamadan oluşur. Bunlar, giriş tohum değerleri ve geri besleme konumları olarak ifade edilir.

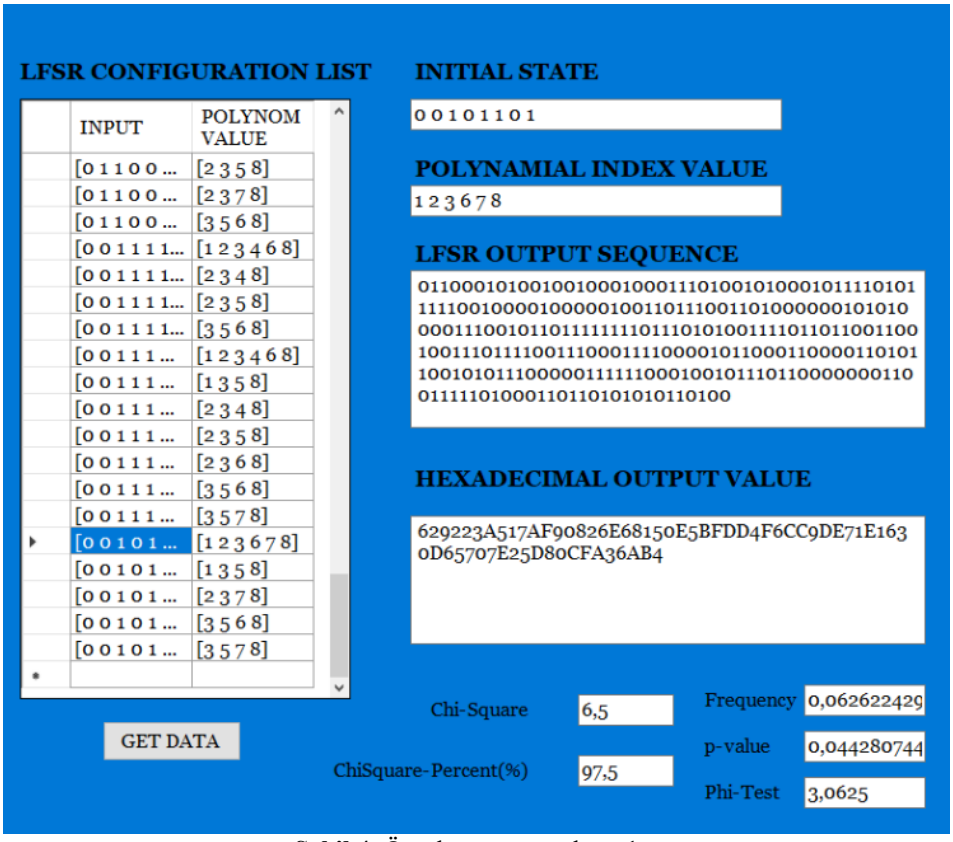
| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | 3 | | 4 | | 5 | | 6 | |
| 2 | input seed values | feedback position | input seed values | feedback position | input seed values | feedback position | input seed values | feedback position |
| 3 | [1 0 1] | [1,3] | [1,0,0,1] | [1,4] | [0 1 0 0 1] | [2,3,4,5] | [1 1 0 1 0 0] | [2,3,5,6] |
| 4 | [1 0 1] | [2,3] | [1,0,0,1] | [3,4] | [0 1 0 0 1] | [2,5] | [1 1 0 1 0 0] | [5,6] |
| 5 | [0 1 1] | [1,3] | [1,0,1,1] | [1,4] | [0 1 0 0 1] | [3,5] | [1 1 0 1 0 0] | [1,2,5,6] |
| 6 | [0 1 1] | [2,3] | [1,0,1,1] | [3,4] | [1 0 1 1 1] | [1,2,3,5] | [1 1 0 1 0 0] | [1,3,4,6] |
| 7 | | | [0,0,1,1] | [1,4] | [1 0 1 1 1] | [1,2,4,5] | [1 1 0 1 0 0] | [1,4,5,6] |
| 8 | | | [0,0,1,1] | [3,4] | [1 0 1 1 1] | [1,3,4,5] | [1 1 0 1 0 0] | [1,6] |
| 9 | | | | | [1 0 1 1 1] | [2,3,4,5] | [0 1 1 1 0 0] | [1,2,5,6] |
| 10 | | | | | [1 0 1 1 1] | [2,5] | [0 1 1 1 0 0] | [1,3,4,6] |
| 11 | | | | | [1 0 1 1 1] | [3,5] | [0 1 1 1 0 0] | [1,4,5,6] |
| 12 | | | | | [0 1 0 0 1] | [1,2,3,5] | [0 1 1 1 0 0] | [1,6] |
| 13 | | | | | [0 1 0 0 1] | [1,2,4,5] | [0 1 1 1 0 0] | [2,3,5,6] |
| 14 | | | | | [0 1 0 0 1] | [1,3,4,5] | [0 1 1 1 0 0] | [5,6] |
| 15 | | | | | [0 0 1 0 1] | [1,2,3,5] | [1 1 0 0 1 0] | [1,2,5,6] |
| 16 | | | | | [0 0 1 0 1] | [1,2,4,5] | [1 1 0 0 1 0] | [1,3,4,6] |
| 17 | | | | | [0 0 1 0 1] | [1,3,4,5] | [1 1 0 0 1 0] | [1,4,5,6] |
| 18 | | | | | [0 0 1 0 1] | [2,3,4,5] | [1 1 0 0 1 0] | [1,6] |
| 19 | | | | | [0 0 1 0 1] | [2,5] | [1 1 0 0 1 0] | [2,3,5,6] |
| 20 | | | | | [0 0 1 0 1] | [3,5] | [1 1 0 0 1 0] | [5,6] |

BBA BPSO +

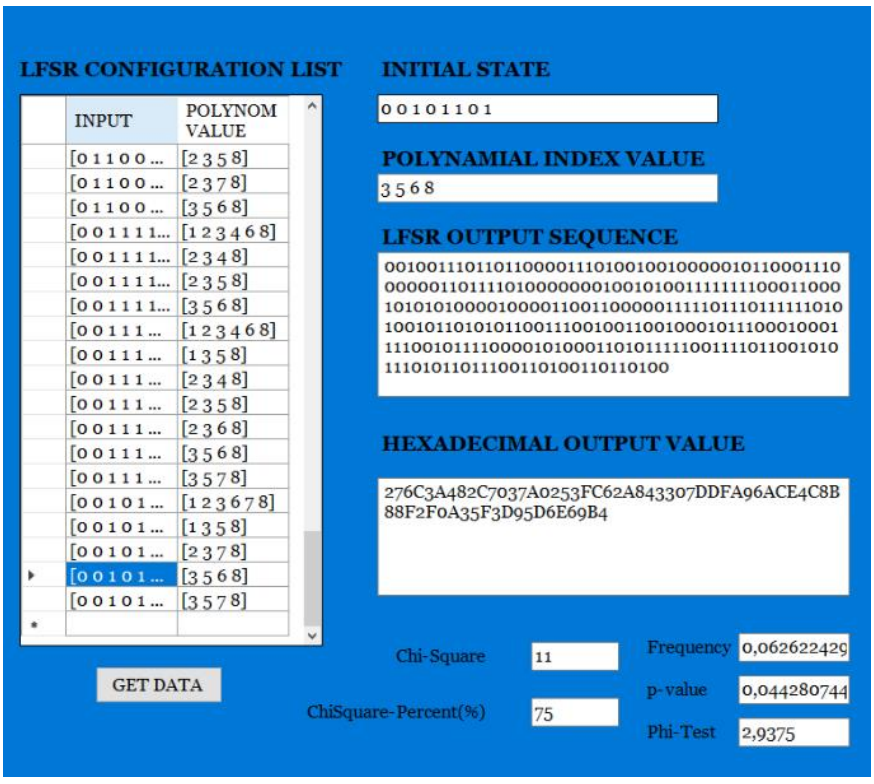
Hedefi belirleyip ENTER tuşuna basın veya Yapıştır'ı seçin Say: 2

**Şekil 3.** LFSR genel gösterimi

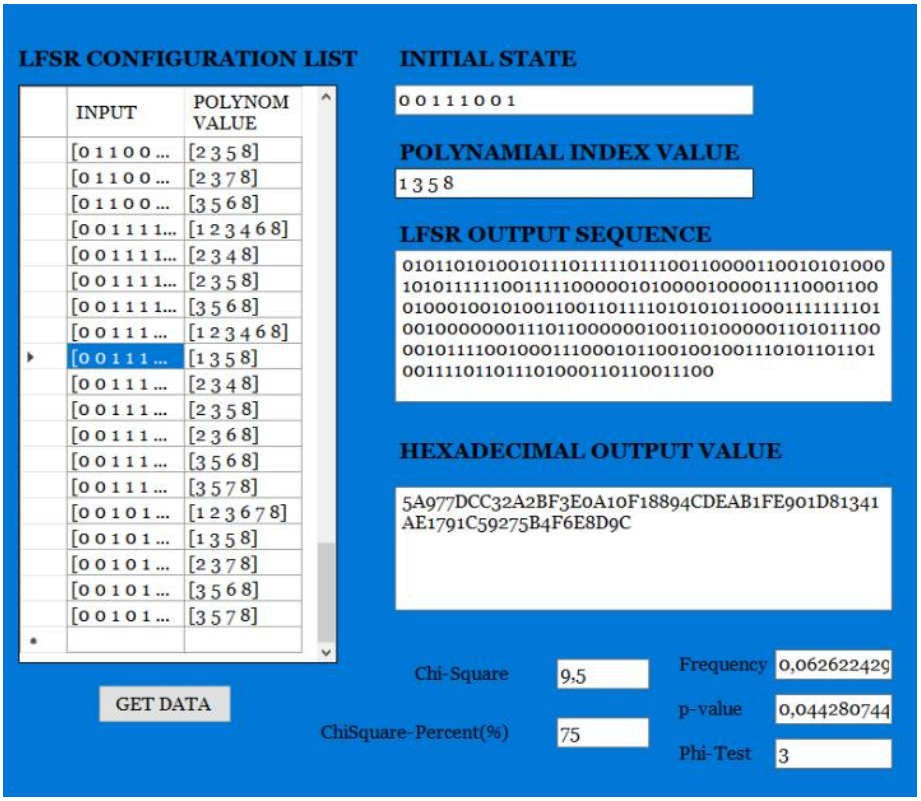
Şekil 4, 5 ve 6'de sırasıyla üç flip-flop kullanılarak LFSR yapıları için PSO kullanılarak elde edilmiş ideal 111 olası konfigürasyondan üçü gösterilmiştir. Diğer konfigürasyonlar programın sol tarafındaki liste kutusunda listelenmiştir. Herhangi bir konfigürasyon seçilerek rastgele bir bit dizisi elde edilebilir. Ayrıca 255 bit uzunluktaki dizileri ve 8 bit uzunluklu LFSR için hesaplanmış ki-kare değerlerini içerir.




**Şekil 4.** Örnek program çıktısı 1




**Şekil 5.** Örnek program çıktısı 2




**Şekil 6.** Örnek program çıktısı 3

Oluşturulan bit dizileri de programda onaltılık dizilere dönüştürülür. Bu dönüşüm, üretilen rastgele değerlerin frekans dağılımını daha kolay incelemeyi mümkün kılmıştır. Bu dönüşüm aynı zamanda ki-kare test sonuçlarının hesaplanmasına da katkıda bulunmuştur. Dönüştürülen değerler 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E ve F arasında değişebilen 16 farklı değerden birine sahiptir. bit dizisi 4 bit'e bölünür, 64 onaltılık değer elde edilir. İdeal olarak, 16 olası seçenek olduğundan, her onaltılık değerden 64/16=4 beklenir. Ki-kare testi, beklenen ve gözlemlenen frekans değerleri arasındaki farkın mümkün olduğunca küçük olmasını bekler. Bir hipotez testi olarak, jeneratörün rastgeleliği güven değerlerine göre değerlendirilir. Şekil 7, ki-kare testi için farklı güven değerlerinin karşılanması gereken değerleri göstermektedir.

| Degree of Freedom | Probability of Exceeding the Critical Value | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 0.99 | 0.95 | 0.90 | 0.75 | 0.50 | 0.25 | 0.10 | 0.05 | 0.01 |
| 1 | 0.000 | 0.004 | 0.016 | 0.102 | 0.455 | 1.32 | 2.71 | 3.84 | 6.63 |
| 2 | 0.020 | 0.103 | 0.211 | 0.575 | 1.386 | 2.77 | 4.61 | 5.99 | 9.21 |
| 3 | 0.115 | 0.352 | 0.584 | 1.212 | 2.366 | 4.11 | 6.25 | 7.81 | 11.34 |
| 4 | 0.297 | 0.711 | 1.064 | 1.923 | 3.357 | 5.39 | 7.78 | 9.49 | 13.28 |
| 5 | 0.554 | 1.145 | 1.610 | 2.675 | 4.351 | 6.63 | 9.24 | 11.07 | 15.09 |
| 6 | 0.872 | 1.635 | 2.204 | 3.455 | 5.348 | 7.84 | 10.64 | 12.59 | 16.81 |
| 7 | 1.239 | 2.167 | 2.833 | 4.255 | 6.346 | 9.04 | 12.02 | 14.07 | 18.48 |
| 8 | 1.647 | 2.733 | 3.490 | 5.071 | 7.344 | 10.22 | 13.36 | 15.51 | 20.09 |
| 9 | 2.088 | 3.325 | 4.168 | 5.899 | 8.343 | 11.39 | 14.68 | 16.92 | 21.67 |
| 10 | 2.558 | 3.940 | 4.865 | 6.737 | 9.342 | 12.55 | 15.99 | 18.31 | 23.21 |
| 11 | 3.053 | 4.575 | 5.578 | 7.584 | 10.341 | 13.70 | 17.28 | 19.68 | 24.72 |
| 12 | 3.571 | 5.226 | 6.304 | 8.438 | 11.340 | 14.85 | 18.55 | 21.03 | 26.22 |
| 13 | 4.107 | 5.892 | 7.042 | 9.299 | 12.340 | 15.98 | 19.81 | 22.36 | 27.69 |
| 14 | 4.660 | 6.571 | 7.790 | 10.165 | 13.339 | 17.12 | 21.06 | 23.68 | 29.14 |
| 15 | 5.229 | 7.261 | 8.547 | 11.037 | 14.339 | 18.25 | 22.31 | 25.00 | 30.58 |
| 16 | 5.812 | 7.962 | 9.312 | 11.912 | 15.338 | 19.37 | 23.54 | 26.30 | 32.00 |
| 17 | 6.408 | 8.672 | 10.085 | 12.792 | 16.338 | 20.49 | 24.77 | 27.59 | 33.41 |
| 18 | 7.015 | 9.390 | 10.865 | 13.675 | 17.338 | 21.60 | 25.99 | 28.87 | 34.80 |
| 19 | 7.633 | 10.117 | 11.651 | 14.562 | 18.338 | 22.72 | 27.20 | 30.14 | 36.19 |
| 20 | 8.260 | 10.851 | 12.443 | 15.452 | 19.337 | 23.83 | 28.41 | 31.41 | 37.57 |
| 22 | 9.542 | 12.338 | 14.041 | 17.240 | 21.337 | 26.04 | 30.81 | 33.92 | 40.29 |
| 24 | 10.856 | 13.848 | 15.659 | 19.037 | 23.337 | 28.24 | 33.20 | 36.42 | 42.98 |
| 26 | 12.198 | 15.379 | 17.292 | 20.843 | 25.336 | 30.43 | 35.56 | 38.89 | 45.64 |
| 28 | 13.565 | 16.928 | 18.939 | 22.657 | 27.336 | 32.62 | 37.92 | 41.34 | 48.28 |
| 30 | 14.953 | 18.493 | 20.599 | 24.478 | 29.336 | 34.80 | 40.26 | 43.77 | 50.89 |
| 40 | 22.164 | 26.509 | 29.051 | 33.660 | 39.335 | 45.62 | 51.80 | 55.76 | 63.69 |
| 50 | 27.707 | 34.764 | 37.689 | 42.942 | 49.335 | 56.33 | 63.17 | 67.50 | 76.15 |
| 60 | 37.485 | 43.188 | 46.459 | 52.294 | 59.335 | 66.98 | 74.40 | 79.08 | 88.38 |
| | *Not Significant* | | | | | | | *Significant* | |

**Şekil 7.** Ki-kare testi için güven aralıkları

## SONUÇ

Diğer birçok disiplinin bilgisayar bilimcilerinden beklediği en önemli gereksinimlerden biri, rastgelelik gereksinimlerini etkin bir şekilde karşılayabilecek bir yaklaşım geliştirmektir. Ancak bu gereksinimleri karşılayabilmek bir meydan okumadır. Bu zorluğun sıcak bir örneği, uzun bit dizileri oluşturmak için bir rastgelelik kaynağı oluşturmak için birçok çalışmanın temel aldığı LFSR yapılarını kullanmaktır. Uzun bit dizileri daha fazla flip-flop gerektirdiğinden, klasik yöntemler hangi flip-flop yapılarının geri besleneceğini ve başlangıç tohum değerlerinin ne olacağını belirleyemez. Çünkü olasılıkların sayısı çoktur ve sonlu bir zamanda elde edilemez. Başka bir deyişle, problemin hesaplama karmaşıklığı bir NP problemidir.

Bu çalışmada, parçacık sürü optimizasyon algoritması kullanılarak sekiz dereceye kadar tüm doğrusal geri beslemeli kaydırmalı yazmaç yapıları için kullanılabilecek tüm olası konfigürasyonlar belirlenmiştir. Bu konfigürasyonlar kullanılarak oluşturulan rastgele dizilerin istatistiksel özellikleri, ki-kare testi kullanılarak test edilmiştir. Ortaya çıkan tüm yapılandırmalar, genel bir veri kümesi olarak paylaşılır. Bu başarılı sonuçların gelecekte birçok araştırmacı tarafından çeşitli pratik uygulamalarda etkin bir şekilde kullanılabileceği düşünülmektedir. Konfigürasyon parametrelerinin kaosa dayalı seçimi, gelecekteki çalışmalarda olası çalışmalardan biri olarak kabul edilmektedir [17, 18]. Bu çalışmalar, kaos tabanlı kriptoloji çalışmalarında [20] optimizasyon algoritmaları [19] ile geliştirilmiş güçlü bir kriptolojik bileşen sağlayacaktır. Gelecekte, farklı istatistiksel test yaklaşımları kullanılarak, farklı bakış açılarından önerilen yaklaşım [21]. Ayrıca uygulamanın görüntü şifreleme gibi olası pratik uygulamalar içerisinde değerlendirilmesi planlanmaktadır [22].

KAYNAKLAR


[1]. [1 M. Robshaw and O. Billet, editors. New Stream Cipher Designs: The eSTREAM Finalists, volume 4986 of LNCS. Springer, 2008.

[2]. M., Garipcan, E. Erdem, Implementation and Performance Analysis of True Random Number Generator on FPGA Environment by Using Non-periodic Chaotic Signals Obtained from Chaotic Maps. Arab J Sci Eng 44, 9427–9441 (2019). https://doi.org/10.1007/s13369-019-04027-x

[3]. A. M., Garipcan, E. Erdem, A TRNG using chaotic entropy pool as a post-processing technique: analysis, design and FPGA implementation. Analog Integr Circ Sig Process 103, 391–410 (2020). https://doi.org/10.1007/s10470-020-01605-0

[4]. C. Paar, J. Pelzl, Understanding Cryptography A Textbook for Students and Practitioners, 2010, DOI 10.1007/978-3-642-04101-3, Springer Heidelberg Dordrecht London New York

[5]. M. Sheveleva and S. A. Belyaev, "Development of the Software for Solving the Knapsack Problem by Solving the Traveling Salesman Problem," 2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus), 2021, pp. 652-656, doi: 10.1109/ElConRus51938.2021.9396448.

[6]. M. Asif and R. Baig, "Solving NP-complete problem using ACO algorithm," 2009 International Conference on Emerging Technologies, 2009, pp. 13-16, doi: 10.1109/ICET.2009.5353209.

[7]. S. Maheswari and K. Arunesh, "Unsupervised Binary BAT algorithm based Network Intrusion Detection System using enhanced multiple classifiers," 2020 International Conference on Smart Electronics and Communication (ICOSEC), 2020, pp. 885-889, doi: 10.1109/ICOSEC49089.2020.9215453.

[8]. H. Tufail, K. Zafar and R. Baig, "Digital Watermarking for Relational Database Security Using mRMR Based Binary Bat Algorithm," 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), 2018, pp. 1948-1954, doi: 10.1109/TrustCom/BigDataSE.2018.00298.

[9]. M. Göl and A. Abur, "A modified Chi-Squares test for improved bad data detection," 2015 IEEE Eindhoven PowerTech, 2015, pp. 1-5, doi: 10.1109/PTC.2015.7232283

[10]. F. Nielsen and R. Nock, "On the chi square and higher-order chi distances for approximating f-divergences," in IEEE Signal Processing Letters, vol. 21, no. 1, pp. 10-13, Jan. 2014, doi: 10.1109/LSP.2013.2288355.

[11]. W. Schindler, "Random number generators for cryptographic applications", C .K. Koc (ed.): Cryptographic Engineering. Springer, Signals and Communication Theory, Berlin, 2009. DOI: 10.1007/978-0-387-71817-0_2.

[12]. M. Stipčević and Ç. K. Koç, "True random number generators", in Koç Ç. K. (eds) Open Problems in Mathematics and Computational Science. Springer, Cham, 2014. DOI: 10.1007/978-3-319-10683- 0_12.

[13]. J. Kennedy, R. C. Eberhart (1997) A discrete binary version of the particle swarm algorithm. In: IEEE international conference on computational cybernetics and simulation, pp 4104–4108

[14]. E. Rashedi, H. Nezamabadi-pour, S. Saryazdi (2009) BGSA: binary gravitational search algorithm. Nat Comput 9:727–745

[15]. S. Mirjalili, S.M. Mirjalili, X.S. Yang, Binary bat algorithm, Neural Comput. Appl. 25 (2014) 663–681.

[16]. https://github.com/eyperz

[17]. F. Özkaynak, Cryptographically secure random number generator with chaotic additional input. Nonlinear Dyn 78, 2015–2020 (2014). https://doi.org/10.1007/s11071-014-1591-y

[18]. F. Özkaynak, (2020). A Novel Random Number Generator Based on Fractional Order Chaotic Chua System. Elektronika Ir Elektrotechnika, 26(1), 52-57. https://doi.org/10.5755/j01.eie.26.1.25310

[19]. E. Tanyildizi and F. Özkaynak, "A New Chaotic S-Box Generation Method Using Parameter Optimization of One Dimensional Chaotic Maps," in IEEE Access, vol. 7, pp. 117829-117838, 2019, doi: 10.1109/ACCESS.2019.2936447.

[20]. L. Kocarev and S. Lian, Chaos-Based Cryptography: Theory Algorithms and Applications, Cham, Switzerland:Springer, 2011.

[21]. A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, and S. Vo, "A statistical test suite for random and pseudorandom number generators for cryptographic applications", NIST Special Publication, Report no. 800–22rev1a, 2010.

[22]. A. S. Muhammad, F. Özkaynak, "SIEA: Secure Image Encryption Algorithm Based on Chaotic Systems Optimization Algorithms and PUFs", Symmetry 2021.