Journal of the Faculty of Engineering and Architecture of Gazi University 36:2 (2021) 1013-1027

Mühendislik Mimarlık Fakültesi Dergisi
Journal of The Faculty of Engineering and Architecture of Gazi University

Elektronik / Online ISSN : 1304 - 4915
Basılı / Printed ISSN : 1300 - 1884

Finding the optimum parameter values of the round robin CPU scheduling algorithm with genetic algorithm

Selçuk Ökdem[1], Betül Koşmaz[2]*

[1]Erciyes University, Engineering Faculty, Computer Engineering Department, 38000, Kayseri, Turkey
[2]Gümüşhane University, Faculty of Engineering and Natural Sciences, Mathematical Engineering Department, 29100, Gümüşhane, Turkey

Highlights:

- Optimization Methods
- Scheduling Algorithms
- Operating Systems

Keywords:

- Round robin
- Genetic algorithm
- Optimization
- Cpu scheduling
- Operating systems

Article Info:

Research Article
Received: 09.09.2019
Accepted: 07.12.2020

DOI:

10.17341/gazimmfd.617418

Acknowledgement:

Part of this study is derived from the master's thesis titled "Finding Optimum Parameter Values of Rotary Core CPU Scheduling Algorithm with Genetic Algorithm" by one of the article authors, Betül Koşmaz

Correspondence:

Author: Betul Koşmaz
e-mail: betul.kosmaz@gumushane.edu.tr
phone: +90 456 233 1068

Graphical/Tabular Abstract

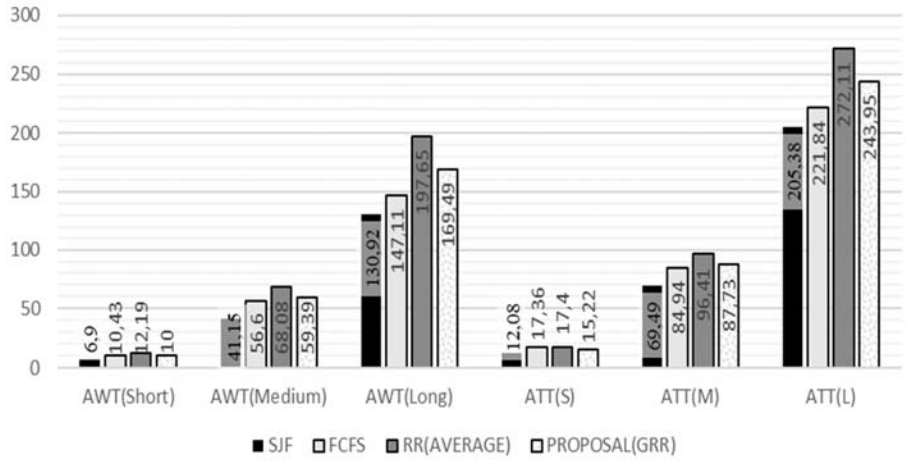


Figure A. Average waiting and turnaround times of standard scheduling algorithms and GRR algorithm.

Purpose: This article is aimed to obtain an algorithm that yields average waiting time in shorter periods by calculating the optimum value for the quantum time parameter. This parameter is used for Round Robin (RR) scheduling, which is among the preemptive scheduling algorithms.

Theory and Methods:
Finding the best value for quantum time is the main problem with RR. If the quantum time is kept too short, the CPU has to perform more switching operations and spends much more time between continuous processes. It causes busy the CPU unnecessarily, resulting in the loss of actual service time. Assigning too big values to quantum, causes the waiting time to have big values, while other processes stay in the queue for long times.
In this paper, Genetic Algorithm (GA), which is a widely used optimization method, was preferred as the optimization method. The software has been developed in C# and Octave for RR and GA to work together. Using these software, four methods were written for RR, roulette wheel selection, crossover and mutation.

Results:
In this paper, GRR(Genetic Round Robin) and GRRS(Genetic Round Robin with Sorting) were developed, and comparisons were made with proposed algorithms and other standard tariff algorithms. The proposed algorithms produce superior performance than Shortest Job First (SJF) and RR. Better outputs are obtained than RR and the recent proposals in general. The results vary according to the methods and the lengths of the processes.

Conclusion:
As a result of detailed comparisons, it was found that GRR and GRRS achieve the best results in most of the cases. This is due to the fact that traditional algorithms do not always guarantee to achieve the best results according to changing burst lengths of processes.

Journal of the Faculty of Engineering and Architecture of Gazi University 36:2 (2021) 1013-1027

Mühendislik Mimarlık Fakültesi Dergisi
Journal of The Faculty of Engineering and Architecture of Gazi University

Elektronik / Online ISSN : 1304 - 4915
Basılı / Printed ISSN : 1300 - 1884

Döner çekirdek CPU tarifeleme algoritmasının optimum parametre değerlerinin genetik algoritma ile bulunması

Selçuk Ökdem[1], Betül Koşmaz[2]*

[1]Erciyes Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, 38000, Kayseri, Türkiye
[2]Gümüşhane Üniversitesi, Mühendislik ve Doğa Bilimleri Fakültesi, Matematik Mühendisliği Bölümü, 29100, Gümüşhane, Türkiye

Ö N E Ç I K A N L A R

- Optimizasyon yöntemleri
- Tarifeleme algoritmalarının optimizasyonu
- İşletim sistemleri

Makale Bilgileri

Araştırma Makalesi
Geliş: 09.09.2019
Kabul: 07.12.2020

DOI:
10.17341/gazimmfd.617418

Anahtar Kelimeler:
Döner çekirdek,
genetik algoritma,
işlemci tarifeleme,
işletim sistemi,
optimizasyon

ÖZET

Bilgisayarların günümüzde her alanda kullanılmaya başlanmasıyla işlemcilere düşen yoğunluk da gün geçtikçe artmaktadır. Bu yoğunlukla baş edebilmek adına işlemcilerin sürekli olarak güncellenerek daha iyi hale getirilmesi gerekmektedir. İşlemcilerin sürekli geliştirilip hızlandırılmasına ve işlem yoğunluğu ile başa çıkabilmesi adına her geçen gün çeşitli tarifeleme algoritmaları geliştirilmeye devam edilmektedir. Tarifeleme algoritmalarının temel amacı işlemcinin verimliliğini arttırmaktır. Bunun için işlemcinin bekleyen işlemler arasında bir planlama yapması gerekmektedir. İşlemlere farklı öncelikler tanıyarak veya farklı yöntemler kullanarak bir planlama yapılabilmektedir. İşlemcinin verimliliğini arttırabilmek amacıyla bu makalede, Döner Çekirdek Algoritması, Genetik Algoritma ile geliştirilmiştir. Kuyrukta bekleyen işlemler ve onlara ait parametrelere uygun kuantum süresi seçilerek optimum değerlerin bulunması hedeflenmiştir. Geliştirilen yeni yöntem, var olan standart algoritmalar ile kıyaslandığında, bekleme zamanının ve dönüş süresinin diğer algoritmalardan daha kısa olduğu gözlenmiştir.

Finding the optimum parameter values of the round robin CPU scheduling algorithm with genetic algorithm

H I G H L I G H T S

- Optimization methods
- Optimization of scheduling algorithms
- Operating systems

Article Info

Research Article
Received: 09.09.2019
Accepted: 07.12.2020

DOI:
10.17341/gazimmfd.617418

Keywords:
RR,
genetic algorithm,
CPU scheduling,
Operating systems,
optimization

ABSTRACT

With the use of computers in every field, the usage intensity of processors is increasing day by day. In order to cope with this intensity, processors need to be constantly updated and improved. Various scheduling algorithms are being developed every day in order to continuously improve and accelerate the processors and to cope with the processing intensity. The main purpose of the scheduling algorithms is to increase the efficiency of the processor. For this, the processor needs to make a planning between the pending processes. Planning can be done by giving different priorities or using different methods. In order to increase the efficiency of the processor, in this paper, existing algorithms were examined. Round Robin Algorithm was improved using Genetic Algorithm. It is aimed to find the optimum values by selecting the quantum value according to the pending processes and their parameters. When the developed new method compared with the existing standard algorithms, it is observed that the waiting time and the turnaround time were shorter than the other algorithms.

*Sorumlu Yazar/Yazarlar / *Corresponding Author/Authors: okdem@erciyes.edu.tr, *betul.kosmaz@gumushane.edu.tr / Tel: +90 456 233 1068

1. GİRİŞ (INTRODUCTION)

İşlemcide çalıştırılan, herhangi bir dilde yazılıp derlenmiş programlara proses (işlem-süreç) ismi verilir. İşlemcilerin verimliliği için birçok işletim sistemi çoklu işlemi destekler. Elimizde çalıştırmak istediğimiz proses sayısından az işlemci varsa burada devreye Merkezi İşlemci Birimi (Central Processing Unit, CPU) çizelgeleme girer. İşletim sistemlerinde birden fazla hazır durumda proses bulunduğunda bu prosesler işlemciyi kullanmak için birbirleriyle yarışmaya başlarlar, işte böyle durumlarda işlemcinin bir çizelgeleme yani işlemler arasında planlama yapması gerekecektir. Bu planlamayı yapmak için kullanılan algoritmalara işlemci çizelgeleme (zamanlama/planlama) algoritmaları denir. Her algoritma kendine ait yöntemlerle proseslere farklı süreler ya da öncelikler tanıyıp prosesleri planlayarak işlemci verimliliğini arttırmayı amaçlar.

Genetik Algoritma, evrimsel hesaplama tekniğinin bir parçası olup Darwin'in evrim teorisinden esinlenerek oluşturulmuş bir optimizasyon algoritmasıdır. Doğal seleksiyon prensibinden yola çıkarak geliştirilen bir arama algoritmasıdır. Sezgisel bir algoritmadır ve her zaman küresel optimum sonucu üretmeyebilir fakat diğer algoritmaların çözemediği ya da uzun zamanlarda çözdüğü problemleri daha kısa sürelerde çözerek optimuma yakın çözümler üretebilir.

Çizelgeleme işleminin geliştirilmesi ve hızlandırılması amacıyla birçok çalışma yapılmıştır. Örneğin B. Alam [1], Bulanık RR (Round Robin, Döner Çekirdek) algoritmasını önermiştir. Bu algoritma ile kuantum süresinin büyüklüğünün yol açtığı sorunlar giderilmeye çalışılmıştır. Ortalama bekleme süresinin bu yöntem sonucunda azaldığı ve belirli bir zaman biriminde tamamlanan proses sayısının da arttığı görülmüştür [1]. Kumarsaroj vd. [2], 2016 yılında yaptıkları bir çalışma ile proses sürelerinin ortalama farkına dayalı ve değişken kuantum süresine sahip bir yöntem geliştirmişlerdir. Daha önce yapılan benzer çalışmalarda sadece belli başlı parametrelerin iyileştirilmesi üzerinde durulduğu için tüm parametrelere odaklanılmamıştır. Bu yöntemi geliştirmekteki amaçları, dikkate alınmayan parametreleri göz önünde bulundurarak optimize etmek ve açlık sorununu da azaltarak daha iyi sonuçlar elde etmektir [2]. Parekh ve Chaudhari yaptıkları çalışma [3] ile Döner Çekirdek (Round Robin, RR), İlk Gelen İlk Servis (First Come First Served, FCFS), En Kısa İlk (Shoretest Job First, SJF) algoritmalarının üçünü birleştiren bir algoritma önermişlerdir. Bu yöntemde, her işleme verilen öncelik sayılarına göre kuantum süreleri belirlenerek işlemler yürütülmektedir. Aynı zamanda kalan süresi en kısa olan işlemler ön işlemlerden geçirilmeden işleme alınarak algoritmanın üretkenliğini de sağlamışlardır [3].

Neshat vd., "I/O servis zamanı" ve "proseslerin önceliği" parametrelerini de kullanarak Fonseca ve Fleming'in genetik algoritmasıyla RR algoritmasını optimize etmişlerdir [4, 5]. Kullanılan yöntem sonucunda, proseslerin önceliği zamanla artmış ve hiçbir proses açlık problemi ile karşılaşmamıştır. Diğer standart planlama algoritmalarıyla kıyaslandığında, algoritmanın ortalama bekleme ve tepki süresinin azaldığı görülmüştür. Bu yöntemin en önemli özelliği prosesler arasında adaletsizlik olmasıdır. Banerjee vd. [6], RR algoritmasında kullanılan kuantum süresinin hesabında hazır kuyruğunda bekleyen proseslerin işlem zamanlarının ortalamasını kullanmışlardır. Dhumal vd. [7], bu sürenin nasıl seçileceği ile ilgili olarak üç farklı yöntem denemişlerdir. Kuantum süresinin sabit olarak kaldığı, dinamik olarak belirlendiği ve GA ile belirlendiği üç farklı yöntemi kıyaslamışlardır. GA destekli dinamik kuantum değerli RR algoritmasının diğerlerine göre daha iyi performans gösterdiğini ortaya koymuşlardır. Hussein ve Hasoon [8], çok işlemcili sistemlerde işlem sürelerini iyileştirmek amacıyla GA'dan faydalanmışlardır. Önerdikleri algoritma ile maksimum verim almayı ve işlemcinin açlık problemlerini çözmeyi amaçlamışlardır. Siregar [9], Genetik Algoritma (GA) yardımıyla en kısa bekleme süresini sağlayan, optimum kuantum süresini elde etmeye çalışmıştır. Kuantum süreleri birey olarak kabul edilip genetik algoritmaya uygulanmıştır [9]. Elde edilen veriler ışığında, GA ile entegre edilen RR algoritmasının Ortalama Bekleme Zamanı (Average Waiting Time, AWT) bakımından RR'den daha iyi sonuçlar verdiği görülmüştür. Bu sonucun yanı sıra genetik algoritmanın değişken çözüm arama yapısından dolayı, önerilen yöntemin bazı durumlarda çalışmadığı görülmüştür. Bu sebeple önerilen yöntemin geliştirilmesi gerektiği ortaya konulmuştur. Bu makalede önerilen yaklaşımda optimizasyon yöntemi olarak GA kullanılmıştır. GA, gelişime dayalı algoritmalar içinde yaygın olarak kullanılan yapısı ve gerçek zamanlı uygulamalarda gömülü sistem programcılarının kullanılabileceği kolay erişimli GA kütüphane/sürücü yazılımlarının çokluğu nedeniyle tercih edilmiştir.

Bu makalede kesintili algoritmalar arasında yer alan RR için kuantum zamanının optimum değerleri genetik algoritma ile hesaplanmış ve AWT değerlerini daha iyi seviyelerde tutan Genetik tabanlı Döner Çekirdek (Genetic based Round Robin, GRR) ve Sıralamalı Genetik tabanlı Döner Çekirdek (Genetic based Round Robin with Sorting, GRRS) algoritmaları geliştirilmiştir. GRRS algoritması GRR algoritmasının iyileştirilmiş bir türevidir. Sonuçlar, standart yöntemlerle ve bilinen güncel algoritmalarla karşılaştırılmıştır. Bu karşılaştırmalar sonucunda önerilen algoritmaların başarılı sonuçlar elde ettiği görülmüştür. Makalenin devamında 2. Bölümde RR algoritmasından ve GA'dan bahsedilmiştir. 3. Bölümde sonuçlara ve tartışmaya yer verilmiştir. Son bölümde elde edilen bulgular özetlenerek makale sonlandırılmıştır.

2. TEORİK METOT(THEORETICAL METHOD)

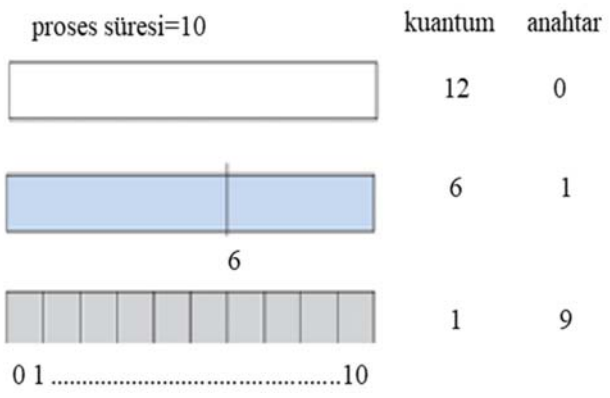
Bu çalışmada çizelgeleme algoritmalarından RR algoritması temel alınmıştır. Bu algoritmaya ait temel giriş verileri olarak proses bilgileri kullanılmıştır. Bu bilgiler proseslerin işlem zamanlarını (burst lengths) içermektedir. RR

algoritmasına ait çıkış parametreleri ise Bekleme Zamanı (Waiting Time, WT), Ortalama Bekleme Zamanı (Average Waiting Time, AWT), Dönüş Zamanı (Turnaround Time, TT), Ortalama Dönüş Zamanı (Average Turnaround Time, ATT), Yanıt Zamanı (Response Time, RT) ve Ortalama Yanıt Zamanı (Average Response Time, ART) parametreleridir. Optimizasyon yöntemi olarak, yaygın kullanılan bir optimizasyon yöntemi olan GA tercih edilmiştir. GA, birden çok çözüm olduğunda ve en iyi çözümün bilinmediği durumlarda en iyiye yakın bir çözümü üretebilme potansiyeline sahiptir [10]. RR algoritması düşünüldüğünde, proseslerin sürelerine göre, en kısa bekleme süresini elde edebilmek için gerekli en iyi kuantum süresi, girdi değişken olarak düşünülebilir. En kısa bekleme süresini elde ederken çeşitli parametreler kötüleşebilir. İşin içerisinde farklı parametrelerin etkilerinin olduğu, bu gibi çok parametreli problemlerde GA başarılı sonuçlara ulaştığı için çoğunlukla tercih edilmektedir.

2.1.Döner Çekirdek Algoritması (Round Robin (RR) algorithm)

RR, basit ve kullanışlı algoritmalardan biridir. Her proses, CPU tarafından, kuantum süresi adı verilen zaman dilimi kadar işleme alınır. Kesintili bir çizelgeleme algoritmasıdır. Kuantum süresi tamamlandığında, proses sona ermediyse, yarıda kesilir ve kuyruğun sonuna atılır. CPU başka bir prosese verilir. Sıradaki proses de kuantum süresi kadar çalıştırılarak devam eder. Kuyruğun sonuna atılan proses ise tekrar sırasını bekler. Eğer prosesin kalan işlem süresi, kuantum süresinden kısaysa ve kuantum süresi tamamlanmadan sona ererse, CPU kuantum süresinin tamamlanmasını beklemez. Böylece işleme alınma hakkı sıradaki prosese geçer.

Bu algoritma ile ilgili tek problem kuantum süresinin uzunluğudur. Kuantum süresi çok kısa tutulursa CPU, sürekli prosesler arasında anahtarlama işlemi yapmak zorunda kalır [11]. Bu da CPU'nun gereksiz yere meşgul tutulup, vakit kaybetmesiyle sonuçlanır. Şekil 1'de kuantum süresinin, anahtarlama sayısına olan etkisi gösterilmiştir. Kuantum süresi azaldıkça anahtarlama sayısının arttığı görülmektedir [12].



Şekil 1. Kısa kuantum süresinin anahtarlama sayısına etkisi (Effect of short quantum time on context switch) [12]

Kuantum süresinin çok uzun olması, anahtarlama işlemiyle kaybedilecek zamanın önüne geçerken, diğer proseslerin uzun süre kuyrukta beklemesine sebep olur. Bu da hızlı işlemesi gereken sistemlerde yavaşlamaya yol açar. Yanıt zamanının düşük tutulması özellikle Endüstri 4.0 uygulamaları başta olmak üzere birçok işletimlerde önemlidir [13]. Bir çalışma esnasında ortalama dönüş süresi diye belirttiğimiz ATT, kuantum süresi arttıkça kötüleşmektedir. ATT'nin iyileşmesi için proseslerin işlemlerini tek kuantum süresinde bitirmesi gerekebilir [12]. Kuantum süresinin artmasıyla RR algoritması'nın çalışma mantığı FCFS algoritmasının çalışma mantığına benzemeye başlar ve RR'nin sunduğu farklılıklar ortadan kalkar.

2.2. Genetik Algoritma (Genetic Algorithm, GA)

Genetik algoritma (GA), Holland vd. tarafından geliştirilerek çalışmaların sonucunda "Adaptation in Natural and Artificial Systems" isimli kitapta yayınlanmıştır [14]. Bu çalışmada geliştirilen yaklaşım, araştırma uzayında bulunan çözümlerden bazılarının bulunduğu bir başlangıç popülasyonu kullanır. GA, bu başlangıç popülasyonuna doğal seleksiyon ve tekrar üreme işlemi uygulayarak en son kuşaktaki optimum bireyi çözüm olarak bulur. Her zaman en iyi çözüm bulunamasa da optimuma en yakın çözüm bulunur.

Genetik algoritmanın temel beş ögesi bulunur. Bu ögeler aslında algoritmanın çalışmasını biçimlendiren elemanlardır.

Çözümleri temsil etmek için farklı yöntemler kullanılmaktadır. İkili kodlama, gri kodlama, ağaç kodlama, gerçek sayılı kodlama, permütasyon kodlama bunlar arasındadır. Bazı problemlerde diğer kodlama yöntemlerini kullanmak algoritmanın işlevini azaltabilir ya da yok edebilir. Bu sebeple bu çalışmada gerçek sayılı temsil yöntemi tercih edilmiştir.

Araştırma ve uygulama problemlerinde tercih edilen popülasyon oluşturma yöntemleri farklılık göstermektedir. Araştırma amacıyla yapılan problemlerde rasgele sayı üretilerek popülasyon oluşturulurken, uygulama amaçlı mühendislik problemlerinde bilinen bazı çözümler kullanılarak başlangıç popülasyonu oluşturulur. Bu çalışma için rasgele kuantum süreleri belirlenerek başlangıç popülasyonu oluşturulmuştur.

Kontrol parametreleri de genetik operatörler gibi algoritmanın çalışmasına önemli etkileri olan parametrelerdir [10]. Bir genetik algoritmanın temel parametreleri popülasyon büyüklüğü, çaprazlama oranı ve mutasyon oranıdır.

Kullanılan genetik operatörler, algoritmadaki çözümlerin daha kaliteli sonuçlara dönüşmesine katkıda bulunur. Kullanılan temel genetik operatörler tekrar üreme, çaprazlama ve mutasyon operatörleridir. Bu operatörler problem tipine ve temsil yöntemine göre değişiklik göstermektedir. Evrimsel mekanizmada doğal seçilim olarak

adlandırılan olayın, genetik algoritmaya uygulanmasıdır. Bu olay, elverişli bireylerin popülasyon içerisinde baskın hale gelmesine ve zayıf bireylerin ise kaybolmasına sebep olacaktır. Bu makalede tekrar üreme operatörü olarak rulet teker yöntemi seçilmiştir. Bu teknikte her bir çözüme, o çözümün kalite değerinin popülasyonun ortalama kalite değerine oranına göre, tekerlek üzerinde bir dilim verilir. Burada i. dizinin seçilme olasılığı Eş. 1'de verildiği üzere hesaplanır. Eşitlikte yer alan *i* seçilen çözüme ait sıra numarasıdır, *j* sayaç olarak belirlenmiştir. *N* popülasyon büyüklüğüdür. Rulet teker *N* defa döndürülür ve her seferinde bir çözüm seçilir.

$$f(i)/\sum_{j=1}^{N} \mathrm{f(j)} \quad (1)$$

Tekerlek yeterli sayıda döndürülür ve top hangi bireye ait kısma girerse, o aday sonraki nesil için seçilir. Kalitesi yüksek adayların oranı büyük olacağından tekerlek üzerindeki payı da büyük olur, böylece seçilme şansı da artmış olur.

Genetik çaprazlama olayı iki bireyin genlerini, genetik çaprazlama yoluyla, yeni oluşan bireye aktarmasıdır. Çaprazlama yapmak için popülasyon içerisinden iki birey seçilir ve çaprazlama operatörleri yardımıyla yeni iki birey meydana getirilir. Bu çalışma kapsamında aritmetik çaprazlama yöntemi kullanılmıştır. Bu yöntem gerçek sayılı temsiller için kullanılan bir yöntemdir [15]. Eş. 2 ve Eş. 3'de görüleceği üzere, bu yöntemde C ve D aday çözümleri A ve B'nin çaprazlanması ile hesaplanır. α değeri ise (0,1) tekdüze dağılımından rasgele olarak seçilmiştir. *α* değeri, ebeveynlerin yeni oluşacak bireyler üzerindeki etkisini belirlemekle görevlidir. Buradan gelecek değere göre hangi ebeveynden daha çok aktarım yapılacağına karar verilmiş olur.

$$C = \alpha A + (1 - \alpha) B \quad (2)$$

$$D = (1 - \alpha) A + \alpha B \quad (3)$$

Çözümlerin optimum derecesini ölçmek için kalite değerlendirilmesinin yapılmasına ihtiyaç duyulur [10]. Çözümün uygunluk değeri arttıkça bu çözümün neslini devam ettirme şansıda yüksek olacaktır. Bu uygunluk değeri amaç fonksiyonu ile orantılıdır. Çalışmamızda uygunluk fonksiyonu olarak Eş. 4 ve Eş. 5 kullanılmıştır. Eşitlikte *f* uygunluk değerini, *AWT* ortalama bekleme zamanını, *ATT* ise ortalama dönüş zamanını ifade etmektedir.

$$f_{GRR} = AWT * ATT \quad (4)$$

$$f_{GRRS} = 1/(1 + AWT) \quad (5)$$

Genetik algoritma durdurma kriteri sağlanıncaya kadar devam eden adımlardan meydana gelmektedir. Basit bir genetik algoritmanın adımları şu şekildedir:

Adım 1. Çözümleri içeren başlangıç popülasyonunu oluştur.
Adım 2. Popülasyondaki her bir çözüm için uygunluk değerini hesapla.
Adım 3. Durdurma kriterini kontrol et, sağlanıyorsa Adım 5'e git. Sağlanmıyorsa aşağıdaki adımlara devam et.
Adım 3.1. Doğal seleksiyon işlemi uygula.
Adım 3.2. Çaprazlama işlemi uygula.
Adım 3.3. Mutasyon işlemi uygula.
Adım 4. Adım 2'ye git.
Adım 5. Araştırmayı sonlandır.

Durdurma kriteri genellikle maksimum jenerasyon sayısıdır. İşlemler verilen maksimum jenerasyon sayısı kadar tekrarlanır. Bu sayı tamamlanınca algoritmanın çalışması sona erer. GA ile geliştirilen RR yaklaşımı, devam eden bölümlerde [16-19] aralığında verilen çalışmalarla karşılaştırılmıştır.

2.3. Kullanılan Yazılım ve Veriler (Software and Data)

Bu algoritmaya ait temel giriş verileri olarak proses işlem bilgileri kullanılmıştır. RR algoritmasına ait çıkış parametreleri ise bekleme zamanı (WT), ortalama bekleme zamanı (AWT), gerçekleştirilen işlem zamanı (TT), ortalama gerçekleştirilen işlem zamanı (ATT), yanıt zamanı (RT) ve ortalama yanıt zamanı (ART) parametreleridir. RT değeri ilgili prosesin aldığı ilk WT değeri olarak kabul edilmiştir.
RR ve GA'nın birlikte çalışabilmesi için bu çalışma kapsamında bir yazılım geliştirilmiştir. C# ve Octave dilinde geliştirilen bu yazılımda dört adet metot kullanılmıştır. Bunlar RR, Rulet Teker Seleksiyon, Çaprazlama ve Mutasyon için yazılmış metotlardır. Bu metotların haricinde yazılımın içerisinde bulunan dahili "main" metodu bulunmaktadır.

Program ilk çalıştırıldığında ekranda sorulan proses sayısı, proses süreleri ve geliş zamanları girildikten sonra algoritma çalışmaya başlamaktadır. Proses bilgileri kullanıcı tarafından girilmektedir ancak istenildiği durumlarda program içerisinde de tanımlanabilmektedir.

"RR" metodu, giriş parametreleri tutan *prs*, geliş zamanlarını tutan *arrivalTime* ve rasgele belirlenmiş kuantum sürelerini tutan *quant* dizileridir. Çıkış parametreleri ise *AWT* ve *ATT* sürelerini tutan dizilerdir. Bu metot ile prosesler RR algoritmasının benzetimini yaparak ortalama süreleri hesaplar. Buradan çıkan sonuçlar, dışarıdaki parametrelere aktarılarak değerlendirme fonksiyonunda kullanılır.

"RuletTekerSeleksiyon" metodu ile hesaplanan uyum değerlerine rulet teker yöntemi ile seçme işlemi uygulanır. Aynı zamanda elitizm işlemini gerçekleştiren kodlar da rulet teker metodunun içinde yer almaktadır. Bu kodlar en iyi sonuçların saklamasını sağlar. Bunu sağlamak için yaptığı işlem ise en iyi değerleri sürekli seçerek bir sonraki jenerasyona aktarmasıdır. Elitizm operatörü için 0,8 değeri belirlenmiştir.

"Caprazlama" metodu, **ç**aprazlama katsayısına ve seçilen rasgele değere göre giriş parametresi olarak aldığı kuantum

değerleri üzerinde çaprazlama işlemini gerçekleştirir. Yeni kuantum değerlerini döndürür. "Mutasyon" metodu, kuantumlar üzerinde mutasyon işlemini uygular ve yeni kuantumları dışarı aktarır.

Önerilen yaklaşımda GA RR yönteminde kullanılan kuantum süresinin en iyi değerini bulmaya uyarlanır. Kuantum süresi her bir proses için bir turda CPU üzerinde işletileceği süreyi verir. Test işlemlerinde genellikle bu süre [1ms 100ms] aralığında alınmaktadır [20]. Kuantum süresi, bu değer aralığında GA ile bulunmaya çalışılır. Amaç fonksiyonu bulunan bu değerin kullanılması durumunda hesaplanan AWT değeridir. Amaç fonksiyonu değeri Tablo 1'de verilen ATT/AWT değerinin hesabıyla bulunur. Amaç fonksiyonunda üretilen kuantum zamanı iyileştirilmeye çalışılan çözümdür. Bu çözüm [1ms 100ms] aralığında tek parametre değerinden oluşmaktadır.

GA parametre değerleri Srivinas vd. [21] çalışmasında önerilen değer aralığında olacak şekilde seçilmiştir. Popülasyon büyüklüğü 30 ve jenerasyon sayısı 100 olarak alınmıştır (Tablo 2).

Tablo 2. Genetik algoritmada kullanılan parametre değerleri (Parameter values used for genetic algorithm)

Çaprazlama	Mutasyon	Popülasyon	Jenerasyon
0,50	0,033	30	100

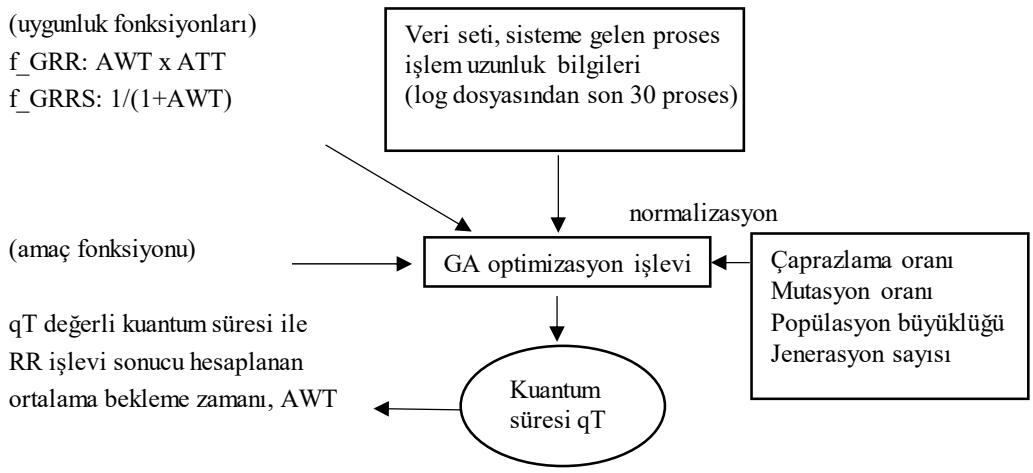
Genetik algoritmayı kullanan GRR ve iyileştirilmiş hali GRRS Şekil 2'de verilen modeli kullanır. GRRS yaklaşımında GRR uygunluk fonksiyonu olan f_GRR yerine f_GRRS kullanılır. Şekil 2'de verilen model Tablo 3'de yer alan algoritmayı kullanır. Genetik algoritma kullanarak en uygun kuantum süresini bulmaya çalışan GRR algoritması

Tablo 1. Amaç fonksiyonunda hesaplanan ATT/AWT değerleri (Values of ATT/AWT calculation of objective function)

1.	turn=1
	//turn: proses sırası, bT: proses işlem zamanı, qT: kuantum süresi,
	//r_bt: prosesin kalan işlem zamanı, pT: proses işletilme süreleri
	//wT: proses bekleme zamanı, n:proses sayısı

2.	//Döngü gerçekleştir (tüm prosesler tamamlana kadar) Eğer($r\_bt_{turn} > 0$)
	a) Eğer($r\_bt_{turn} > qT$)
	$r\_bt_{turn} = r\_bt_{turn} - qT$
	pT=[pT qT]
	//pT dizisine qT eklenir
	Değilse
	pT=[pT r_bt_{turn}]
	//pT dizisine son kalan işlem zamanı eklenir
	$r\_bt_{turn} = 0$
	$wT_{turn} = \sum(pT) - bT_{turn}$
	$tT_{turn} = \sum(pT)$
	b) turn=turn+1
	Eğer(sıra sonuna gelindiyse)
	turn=1

3.	$AWT = \sum(wT)/n$ $ATT = \sum(tT)/n$



Şekil 2. Genetik algoritma tabanlı optimum kuantum süresini hesaplayan model (Genetic algorithm based model calculating optimum quantum time)

önerimizin temel yapısını oluşturur. Performans karşılaştırmalarında test amaçlı olarak oluşturulan proses verilerinde bu değer aralığında rastgele üretilen süreler kullanılmıştır. Kuyrukta proses sıralamasının işlem zamanlarına göre yapılmasıyla ön işlem gerçekleştirilebilir. Her yeni proses anahtarlaması (process switching) sonrası kuantum süresinin hesabı ile son işlem gerçekleştirilebilir. Bahsedilen bu ön ve son işlemler GRR algoritmasına GRRS özelliği kazandırır.

Tablo 3. Genetik algoritmada tabanlı GRR/GRRS algoritması (Genetic algorithm based GRR/GRRS algorithm)

1.	Başlangıç
2.	Rastgele işlem zamanlı prosesler üret
3.	Prosesleri hazır kuyruğuna(ready queue) ata
4.	Eğer GRRS etkin ise hazır kuyruğundaki prosesleri sırala, değilse proses geliş sırasını kullan
5.	GA'dan (Şekil 2 verilen model) kuantum süresini sorgula ve ata
6.	Aşağıdaki adımları tekrarla
7.	- Hazır kuyruğunda yer alan sıradaki prosesi - CPU'ya ata, kuantum süresince işlet - Eğer GRRS etkin ise GA'dan
8.	(Şekil 2 verilen model) kuantum süresini sorgula ve ata, değilse sonraki adıma geç
9.	- Hazır kuyruğu dolu ise Adım 6'ya git, değilse bir sonraki adıma geç
10.	Ortalama performans değerlerini hesapla.

Program sonuçlarını ve parametrelerin etkilerini test edebilmek amacıyla prosesler kısa (0-10 ms), orta(10-50 ms) ve uzun (50-100 ms) olmak üzere üç kategoriye ayrılmıştır. Bu gruplar oluşturulurken, linux işletim sisteminin proses çalıştırmalarında gözlemlenen değerler dikkate alınarak genel bir gruplama yoluna gidilmiştir. Prosesler dizisi, belirlenen aralıklardan düzgün (uniform) dağılımlı rasgele olarak seçilmiştir. Bu şekilde 30 adet veri üretilmiştir. Bölüm 3.6.'da karşılaştırma yapılan çalışmalardaki yayınlarda verilen verilerde veri setimize eklenmiştir. Bu sayede toplamda 34 proses verisi oluşmuştur. Tablo 4'de bahsedilen bölümdeki test işlemlerinde kullandığımız veriler yer almaktadır. Kuyruk uzunluğu 5 olan toplam 33 proses verisi, kuyruk uzunluğu 4 olan 1 adet proses verisi verilmiştir. Tabloda son satırdaki kuyruk uzunluğunun diğerlerinden farklı olarak 4 olmasının nedeni karşılaştırma yapılan ilgili yayınlarda kullanılan veri olmasından dolayıdır.

3. SONUÇLAR VE TARTIŞMALAR
(RESULTS AND DISCUSSIONS)

Proses verileri standart RR, RR tabanlı bazı güncel algoritmalar ve önerilen genetik tabanlı yaklaşımı içeren algoritma ile işleme alınarak istenen süreler hesaplanmıştır. Yapılan bu çalışmalar sonucunda elde edilen veriler ve kıyaslamalar grafikler ve tablolar üzerinde açıklanmıştır.

3.1. Kısa Proseslerde RR ve Genetik Algoritmalı RR Kıyaslaması
(Comparison of RR and RR with Genetic Algorithms in Short Processes)

Kısa prosesler üzerinde RR ile beraber bu makalede önerilen GRR algoritması da uygulanarak deneysel çalışmalar yapılmıştır. Veriler, kuyruk uzunluğu 5 olan, 30 adet düzgün dağılımlı rasgele üretilen proses verisi üzerinde uygulanan yöntem sonunda sonuçların ortalaması alınarak elde edilmiştir. Yapılan çalışmalar sonucunda elde edilen veriler grafikler ile gösterilmiştir.

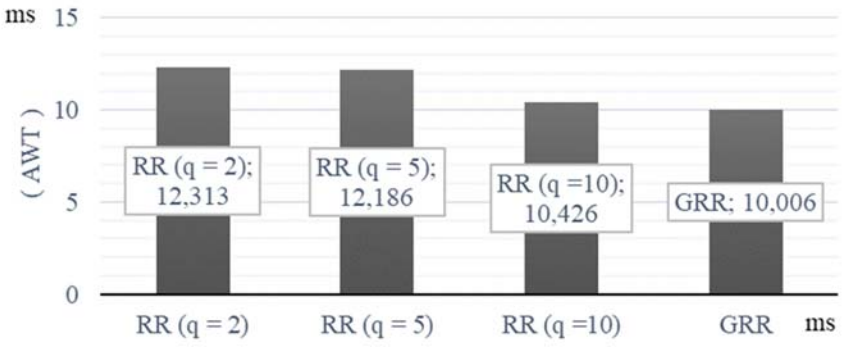
Şekil 3'de RR için üç farklı kuantum süresi seçildiğinde ve GRR uygulandığında ortaya çıkan AWT değerlerinin ortalaması gösterilmiştir. Kuantum süreleri milisaniye (ms) biriminde ele alınmıştır. Bu sonuçlara göre GRR ile elde edilen AWT değerinin diğer RR sonuçlarına göre daha kısa olduğu görülmektedir.

Şekil 4'de RR için üç farklı kuantum süresi seçildiğinde ve GRR uygulandığı zaman ortaya çıkan ATT değerlerinin ortalamaları gösterilmiştir. Bu sonuçlara göre GRR ile elde edilen ATT'nin diğer RR sonuçlarından daha kısa olduğu görülmektedir.

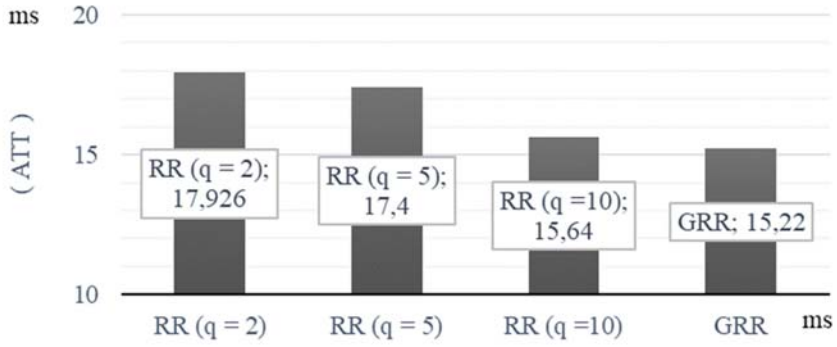
Şekil 5'de RR için üç farklı kuantum süresi seçildiğinde ve GRR uygulandığı zaman ortaya çıkan ART değerlerinin ortalamaları gösterilmiştir. Bu sonuçlara göre GRR ile elde edilen ART değerinin, RR sonuçlarına göre ortalama olarak bir değer aldığı görülmektedir. Kuantum süresi 5 ms olarak seçildiğinde ortalama kuantuma göre hesaplanan ART

Tablo 4. Karşılaştırmalarda kullanılan 34 adet proses verisi (34 process data to be used for comparisons)

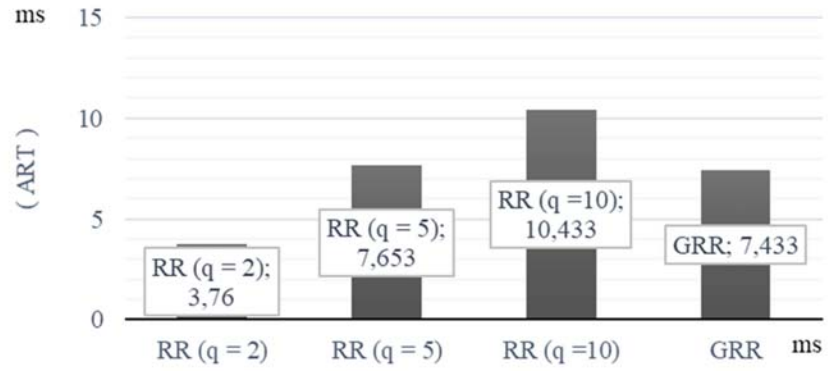
Proses işlem zamanları(ms)																	
1.	53	49	30	23	4	**13.**	38	63	93	58	1	**25.**	16	9	39	17	95
2.	19	94	60	31	12	**14.**	73	7	50	74	39	**26.**	64	60	13	29	73
3.	20	7	97	91	88	**15.**	4	58	70	75	11	**27.**	22	71	77	39	46
4.	69	88	76	53	23	**16.**	91	16	23	27	73	**28.**	53	48	12	74	54
5.	32	55	70	81	89	**17.**	93	71	5	92	87	**29.**	35	91	62	58	72
6.	86	18	24	58	78	**18.**	56	38	24	79	96	**30.**	51	9	60	4	3
7.	57	87	43	34	74	**19.**	43	22	39	95	30	**31.**	22	18	9	10	5
8.	20	68	59	6	8	**20.**	92	94	27	81	2	**32.**	11	24	37	52	71
9.	64	81	45	14	41	**21.**	98	9	12	51	2	**33.**	21	34	47	62	81
10.	40	77	35	72	79	**22.**	11	11	49	72	12	**34.**	10	14	70	120	-
11.	1	43	84	75	54	**23.**	70	87	80	75	24						
12.	36	4	83	67	90	**24.**	16	76	97	63	4						



Şekil 3. Kısa proseslerde ortalama bekleme süresinin RR ve GRR yöntemlerine göre sonuçları
(Results of AWT in short processes according to RR and GRR)



Şekil 4. Kısa proseslerde ortalama dönüş süresinin RR ve GRR yöntemlerine göre sonuçları
(Results of ATT in short processes according to RR and GRR)



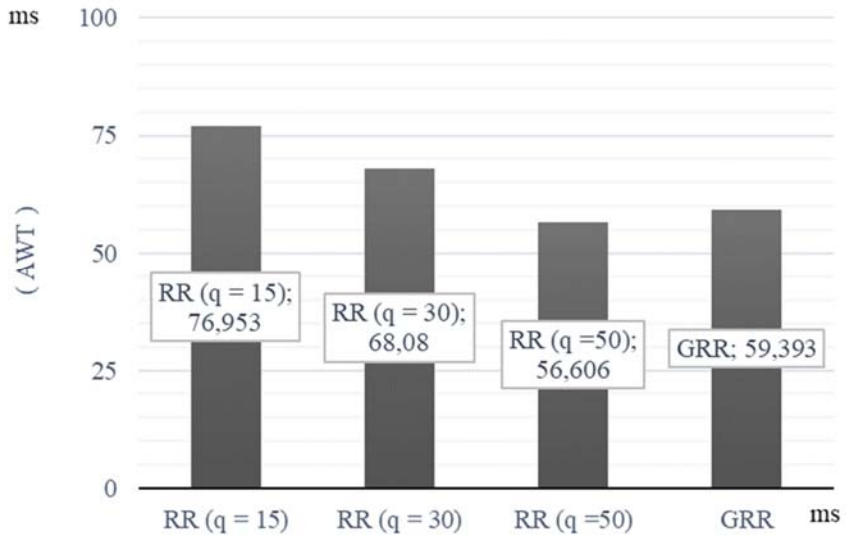
Şekil 5. Kısa proseslerde ortalama yanıt süresinin RR ve GRR yöntemlerine göre sonuçları
(Results of ART in short processes according to RR and GRR)

değeri 7,653 ms olmuştur. GRR'de ise bulunan sonuç 7,433 ms'dir. Bu durumda GRR'nin tüm parametreler bakımından en iyi sonuca ulaşmaya çalıştığı düşünülmektedir. En iyi sonuç ile en yüksek sonuç arasında ortalama değerler vermektedir.

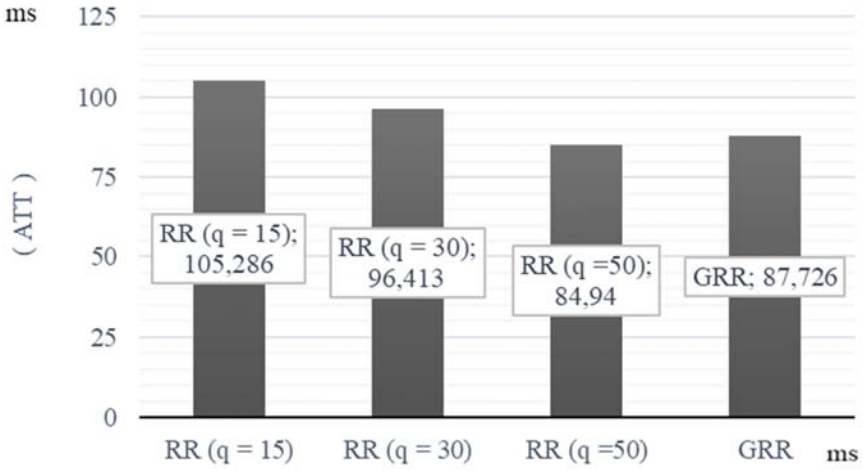
3.2. Orta Proseslerde RR ve Genetik Algoritmalı RR Kıyaslaması (Comparison of RR and RR with Genetic Algorithms in Medium Processes)

Orta prosesler üzerinde RR ile önerilen GRR algoritması da uygulanarak deneysel çalışmalar yapılmıştır. Yapılan çalışmalar sonucunda elde edilen veriler grafikler üzerinde gösterilmiştir. Şekil 6'da RR için üç farklı kuantum süresi seçildiğinde ve GRR uygulandığı zaman ortaya çıkan AWT değerleri gösterilmiştir. Bu sonuçlara göre GRR ile elde edilen AWT'nin RR ile elde edilen en küçük ve ortanca AWT değerleri arasında olduğu görülmektedir.

Şekil 7'de RR için üç farklı kuantum süresi kullanıldığında ve GRR'ye uygulandığında elde edilen ATT değerleri verilmiştir. Bu değerlere bakıldığında GRR ile elde edilen ATT'nin RR ile elde edilen değerlerin arasında değerler aldığı görülmektedir.



Şekil 6. Orta proseslerde ortalama bekleme süresinin RR ve GRR yöntemlerine göre sonuçları
(Results of AWT in medium processes according to RR and GRR)



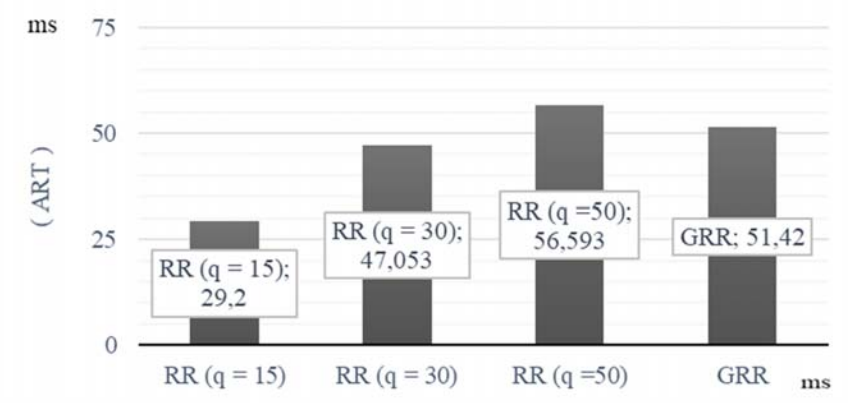
Şekil 7. Orta proseslerde ortalama dönüş süresinin RR ve GRR yöntemlerine göre sonuçları
(Results of ATT in medium processes according to RR and GRR)

Şekil 8'de RR için 3 farklı kuantum süresi seçildiğinde ve GRR işletildiğinde ortaya çıkan ART değerleri gösterilmiştir. Bu sonuçlara göre GRR ile elde edilen ART'nin RR ile elde edilen değerlerin arasında bir değer aldığı görülmektedir.

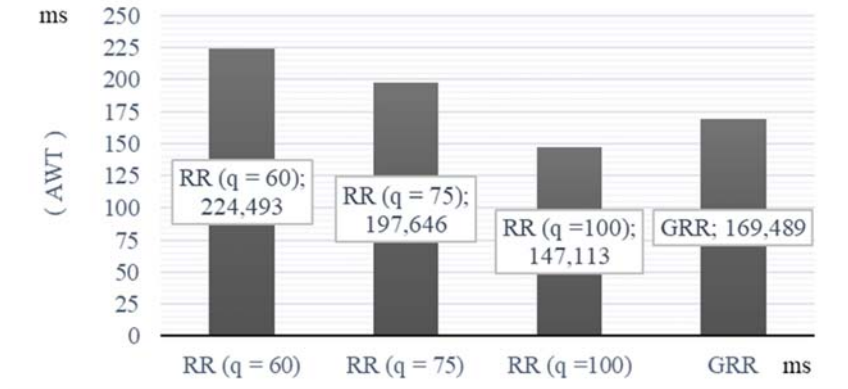
Bu durumda, ART ile verilen ortalama yanıt süresi anlamında GRR'nin tüm parametreler bakımından en iyi sonuca odaklandığı düşünülerek ortalama değeri elde ettiği görülmektedir. Yani elde edilen sonuçlar arasından en iyi sonuç ile en büyük sonuç arasında bir sonuç ortaya koymaktadır. Bu da aslında genetik algoritmanın çalışma mantığının sonucunu yansıtmaktadır. Burada dikkat edilmesi gereken bir diğer ayrıntı ise GRR algoritması dinamik yapısı sayesinde otomatik bir şekilde kuantum süresini belirlemektedir. Standart RR'de olduğu gibi tüm proses setleri için sabit bir değer zorunluluğu bulunmamaktadır.

3.3. Uzun Proseslerde RR ve Genetik Algoritmalı RR Kıyaslaması
(Comparison of RR and RR with Genetic Algorithms in Long Processes)

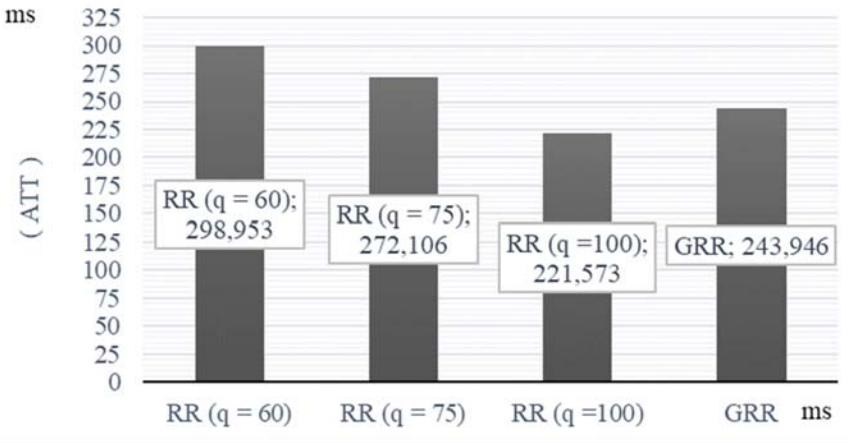
Uzun prosesler üzerinde RR ile önerilen Genetik RR algoritması uygulanarak deneysel çalışmalar yapılmıştır. Yapılan çalışmalar sonucunda elde edilen veriler grafikler üzerinde gösterilmiştir. Şekil 9'da RR için 3 farklı kuantum süresi seçildiğinde ve GRR işletildiğinde elde edilen AWT değerleri gösterilmiştir. Bu sonuçlara göre GRR ile elde edilen AWT değerlerinin RR ile elde edilen en küçük ve ortanca AWT değerleri arasında bir değer aldığı görülmektedir. Şekil 10'da RR için 3 farklı kuantum süresi kullanılmıştır. GRR işletildiğinde ortaya çıkan ATT değerleri verilmiştir. Bu sonuçlara göre GRR ile elde edilen ATT'nin RR ile elde edilen değerler arasında olduğu görülmektedir.



Şekil 8. Orta proseslerde ortalama yanıt süresinin RR ve GRR yöntemlerine göre sonuçları
(Results of ART in medium processes according to RR and GRR)



Şekil 9. Uzun proseslerde ortalama bekleme süresinin RR ve GRR yöntemlerine göre sonuçları
(Results of AWT in long processes according to RR and GRR)



Şekil 10. Uzun proseslerde ortalama dönüş süresinin RR ve GRR yöntemlerine göre sonuçları
(Results of ATT in long processes according to RR and GRR)

Şekil 11'de RR için 3 farklı kuantum süresi seçildiğinde ve GRR'ye uygulandığında ortaya çıkan ART değerleri verilmiştir. GRR'nin ürettiği sonuç değerler standart RR'nin ürettiği değerler arasında yer almaktadır. Bu durumda GRR'nin tüm parametreler bakımından en iyi sonuca odaklandığı düşünülerek ortalama değeri elde ettiği görülmektedir.

3.4. Ortalama Kuantum Sürelerinin Proses Uzunluklarına Göre Değişimi
(Variation of Average Quantum Times According to Process Lengths)

Standart RR ile veriler üzerinde sabit kuantum süreleri kullanılarak yapılan işlemlere ait kuantum sürelerinin ortalaması ve GRR ile optimum olarak hesaplanan kuantum

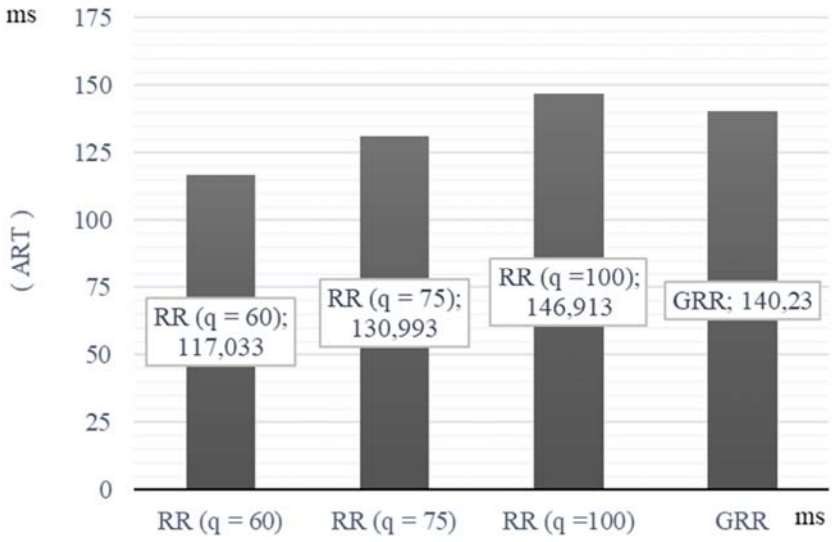
sürelerinin ortalaması Şekil 11'de karşılaştırılmalı olarak gösterilmiştir. Kullanılan sabit kuantum süreleri kısa prosesler için 2, 5 ve 10; orta prosesler için 15, 30 ve 50; uzun prosesler için ise 60, 75 ve 100 olarak belirlenmiştir.

Şekil 12'e göre kısa prosesler için GRR ile hesaplanan kuantum sayısı testlerde kullanılan sayı değerine çok yakındır. Orta ve uzun proseslerde ise GRR ile hesaplanan kuantum sayısı testlerde kullanılan sayı değerine yakın olmakla beraber bu değerlerden büyük olduğu görülmektedir. Elde edilen sonuçlara ve grafiklere bakıldığında GRR'nin Standart RR'ye yakın sonuçlar elde etmesiyle beraber, diğer parametreleri de iyileştirebilecek şekilde tüm sonuçlar açısından (AWT, ART, ATT) daha iyi ve ortak bir kuantum değeri hesapladığı görülmektedir.

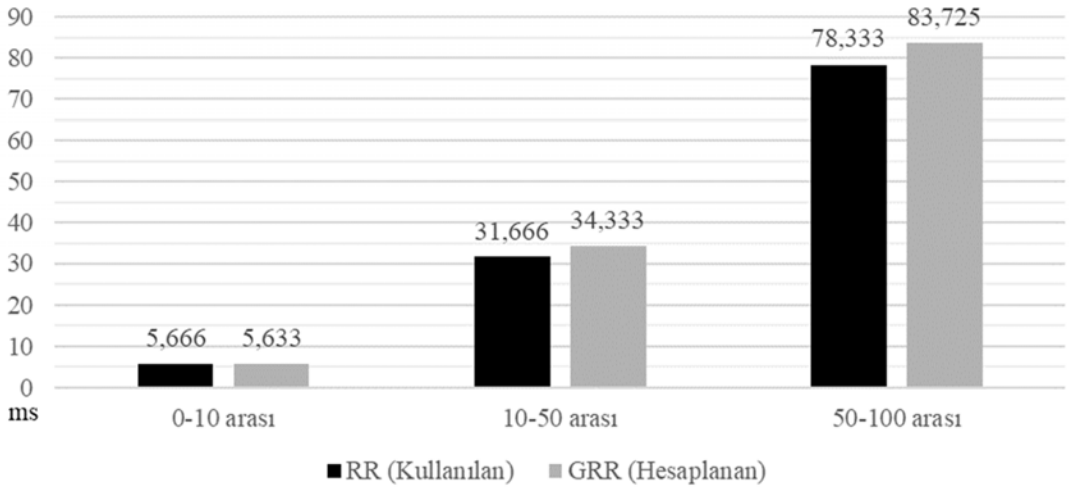
3.5. GRR ve Diğer Tarifeleme Algoritmaları
(GRR and Other Scheduling Algorithms)

Kısa, orta ve uzun proseslerde yapılan deneysel çalışmalar sonucunda elde edilen verilere göre AWT ve ATT grafikleri hazırlanmıştır. Deneysel çalışmalar SJF, FCFS, RR ve geliştirilen yöntemi içermektedir.

Kısa proseslerde elde edilen sonuçlara göre ortaya çıkan grafikte (Şekil 13) en kısa AWT ve ATT'yi SJF algoritması vermektedir. Ancak SJF ve FCFS kesikli (preemptive) algoritmalardan olmadığı için burada sadece bilgi amaçlı olarak verilmiştir. Bu değere en yakın değeri GRR üretmektedir. Önerilen yöntem kısa proseslerde standart RR ve FCFS'den daha iyi sonuç vermiştir.



Şekil 11. Uzun proseslerde ortalama yanıt süresinin RR ve GRR yöntemlerine göre sonuçları
(Results of ART in long processes according to RR and GRR)



Şekil 12. Ortalama kuantum sürelerinin proses uzunluklarına göre değişimi
(Variation of average quantum times according to process lengths)

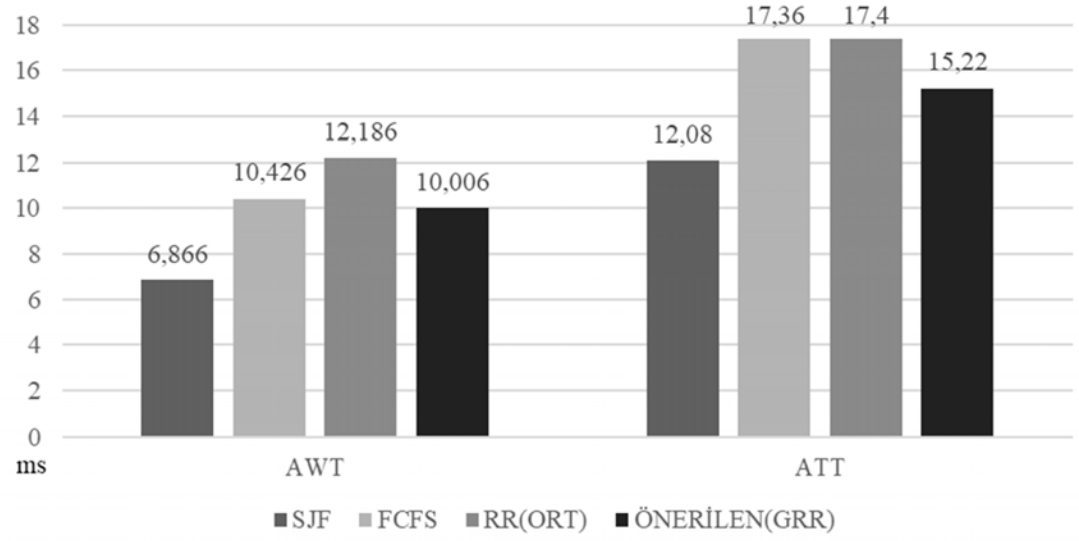
Orta uzunluktaki proseslerde elde edilen sonuçlardan ortaya çıkan grafiğe (Şekil 14) göre en kısa AWT ve ATT'yi SJF algoritması vermektedir. Daha sonra FCFS bu sırayı takip etmektedir. Önerilen yöntem ise orta proseslerde FCFS'ye yakın değerler verirken, RR'den daha iyi değerler vermiştir. SJF ve FCFS kesikli algoritma olmadığı için burada bilgi amaçlı olarak verilmiştir. Bu anlamda GRR'nin standart RR ile karşılaştırılması anlam kazanmaktadır.

Uzun proseslerde elde edilen sonuçlardan ortaya çıkan grafiğe (Şekil 15) göre en kısa AWT ve ATT'yi yine SJF algoritması vermektedir. FCFS bu sırayı takip etmektedir. Önerilen yöntem ise orta proseslerde FCFS'ye yakın değerler üretirken, RR'den daha iyi sonuçlar vermiştir. Elde edilen değerler incelendiğinde önerilen algoritmanın kısa, orta ve uzun prosesler için daha iyi sonuçlar verdiği görülmektedir. GRR için elde edilen değerler, standart RR'den daha performanslı olduğunu göstermektedir.

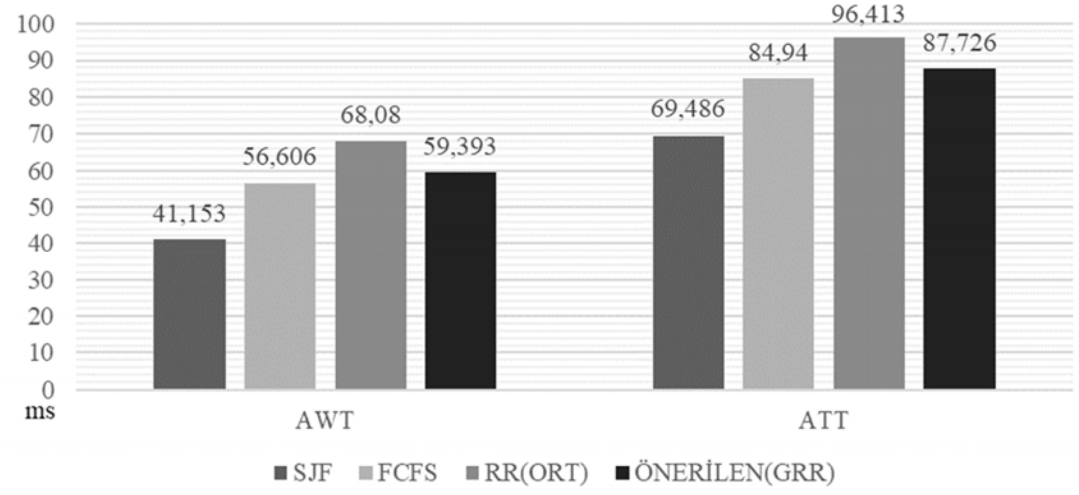
3.6. GRR, GRRS ve Diğer Geliştirilmiş Algoritmalar İçin Performans Sonuçları
(Performance Results for GRR, GRRS and Other Improved Algorithms)

Bu bölüm, standart RR ile güncel bazı RR tabanlı iyileştirilmiş CPU çizelgeleme performanslarının önerdiğimiz GRR/GRRS algoritmalarıyla karşılaştırılmasını içerir. Karşılaştırma verileri, Şekil 2'de verilen model ve bu model üzerinde Tablo 3'de verilen algoritma işletilerek elde edilmiştir.

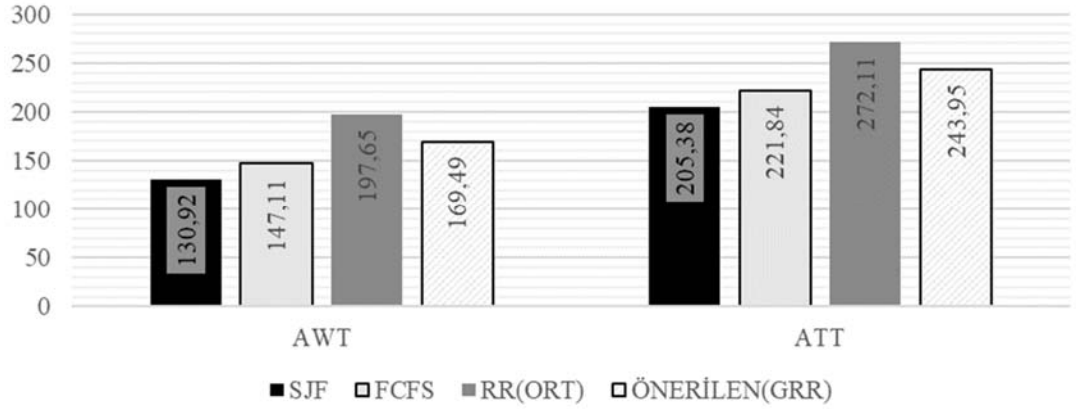
Dhumal vd. çalışmalarındaki [7] verilerle kıyaslandığında (Tablo 5) elde ettiğimiz sonuçların diğer algoritma çıktılarına göre daha performanslı olduğu görülmektedir. Gösterilen 10 adet sonucun ortalamasına bakıldığında, önerilen yöntemin(GRRS) Dhumal'ın yöntemine göre yaklaşık %7 oranında daha kısa süreli ATT değerini ürettiği görülmektedir. Tablo 5'deki sonuçlara ait ortalamalar Eş. 6'da verildiği şekildedir.



Şekil 13. Kısa proseslerde algoritma sürelerinin karşılaştırılması (Comparison of algorithm times in short processes)



Şekil 14. Orta proseslerde algoritma sürelerinin karşılaştırılması (Comparison of algorithm times in medium processes)



Şekil 15. Uzun proseslerde algoritma sürelerinin karşılaştırılması (Comparison of algorithm times in long processes)

GRRS < GA RR Dinamik < GRR < Standart RR (6)

Tablo 5. Dinamik kuantum tabanlı genetik döner çekirdek algoritması [7] ve genetik tabanlı round robin algoritmaları (Dynamic quantum based genetic round robin algorithm [7] and genetic based round robin algorithms)

Proses Bilgileri	Ortalama Bekleme Süresi (AWT)			
Proses Süreleri	Standart RR	GA RR Dinamik [7]	GRR	GRRS
15,12,5,18,16	36	20,6	24,8	20,4
12,14,10,12,11	39,2	22	24,4	21,8
80,70,20,15,75	123	71	117	67
40,45,50,30,35	85	72	85	70
19,40,7,25,18	54,4	35	47	29
47,37,17,15,8	62,4	37,6	59,2	29,6
120,118,55,84,77	292,2	155	273,8	147,4
70,72,46,67,58	194,6	119,2	164,6	112,4
32,26,17,45,38	82,8	53,2	63,4	49,6
78,57,18,28,33	111	62,6	109	55,8
Ortalama	108,06	64,82	96,82	60,3

anarjee vd. ait [6] çalışmada elde edilen sonuçlar ile önerilen yöntem(GRR ve GRRS) kıyaslandığında Tablo 6'de verilen sonuçlar elde edilmiştir. Sonuçlara bakıldığında en iyi performansı sağlayan GRRS algoritmasının ORR algoritmasına göre yaklaşık %47 oranında daha kısa ATT değeri ürettiği görülmektedir. Singh vd. [16] çalışmasından elde edilen sonuçlar ile önerilen yöntemden elde edilen sonuçlar kıyaslandığında Tablo 7 elde edilmiştir. Elde edilen ortalamalar, Eş. 7'de verildiği şekildedir. Buradan GRRS'nin devamında GRR'nin ATT değerinde daha performanslı olduğu görülmektedir.

GRRS < GRR < Singh [16] (7)

Punhani vd. çalışmalarında [17] kullandığı veriler ile GRR/GRRS kıyaslanarak üretilen ATT değerleri Tablo 8'de verilmiştir. Bu tabloya göre GRRS'nin en yakın performanslı algoritmaya göre %18 daha iyi sonuç verdiği görülmektedir. Elde edilen ortalamalar, Eş. 8'de verildiği şekildedir.

Tablo 6. Optimize edilmiş döner çekirdek çizelgeleme (ORR) [6] ile genetik tabanlı round robin algoritmaları (Optimized round robin scheduling(ORR) [6] and genetic based round robin algorithms)

Proses Bilgileri	Ortalama Bekleme Süresi (AWT)		
Proses Süreleri	ORR[6]	GRR	GRRS
11,24,37,52,71	48,4	48,4	48,4
21,34,47,62,81	68,4	68,4	68,4
53,49,30,23,4	104,7	77,6	38,8
19,94,60,31,12	124	86,8	45,4
20,7,97,91,88	198,8	77,2	71
69,88,76,53,23	228,9	149	93
32,55,70,81,89	102,8	102,8	102,8
86,18,24,58,78	84,4	100,8	67,6
57,87,43,34,74	182,9	121,8	90,6
20,68,59,6,8	87	57,4	29,4
Ortalama	123,0	89,0	65,5

Tablo 7. CPU çizelgeleme için optimimum döner çekirdek çizelgeleme [16] ile genetik tabanlı round robin algoritmaları (An optimized round robin scheduling algorithm for cpu scheduling [16] and genetic based round robin algorithms)

Proses Bilgileri	Ortalama Bekleme Süresi (AWT)		
Proses Süreleri	Singh[16]	GRR	GRRS
22,18,9,10,5	29,8	33,8	29,8
21,34,47,62,81	80,4	68,4	68,4
53,49,30,23,4	55,6	77,6	38,8
19,94,60,31,12	56,4	86,8	45,4
20,7,97,91,88	98	77,2	71
69,88,76,53,23	109	149	93
32,55,70,81,89	122,4	102,8	102,8
86,18,24,58,78	84,8	100,8	67,6
57,87,43,34,74	106,6	121,8	90,6
20,68,59,6,8	41,4	57,4	29,4
Ortalama	78,4	87,6	63,7

GRRS < Punhani [17]< Priority< GRR (8)

Alsulami vd. yaptığı çalışmayla [18] kıyaslandığında önerilen GRR ve GRRS algoritmalarının ATT değerlerinde daha iyi düşüşler sağladığı görülmektedir (Tablo 9).

Tablo 8. CPU Evrimsel algoritma yardımıyla çoklu kriterlere dayalı bir CPU çizelgeleme [17] ile genetik tabanlı round robin algoritmaları
(A CPU scheduling based on multi criteria with the help of evolutionary algorithm [17] and genetic based round robin algorithms)

Proses Bilgileri	Ortalama Bekleme Süresi (AWT)			
Proses Süreleri	Priority	Punhani[17]	GRR	GRRS
63,37,87,35,5	114,4	63,2	104,4	52,4
93,5,88,50	114	72,6	91	50,8
12,81,65,11	78,4	6l,6	86,4	30,5
82,39,35,27	104,2	75,54	113	47,5
85,80,35,83,52	120,4	108,2	192,4	107,8
79,50,14,58,38	104,6	68	110,4	65,6
90,6,89,65,59	124,2	88,8	164,6	84
59,38,43,41,71	112,6	84,4	95,4	84
89,63,57,51,66	142,6	132	173,4	113,4
15,100,46,50,75	110,6	89,6	100,4	74,6
Ortalama	112,6	86,9	123,1	71,1

Tablo 9. Manhattan uzaklığı algoritması kullanılarak optimum döner çekirdek [18] ile genetik tabanlı round robin algoritmaları
(Optimal round robin scheduling using manhattan distance algorithm [18] and genetic based round robin algorithms)

Proses Bilgileri	Ortalama Bekleme Süresi (AWT)		
Proses Süreleri	Alsulami[18]	GRR	GRRS
40,77,35,72,79	133,4	106,6	96,2
1,43,84,75,54	100,6	75,2	63,2
36,4,83,67,90	77,8	77,8	68,2
38,63,93,58,1	128,4	116,8	59,4
73,7,50,74,39	132,8	97,4	63,6
4,58,70,75,11	82,4	82,2	47
91,16,23,27,73	112	84,4	52
93,71,5,92,87	201,2	137,4	99,8
56,38,24,79,96	107,4	93	80,2
43,22,39,95,30	83,8	77,8	59,8
Ortalama	116,0	94,9	68,9

Noon vd. [19] çalışmasında kullanılan yaklaşımdan elde edilen sonuçlar, GRR/GRRS algoritmalarının sonuçlarıyla birlikte Tablo 10'da verilmiştir. Elde edilen ortalama değerlere bakıldığında GRR ve GRRS algoritmalarının sırasıyla ANN'den %25 ve %45 oranında daha düşük (daha iyi performans sağlayacak şekilde) AWT değerleri ürettiği görülmektedir. Elde edilen ortalamalar, Eş. 9'da verildiği şekildedir [20].

GRRS<GRR<AN Metodu (9)

Tablo 10. İşletim sistemleri için yeni bir döner çekirdek tabanlı çizelgeleme algoritması [19] ile genetik tabanlı round robin algoritmaları
(A new round robin based scheduling algorithm for operating systems [19] and genetic based round robin algorithms)

Proses Bilgileri	Ortalama Bekleme Süresi (AWT)		
Proses Süreleri	AN[19]	GRR	GRRS
10,14,70,120	32	32	32
98,9,12,51,2	68,3	43	22
11,11,49,72,12	50,7	42,5	30
70,87,80,75,24	200,3	155,2	107,2
16,76,97,63,4	120,3	103,6	53,2
16,9,39,17,95	37,2	37,2	31,4
64,60,13,29,73	122,3	98,2	64,6
22,71,77,39,46	137,4	98,8	73,6
53,48,12,74,54	134	90,8	70,4
35,91,62,58,72	163,4	97,6	102
Ortalama	106,6	79,9	58,6

4. SONUÇLAR (CONCLUSIONS)

Bu makalede genetik tabanlı olarak standart RR'de yer alan kuantum süresini hesaplayan GRR ve GRRS algoritmaları önerilmiştir. Diğer standart CPU çizelgeleme algoritmalarıyla yapılan kıyaslamalar sonucunda geliştirilen yöntem, ATT değeri için kısa proseslerde en performanslı algoritma olarak bilinen SJF'den hemen sonra yer almıştır. Dolayısıyla FCFS ve RR'den daha iyi sonuçlar üretmiştir. SJF ve FCFS'in prosesleri kesikli işletme özelliği olmamasından dolayı RR ve diğer RR üzerinde iyileştirmelerle geliştirilen algoritmalar sıklıkla tercih edilmektedir. Önerilen algoritmaların ATT ve AWT değerlerinde RR'den daha iyi bir sonuç elde ettiği gözlenmiştir. GRR üzerinde yapılan iyileştirme ile etkinliğini arttırdığımız GRRS algoritmasının diğer kesikli algoritmalar içerisinde en iyi sonucu ürettiği yaptığımız testlerde gözlenmiştir.

TEŞEKKÜR (ACKNOWLEDGEMENT)

Bu çalışmanın bir kısmı makale yazarlarından Betül Koşmaz'ın "Döner Çekirdek CPU Tarifeleme Algoritmasının Optimum Parametre Değerlerinin Genetik Algoritma ile Bulunması" isimli yüksek lisans tezinden türetilmiştir.

KAYNAKLAR (REFERENCES)

1. Alam, B., Fuzzy round robin CPU scheduling algorithm. Journal of Computer Science, 9 (8), 1079-1085, 2013.
2. Kumarsaroj, S., Sharma, A. K., Chauhan, S. K., A novel CPU scheduling with variable time quantum based on mean difference of burst time, Proceeding-IEEE International Conference on Computing, Communication and Automation (ICCCA), 1342-1347, 2016.

3. Parekh, H. B., Chaudhari, S., Improved round robin CPU scheduling algorithm: round robin, shortest job first and priority algorithm coupled to increase throughput and decrease waiting time and turnaround time, International conference on global trends in signal processing, ınformation computing and communication (ICGTSPICC), 184–187, 2016.

4. Neshat, M., Sargolzaei, M., Najaran, A., Adeli, A., The new method of adaptive CPU scheduling using fonseca and fleming's genetic algorithm, Journal of Theoretical and Applied Information Technology, 37 (1), 1-16, 2012.

5. Fonseca, C. M., Fleming, P. J., Genetic algorithms for multiobjective optimization: Formulation, discussion, and generalization,. 5th International Conference on Genetic Algorithms, California, 416–423, 1993.

6. Banerjee, P., Singh, A., Kumar, R. Comparative performance analysis of optimized round robin scheduling (ORR) using dynamic time quantum with round robin scheduling using static time quantum in Real Time System. International Journal of Engineering and Computer Science, 8 (12), 24890-24893, 2019.

7. Dhumal, R. A., Maktum, T. A., Ragha, L., Dynamic quantum based genetic round robin algorithm, International Journal of Advanced Research in Computer and Communication Engineering, 3 (3), 5905-5908, 2014.

8. Hussein, Q. M., Hasoon, A. N., Dynamic process scheduling using genetic algorithm, Annual Conference on New Trends in Information and Communications Technology Applications (NTICT), 111–115, 2017.

9. Siregar, M. U., A new approach to CPU scheduling algorithm: genetic round robin. Int. J. Comput. Appl. Technol., 47 (19), 18-25, 2012.

10. Karaboğa, D., Yapay Zeka ve Optimizasyon Algoritmaları, Nobel Yayın Dağıtım, Ankara, 2011.

11. Tanenbaum, A. S., Modern Operating Systems 2nd Edition, Pearson Education, USA, 2002.

12. Silberschatz, A., Galvin, P. B., Gagne, G., Operating System Concepts Essentials, John Wiley & Sons, USA, 2014.

13. Kiraz A., Canpolat O., Ozkurt C., Taşkın H., Sarp, E., Examination of the criteria affecting Industry 4.0 with structural equation model and a pilot study, Journal of the Faculty of Engineering and Architecture of Gazi University, 35 (4), 2183-2196, 2020.

14. Holland, J. H., Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT press, 1992.

15. Satman, M. H., Genetik Algoritmalar, Türkmen Kitabevi, İstanbul, 2016.

16. Singh, A., Goyal, P., Batra, S., An optimized round robin scheduling algorithm for CPU scheduling, International Journal on Computer Science and Engineering(IJCSE), 02 (7), 2383–2385, 2010.

17. Punhani, A., Kumar, S., Chaudhary, R., Sharma, A. K., A CPU scheduling based on multi criteria with the help of evolutionary algorithm, 2nd IEEE International Conference on Parallel, Distributed and Grid Computing, 730-734, 2012.

18. Alsulami, A.A., et al. Performance Evaluation of Dynamic Round Robin Algorithms for CPU Scheduling. In Proceeding of IEEE SoutheastCon, 1-5, 2019.

19. Noon, A., Kalakech, A., Kadry S., A new round robin based scheduling algorithm for operating systems: dynamic quantum using the mean average, International Journal of Computer Science Issues, 8 (3), 224-229, 2011.

20. Kosmaz, B., Döner çekirdek cpu tarifeleme algoritmasının optimum parametre değerlerinin genetik algoritma ile bulunması, Yüksek Lisans Tezi, Fen Bilimleri Ens., Erciyes Üniv., 2019.

21. Srinivas, M., Patnaik, L. M. Genetic algorithms: A survey. Computer, 27 (6), 17-26, 1994.