TABU ARAMA ALGORİTMASININ KUYRUK PROBLEMİNE UYGULANMASI

APPLICATION OF TABU SEARCH ALGORITHM TO QUEUE PROBLEM

ÖMÜR GÜRBÜZ

PROF. DR. HÜLYA ÇINGI

Tez Danışmanı

Hacettepe Üniversitesi

Lisansüstü Eğitim – Öğretim ve Sınav Yönetmeliğinin

İstatistik Anabilim Dalı için Öngördüğü

YÜKSEK LİSANS TEZİ olarak hazırlanmıştır.

2015

Ömür GÜRBÜZ'ün hazırladığı "**Tabu Arama Algoritmasının Kuyruk Problemine Uygulanması**" adlı bu çalışma aşağıdaki jüri tarafından **İSTATİSTİK ANABİLİM DALI**'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Prof.Dr. M.Özgür YENİAY
Başkan

Prof.Dr. Hülya ÇINGI
Danışman Üye

Doç.Dr. Ufuk YOLCU
Üye

Bu tez Hacettepe Üniversitesi Fen Bilimleri Enstitüsü tarafından **YÜKSEK LİSANS TEZİ** olarak onaylanmıştır.

Prof. Dr. Fatma SEVİN DÜZ
Fen Bilimleri Enstitü Müdürü

TEŞEKKÜR

Tez çalışmamın her aşamasında değerli katkı ve eleştirileriyle yol gösteren, büyük bir anlayış ve sabırla çalışmamı bitirmem için beni teşvik eden danışmanım Sayın Prof. Dr. Hülya ÇINGI'ya; tüm çalışmalarımda beni destekleyip, moral ve motivasyon veren eşim Begüm GÜRBÜZ'e; her çıkmaza girdiğimde ve teorik anlamda desteğe ihtiyaç duyduğumda tüm sorularıma, zamanını esirgemeden cevaplayan Sayın Prof. Dr. Cem KADILAR, Sayın Doç. Dr. Haydar DEMİRHAN ve Sayın Doç. Dr. Çağdaş Hakan ALADAĞ'a içtenlikle teşekkür ederim.

ETİK

Hacettepe Üniversitesi Fen Bilimleri Enstitüsü, tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında,

- tez içindeki bütün bilgi ve belgeleri akademik kurallar çerçevesinde elde ettiğimi,
- görsel, işitsel ve yazılı tüm bilgi ve sonuçları bilimsel ahlak kurallarına uygun olarak sunduğumu,
- başkalarının eserlerinden yararlanılması durumunda ilgili esere bilimsel normlara uygun olarak atıfta bulunduğumu,
- atıfta bulunduğum eserlerin tümünü kaynak olarak gösterdiğimi,
- kullanılan verilerde herhangi bir tahrifat yapmadığımı,
- ve bu tezin bir bölümünü bu üniversitede veya başka bir üniversitede başka bir tez çalışması olarak sunmadığımı

beyan ederim.

…/…/2015

ÖMÜR GÜRBÜZ

ÖZET

Kuyruk problemi bilgisayar ağları ve telekomünikasyon gibi teknik konularda karşımıza çıktığı gibi, gündelik yaşamımızda da market kasaları, toplu taşıma, banka gişeleri, bilet gişeleri ya da trafik sıkışıklığı vb şeklinde karşımıza çıkmaktadır [1]. Ancak bu kuyruklar arasında en sık karşılaşılanı ve en çok zaman harcananlarından biri de market kasa kuyruklarıdır. Birçok insan gündelik yaşamını idame ettirebilmek için zorunlu ihtiyaçlarını karşılamak maksadı ile marketlere gitmekte ve market kasalarında kuyrukta beklemek zorunda kalmaktadır.

Perakende sektöründe kasa kuyruklarını yönetmek, hem müşteri memnuniyeti sağlamak hem de bordro maliyetlerini yönetebilmek adına çok büyük önem arz etmektedir. Ancak günümüz teknolojisinde bile birçok işletmede kuyruk problemi ya yönetici tecrübeleriyle ya da personel bazında detaylandırılmamış programlar aracılığıyla yapılmaktadır. Kullanılan programlar ise personellerin bireysel performanslarına göre çalışmayıp, çıktı olarak sadece çalışacak personel sayılarını belirttikleri ve sonucunda da kuyruk uzunluğuna ilişkin detaylı bir çıktı verememeleri sebebi ile kriz anları ya da boş zamanları bu çıktıya bakarak tahmin etmek mümkün olmamaktadır. Bunun sonucunda da büyük ölçüde müşteri kaybı ya da atıl personel oluşmaktadır. Bu çalışmada personelin bireysel performansları göz önüne alınarak kasa hattının benzetimi oluşturulmuş ve bu benzetim sonuçlarına göre personel vardiyası hazırlanmıştır. Bu sayede personelin çalışma performansına göre hangi saatlerde daha verimli olabileceği ve buna bağlı olarak üretilen vardiya sonucunda ise hangi gün hangi saatte kuyruk uzunluğunun ne kadar olacağı tahmin edilebilmiştir.

ABSTRACT

Delays and queuing problems are most common features not only in our daily-life situations such as at a bank or postal office, at a ticketing office, in public transportation or in a traffic jam but also in more technical environments, such as in manufacturing, computer networking and telecommunications [1]. However, supermarket checkout queue system is the most frequently used and one of the most time elapsed places. Many people buy their vital need for living from a supermarket and they wait in checkout queues.

In retail market, management of checkout queue lenght is very important for minimize payroll expences and maximize customer satisfaction. However , even today's technology, in many retail company checkout queue lenght is still tried to manage according by managers experiences or some programs which are not detailed about staffs' own performance. These programs are not detailed about staffs' own performans so the output of these programs show only working staffs counts and could not show detailed checkout queue lenght forecasts. Because of insufficient detail of these programs' output, companies could not forecast neighter crisis moments of checkout queue nor idle time of staffs. As a result of this stuation, companies lost customer satisfaction and have got lots of idle staffs. In this study, simulated checkout queue line according to staffs' personal performance and according to simulation's result, the best shift could be find. Thus, companies could forecast, which working time is more efficient for staffs according to its personal performance and what will be the checkout queue lenght during day.

İÇİNDEKİLER

ÖZET i
ABSTRACT ii
İÇİNDEKİLER iii
ÇİZELGELER v
ŞEKİLLER v
SİMGELER VE KISALTMALAR vi
1. GİRİŞ 1
1.1. Bu Çalışmanın Sağlayacağı Faydalar 1
1.2. Çalışmanın Genel Mantığı 2
2. KUYRUK TEORİSİ 3
2.1. Kuyruğa Geliş Süreci 6
2.2. Servis Mekanizması 6
2.3. Kuyruk Disiplini 6
3. MONTE CARLO BENZETİMİ 7
3.1. Rastgele Sayı Üreteçleri (RSÜ) 7
3.2. Ters Dönüşüm Tekniği 8
3.3. Üstel Dağılım için Ters Dönüşüm Tekniği 8
3.4. Poisson Dağılımı için Knuth Algoritması 9
3.5. MC Benzetim Tekniğinde Uygulanan Adımlar 9
3.5.1. Statik Modelin Oluşturulması 9
3.5.2. Girdi Değişkenleri İçin Dağılımların Tanımlanması 9
3.5.3. Rastgele Sayı Üreteçlerinin Oluşturulması 9
3.5.4. Analiz ve Karar Verme 9
4. KUYRUK TEORİSİ VE MONTE CARLO BENZETİMİNİN BİR ARADA KULLANILMASI 10
4.1. Kuyruk Probleminde MC Benzetim Tekniğinin Aşamaları 11
5. TABU ARAMA ALGORİTMASI (TA) 12
5.1. TA Algoritmasının Adımları 14
6. TABU ARAMA ALGORİTMASI VE MC BENZETİM TEKNİĞİNİN BİRLİKTE KULLANILMASI 14
7. TA ALGORİTMASININ KUYRUK PROBLEMİNE UYGULANASI 17

7.1. Tabu Arama Algoritması 17
7.1.1. TA Algoritması İçin Girdi Verileri 17
7.1.2. TA Algoritmasında Uygulanacak Kısıtlar 17
7.1.3. TA Algoritmasının İlk Vardiyayı Üretmesi ve Çözüm Değiştirme Mantığı 18
7.1.4. Çözümün Değiştirilmesi 19
7.2. Kuyruk Yapısının İncelenmesi 20
7.3. Kuyruk Sisteminin MC Benzetim Tekniği İle Modellenmesi 20
8. PROGRAMIN YAZILMASI 23
9. UYGULAMA 24
9.1. Programda Kullanılan ve Rastgele Üretilmiş Veriler 24
10. SONUÇ 33
11. PROGRAM KODLARI 34
11.1. TA Algoritması için Değişkenlerin Tanımlanması ve Okutulması 34
11.2. İlk Vardiyanın Oluşturulması 35
11.3. Tabu Arama Algoritması 36
11.4. Üretilen Vardiyalar Kısıtlara Uygun mu Kontrolü 37
11.5. Kasiyer Vardiyasını Hafızaya Al 37
11.6. Vardiyayı Değiştirmek 37
11.7. Poisson Dağılımından Rastgele Sayı Üretmek 39
11.8. Üstel Dağılımından Rastgele Sayı Üretmek 39
11.9. MC Benzetim Tekniği Kodları 39
KAYNAK 43
ÖZGEÇMİŞ 44

ÇİZELGELER

Tablo 1: Kasiyerler Hangi Günler Çalışıyor? 24

Tablo 2: Kasiyerlerin Pazartesi Günü Gelebilecekleri En Erken Vardiya 25

Tablo 3: Kasiyerlerin Ürün Okutma Performansları 25

Tablo 4: Kasiyerin Ödeme Alma Performansları 26

Tablo 5: Pazartesi ve Salı Günü İçin Mağaza Performansı 27

Tablo 6: Çarşamba ve Perşembe Günü İçin Mağaza Performansı 27

Tablo 7: Cuma ve Cumartesi Günü İçin Mağaza Performansı 28

Tablo 8: Pazar Günü İçin Mağaza Performansı 28

Tablo 9: Programın Üretmiş Olduğu Vardiya Çıktısı 29

Tablo 10: Düzenlenmiş Vardiya Çıktısı 29

Tablo 11: Üretilen En İyi Vardiya ile Ulaşılacak Kuyruk Uzunluğu Tahmini 30

Tablo 12: TA algoritması ile Üretilen 15 Vardiyanın 10 Kere Tekrarlanması İle Elde Edilen Kuyruk Sonuçları 31

Tablo 13: Model Özet Tablosu 32

Tablo 14: Anova Tablosu 32

Tablo 15: Katsayılar Tablosu 32

ŞEKİLLER

Şekil 1: Tek Aşamalı Tek İstasyonlu Kuyruk Yapısı 4

Şekil 2: Tek Aşamalı Çok İstasyonlu Kuyruk Yapısı 4

Şekil 3: Çok Aşamalı Tek İstasyonlu Kuyruk Yapısı 5

Şekil 4: Çok Aşamalı Çok İstasyonlu Kuyruk Yapısı 5

Şekil 5: Deneme Sayısı ve Yapılan Tekrarlara Göre Kuyruk Uzunluklarının Dağılımı 31

SİMGELER VE KISALTMALAR

Simgeler

λ:	Üstel ve Poisson dağılımına ait parametre

Kısaltmalar

TA:	Tabu Arama
MC:	Monte Carlo
RSÜ:	Rastgele Sayı Üreteci
MAS:	Mağazanın Açılış Saati
MKS:	Mağazanın Kapanış Saati
KSÇ:	Mağazanın toplam Kaç Saat Çalıştığı
KKÇ:	Mağazada Kaç Kasiyer Çalışıyor
MİS:	TA algoritması için Maksimum İterasyon Sayısı
HGÇ:	Kasiyerler Hangi Günler Çalışıyor
PEE:	Personelin En Erken vardiyaya başlanabilecek saat
KV:	Kasiyer Vardiyası
HSÇ:	Hangi Saatler Çalışıyor
MÇKS:	Mağazada Çalışan Kasiyer Sayısı
KU:	Kuyruk Uzunluğu
VÖKU:	Vardiya başlamadan 1 saat önceki kuyruk uzunluğu
VSKU :	Vardiya bittikten 1 saat sonraki kuyruk uzunluğu
KGS:	Müşterilerin kasa hattına geliş sıklıkları
AÜS:	Müşterilerin almış olduğu ürün sayıları
ÜOP:	Kasiyerlerin ürün okutma performansları
ÖAP:	Kasiyerlerin ödeme alma performansları
GM:	Mağazaya Gelen Müşteriler
OKU:	Ortalama Kuyruk Uzunluğu
SMS:	Saatlik Müşteri Sayıları
SMSV:	Saatlik Müşteri Sayıları Varyansı

1. GİRİŞ

Günümüzde birçok işletmede kuyruk problemleri hala yöneticinin tecrübeleri ile yönetilmektedir. Bunun sonucu olarak da önerilen çözüm sonrasında kuyruk için belirlenen kuyruk performans kriterleri önceden kestirilememektedir. Uygulanan çözüm sonrasında oluşabilecek kriz anlarını yönetmek işletmelere çok büyük ek maliyet ya da müşteri memnuniyeti kaybı oluşturmaktadır.

Perakende sektöründe kasa kuyruklarını yönetmek hem müşteri memnuniyetini yüksek tutmak hem de bordro maliyetlerini düşürebilmek adına çok önemlidir. Tüm bir alışveriş hiçbir yerde sıra beklemeden ve hiçbir personel ile iletişime geçmeden tamamlansa bile; kasa hattına gelindiğinde personel ile iletişime geçmemek ya da kuyrukta beklememek çok düşük bir ihtimaldir. Çoğu zaman müşteriler işletmeleri, işletme içerisinden iletişime geçtikleri personel ile hatırlar ve tanımlarlar. Bu yüzden işletmeden ayrılmadan önce görüşülen en son kişi olan kasa hattı personeli ile geçilen iletişimin olabildiğince iyi olmasını sağlamak, işletmelerdeki en önemli müşteri memnuniyeti çalışmasıdır. Bu iletişimi iyi olarak hatırlanmasını sağlamak ise çoğunlukla kuyrukta geçirilecek sürenin kısa olması ile alakalıdır. Bordro maliyetlerini minimize ederek, kuyrukta geçirilecek süreyi azaltmaya çalışmak tüm işletmeler için en büyük hedeflerden birisidir.

1.1. Bu Çalışmanın Sağlayacağı Faydalar

Bu çalışmada, perakende sektöründe faaliyet gösteren bir marketin kasa hattındaki kuyruk uzunluğunu en düşük seviyeye indirmek için kasiyer vardiyalarının nasıl olması gerektiği tespit etmeye çalışılmıştır. Ancak sistemin kullanım alanı sadece perakende sektörü ile sınırlı değil; çağrı merkezleri, vardiyalı çalışan banka şubeleri gibi personel performansları ölçülebilen ve personelleri vardiyalı çalışan tüm işletme ve kurumlarda da kullanılabilir. Bu çalışmada örnek olması için bir market kasa kuyruk sistemi ele alınacaktır. Bunun için marketteki kuyruk yapısının iyi öğrenilmesi, kuyrukta geçirilecek süreyi nelerin etkilediğinin tespit edilmesi ve bu etkiler arasındaki ilişkilerin doğru bir şekilde kurulması gerekmektedir. Kuyruk uzunluğunu etkileyen tüm faktörleri göz önüne alan bir model kullanılarak, hangi kasiyer vardiyası ile en kısa kuyruk uzunluğuna ulaşılabileceği tespit edilmeye çalışılacaktır.

Kullanılacak olan modelde, haftanın günleri ve günün saatlerine göre mağazaya giren müşteri sayıları ve müşterilerin aldığı ürün sayılarının stokastik olarak belirlenmesi ile kaynaktan kuyruğa gelişler tespit edilecektir. Kuyruk sisteminde her kasiyerin işlemi

tamamlaması için yapması gereken iki adım bulunmaktadır; ürünleri okutmak ve ödeme işlemini tamamlamak. Her bir kasiyerin bu adımlar için farklı performansı olacağından (bazı kasiyerlerin çok hızlı ürün okutabilmesi ya da çok hızlı ödeme alabilmesi gibi) kuyruk sisteminde bu performanslarında stokastik olarak belirlenmesi gerekmektedir.

Kuyruk sisteminin yapısı ve kuyruğu etkileyen faktörler tespit edilip modellendikten sonra, sıra en iyi kasiyer vardiyasının tespit edilmesindedir. Bu aşamada TA algoritması en iyi kasiyer vardiyasını bulmak için farklı kasiyer vardiyaları deneyip, bu vardiyalara göre kuyruk sonuçlarını değerlendirecektir.

En iyi kasiyer vardiyası tespit edildiğinde kurulmuş olan model, "hangi gün, hangi saatte kuyruk uzunluğu ne kadar olacak?" sorusuna cevap verebilecektir. Bu sayede işletmeci seçilen kasiyer vardiyasını uygulandığında hangi zamanlarda ve ne kadar süreliğine kuyruk uzunluğunun belirlenen bir değerin üzerine çıkabileceğini tahmin edebilecek ve önceden müdahale etme imkanı olacaktır. Ya da hangi saatlerde kasiyerlerin fazlası ile boş kalabileceğini tespit edip bu personeli mağaza içinde başka işlere yönlendirebilecektir.

Modelin detaylı bir sonuç üretmesi sayesinde işletmeci, önceden kasa kuyruklarını tahmin edebilmiş ve bu sayede müşteri memnuniyetini arttırmış olacaktır. Aynı zamanda personel yönetimini daha detaylı olarak yapabilmiş ve bordro maliyetlerini minimize etmiş olacaktır. Bu çalışma sayesinde işletmenin kazandıracağı müşteri memnuniyeti ile satış artışı ihtimali yükselmiş ve düşürülen bordro maliyetleri ile işletmenin karlılığına katkı sağlanmış olunacaktır.

1.2. Çalışmanın Genel Mantığı

Bu çalışma, en basit anlamıyla bir kuyruk problemidir. Ancak kullanılacak olan kuyruk sisteminin yapısı gereği matematiksel kuyruk problemi çözüm teknikleri işe yaramamaktadır. Kuyruk sisteminde çok fazla servis istasyonu olması ve her servis istasyonunun performansının birbirinden farklı olması; kuyruğa geliş süreçlerinin haftanın günü ve günün saatine göre değişiklik gösteresi gibi çok fazla faktörün kuyruk sistemini etkilemesi sebebi ile matematiksel olarak sistemi modellemek mümkün değildir. Bu yüzden kuyruk sisteminin MC benzetim tekniği ile modellenmesi gerekmektedir.

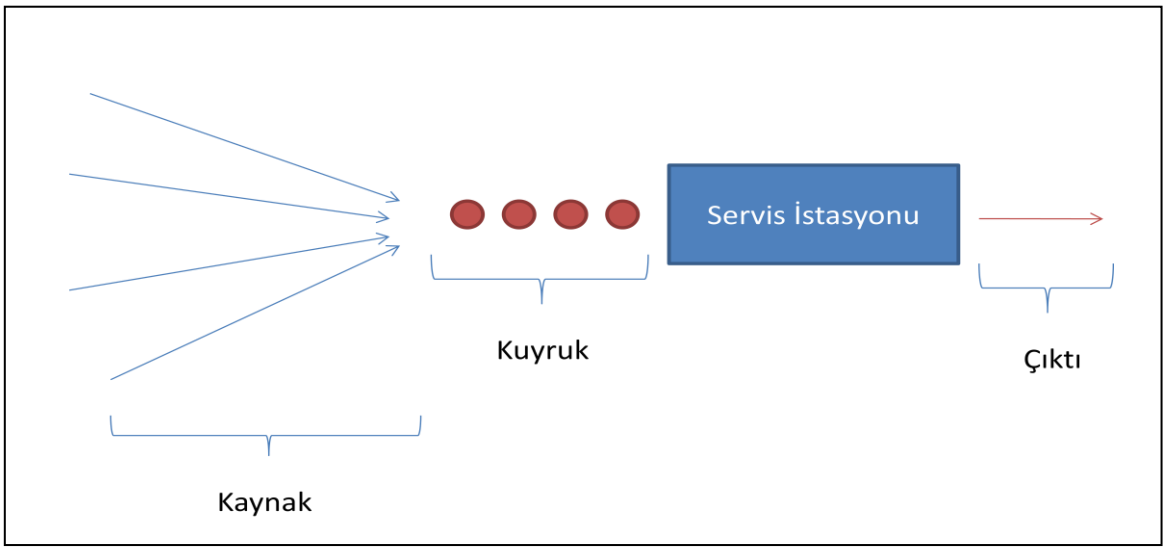
2. KUYRUK TEORİSİ

Kuyruk problemi, bilgisayar ağları ve telekomünikasyon gibi teknik konularda karşımıza çıktığı gibi, gündelik yaşamımızda da market kasaları, toplu taşıma, banka gişeleri, bilet gişeleri ya da trafik sıkışıklığı vb alanlarda karşımıza çıkmaktadır [1]. Kuyruk teorisi ile ilgili ilk çalışma Karl Erlang'a (Kopenhag telefon hatları ile ilgili yaptığı çalışmaya) atfedilmesine rağmen, ilk eser 1907'de yayınlanan "Waiting Times and Number of Calls" başlıklı, telefon hatlarında bekleme süresini ve bekleyen çağrı sayısını azaltmak amaçlı Johannsen'in makalesidir [2].

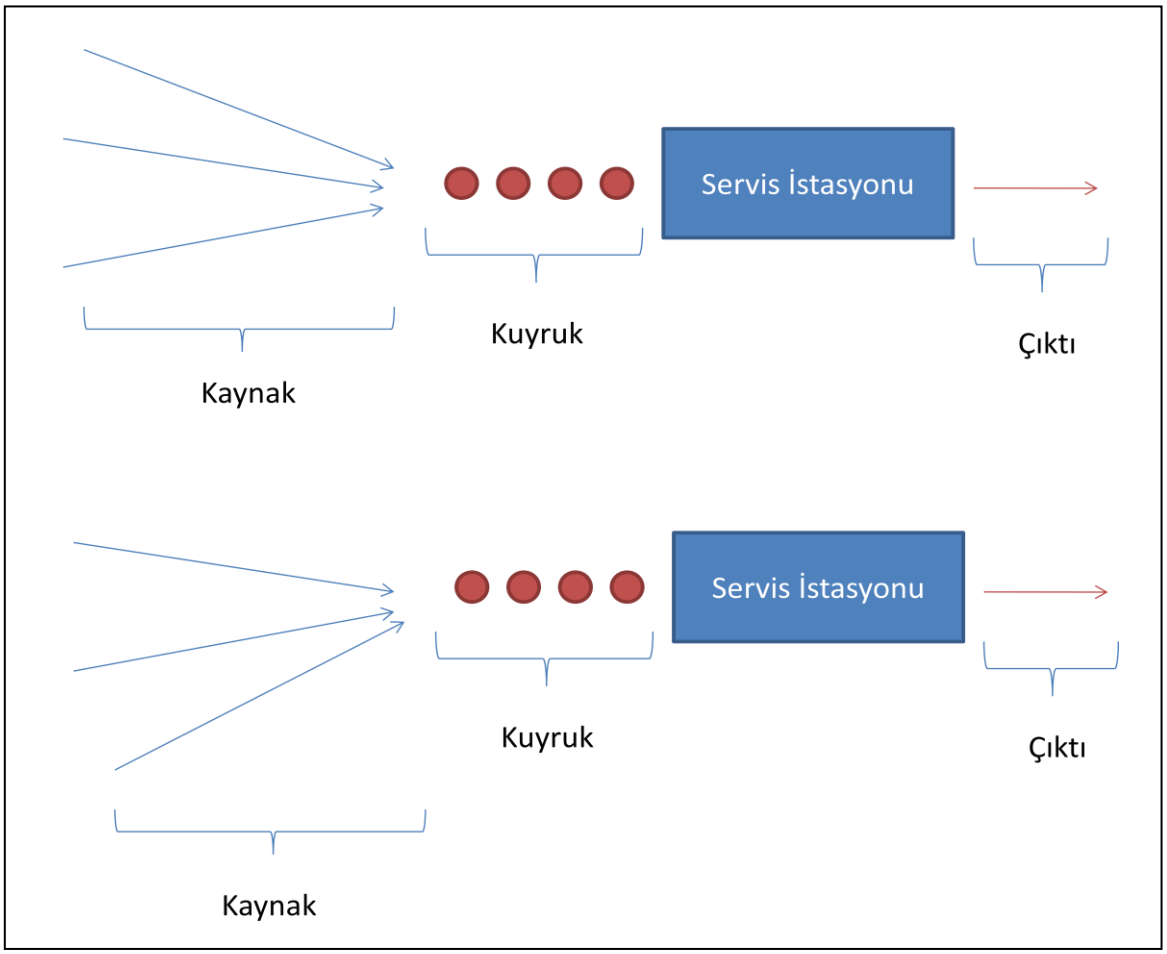
Birçok insan gündelik yaşamını idame ettirebilmek için zorunlu ihtiyaçlarını karşılamak maksadı ile marketlere gitmekte ve market kasalarında kuyrukta beklemek zorunda kalmaktadır. Müşterilerin birçoğu market kasalarına geldiklerinde kuyrukta beklemek zorunda kalmaktadır. Kasaya gelen müşteriler genelde birden çok kasa çalıştığında ve kendi belirledikleri bazı kriterlere göre bir kasa seçmektedir. Bazı durumlarda yeterli kasanın çalışmıyor olması kuyrukta bekleme süresini uzatmakta ve müşteri memnuniyetsizliği oluşturarak satış kaybına neden olabilmektedir. Aynı anda gereğinden fazla çalışan kasiyer fazla maliyet oluşturmaktadır. İşletmelerin müşteri memnuniyeti ile maliyetler arasında bir denge kurması gerekmektedir. Kuyruk teorisinin amacı, kuyruk sisteminin bir modelini oluşturup, boş zamanları ve yoğunlukları tespit edip maliyet ile müşteri memnuniyeti arasındaki dengeyi kurmaktır.

Kuyruk modelleri tek servis istasyonlu ve tek aşamalı olabileceği gibi çok istasyonlu ve çok aşamalı da olabilmektedir. Tek istasyonlu modellerde sadece bir istasyon sırası ile gelen tüm iş taleplerine karşılık vermektedir [1]. Çoklu istasyon modellerinde ise birçok istasyon aynı anda gelen birçok iş talebine karşılık vermektedir. Mesela tek bir kasanın çalıştığı market tek istasyonlu; birden çok kasanın çalıştığı bir market çok istasyonlu kuyruk modelindedir. Aynı mantık ile eğer kuyruktaki iş talepleri tek bir işleme tabi tutuluyor ise tek aşamalı; peş peşe birçok işleme tabi tutuluyor ise çok aşamalı kuyruk modeli olmaktadır.

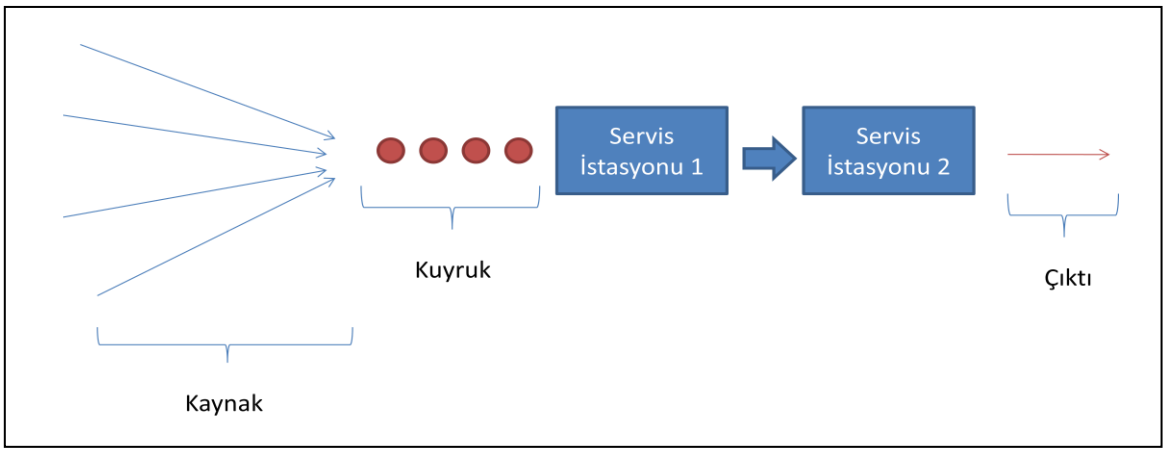
Tek aşamalı çok istasyonlu bir kuyruk modelinde iki tip kuyruk sistemi kullanılabilir. Sistemlerden biri her iş talebinin tek bir kuyrukta bekledikten sonra boşalan ilk istasyona geçmesidir [3]. Diğer sistem ise her bir istasyonun ayrı kuyruklarının olmasıdır.



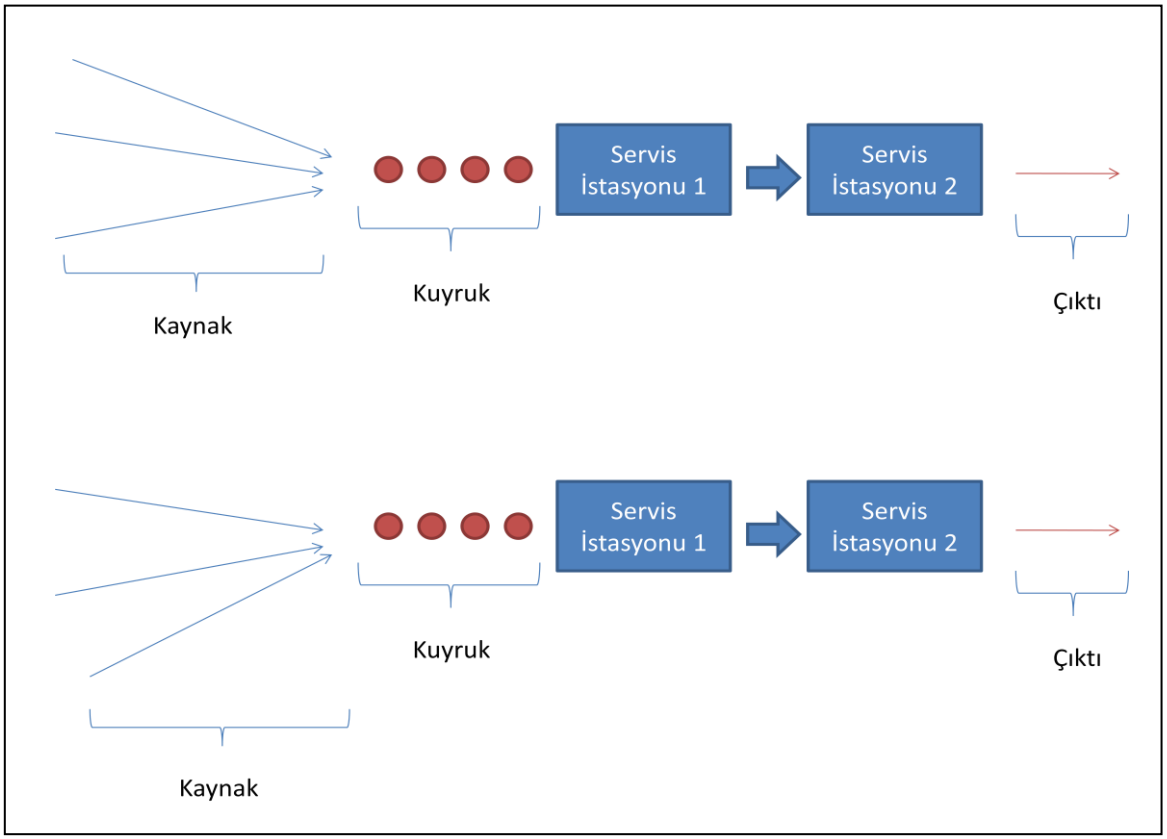
Şekil 1: Tek Aşamalı Tek İstasyonlu Kuyruk Yapısı



Şekil 2: Tek Aşamalı Çok İstasyonlu Kuyruk Yapısı



Şekil 3: Çok Aşamalı Tek İstasyonlu Kuyruk Yapısı



Şekil 4: Çok Aşamalı Çok İstasyonlu Kuyruk Yapısı

Kuyruk sistemini oluşturan bileşenler:

2.1. Kuyruğa Geliş Süreci

Kuyruğa gelen iş talepleri tek bir kaynaktan gelebileceği gibi aynı anda birçok kaynaktan da gelebilir [2]. Ayrıca bu kaynaklar üretim alanında üretimi yapan cihazların bozulma durumunun oluşturduğu kuyruklar gibi sınırlı bir kaynak olabileceği gibi, bir market kasa hattına gelecek müşterilerin oluşturduğu kuyruklar gibi sonsuz bir kaynak da olabilir.

2.2. Servis Mekanizması

Gelen iş talepleri tek bir kuyrukta bekledikten sonra servis istasyonuna geçebileceği gibi her bir servis istasyonunun kendi kuyrukları da olabilir.

2.3. Kuyruk Disiplini

Servis istasyonu gelen iş taleplerini birçok farklı sırada alabilir.

- **FIFO (First In First Out):** İlk giren ilk çıkar prensibi.
- **LIFO (Last In First Out):** Son giren ilk çıkar prensibi.
- **Öncelikli:** İş taleplerinin önem derecelerine göre belirlenen önceliğe göre seçilmesi prensibi.

Kuyruk sistemlerinde ölçülmek istenen birçok performans kriteri olabilir:

- Ortalama kuyruk uzunluğu
- Ortalama kuyrukta bekleme süresi
- Kuyrukta bekleme olasılığı
- Kuyrukta geçirilecek sürenin önceden belirlenmiş bir süreden daha fazla olma olasılığı
- Herhangi biz zamanda kuyruk uzunluğunun önceden belirlenmiş bir Uzunluktan Daha fazla olma olasılığı
- Tüm servis istasyonlarının boşta kalması olasılığı
- Servis istasyonunun ortalama boşta kalma süresi
- Yetersiz servis istasyonu sebebi ile geri dönüş olasılığı

3. MONTE CARLO BENZETİMİ

Doğa bilimleri, sosyal bilimler ya da mühendislik dallarında kullanılan matematiksel modeller, girdileri matematiksel formüller aracılığı ile bir ya da daha fazla çıktıya dönüştürmektedir. MC benzetimi, girdileri tekrarlanan rastgele örneklemeler olarak alıp bazı istatistiksel analizler ile çıktıyı üreten bir benzetim tekniğidir [4]. Bu teknik, ilk olarak 1944 yılında ilk atom bombasının geliştirilmesi esnasında kullanılmıştır. Manhattan Projesi'nde çalışan bilim adamlarının, fisyon ile uranyum atomundan ayrılan nötronun tekrar başka bir bölünmeye sebep olup olmayacağını tespit etmek için çok karmaşık bir denklemi çözmeleri gerekiyordu. Denklem çok karmaşıktı çünkü gerçek bir atomun geometrisini yansıtması gerekiyordu ve sonuçların kesinlikle doğru olması gerekiyordu [5].

Bilim adamları nötronların yörüngelerini takip edebilecekleri bir benzetim oluşturdular. Bu benzetimdeki her bir adımı bilim adamlarının mekanik hesap makineleri ile hesaplamaları gerekiyordu. Bu benzetim sayesinde her bir adımda nötronun absorbe edilme, bombanın dışına kaçması ya da yeni bir bölünmeye sebep olması olasılığını hesaplayabiliyorlardı. Bu benzetim ile ilgili en önemli nokta, elde edilen sonuçların tanımlı istatistiksel özellikler taşıyor olması ve bu sayede gerçek bir zincirleme reaksiyon için gerekli olan parametreleri tahmin etmek için kullanılabilmesiydi [5].

Benzer şekilde de günümüzde MC benzetim tekniği rastgele sayılarla üretilen girdiler kullanılarak elde edilen sonuçlar, parametre tahmininde kullanılmaktadır. MC benzetim tekniğinde, model girdi değişkenlerinin hepsi ya da bir kısmı için belirli bir istatistiksel dağılım belirlenir ve bu değişkenlerin her biri için parametreler bulunur. Bu parametreler ile rastgele sayı üreteçleri kullanılarak girdi değişkenleri için bir örneklem oluşturulur. Bu örneklem modelde girdi değerleri olarak kullanılır [5].

Üretilen girdi değerleri ile benzetim modeli belirli bir sayıda tekrarlanır. Bu sayede çıktı değişkeni için bir örneklem bulunmuş olur. Örnekleme yöntemleri kullanılarak çıktı değişkenine ilişkin dağılım ve parametreler tahmin edilir.

3.1. Rastgele Sayı Üreteçleri (RSÜ)

Rastgele sayı üreteçleri yazılımsal ya da fiziksel olarak art arda, birbirinden bağımsız sayılar üreten ve üretilen bu sayıların bir dizi istatistiksel testten (bağımsızlık testleri gibi) geçmesini sağlayan sayı üreteçleridir [4]. Yazılımsal sayı üreteçlerinin temelinde 0 ile 1 arasında tek düze dağılımdan sayı üreten RSÜ vardır. Üretilen bu sayılar, dönüşüm teknikleri ya da farklı teknikler (reddetme tekniği gibi) kullanılarak istenilen dağılımda

sayılara dönüştürülür. Bu çalışmada Ters Dönüşüm Tekniği ve Knuth algoritması kullanıldığı için sadece bu teknikler detaylandırılacaktır.

3.2. Ters Dönüşüm Tekniği

Ters Dönüşüm Tekniği belirli bir dağılımdan rastgele sayılar üretmek için kullanılan en temel ve basit yoldur. Bu teknikte rastgele sayı üretilmek istenen dağılıma ait olasılık yoğunluk fonksiyonu (sürekli dağılımlar için), olasılık kütle dağılımı (kesikli dağılımlar için) ve 0 ile 1 arasında tek düze dağılımdan üretilmiş rastgele sayılar kullanılır. Bu tekniğin matematiksel açıklaması aşağıdaki gibidir:

X; üretmek istediğimiz dağılıma sahip bir rastgele değişken, X değişkeni için oyf ise F(x) olsun. 0 ile 1 arasında üretilen rastgele sayı U ise:

$F(x) = U \rightarrow F^{-1}(U) = x$ olur.

Yani tersi alınabilir bir oyf fonksiyonuna sahip bir X rastgele değişkeni için 0 ile 1 arasında tek düze dağılımdan üretilmiş değerler $F^{-1}(U) = x$ fonksiyonu kullanılarak, X rastgele değişkeninin dağılımına dönüştürülebilir.

3.3. Üstel Dağılım için Ters Dönüşüm Tekniği

X, λ parametresi ile üstel dağılıma ve U, 0 ile 1 arasında tek düze dağılıma sahip rastgele değişkenler olsun.

$$f(x) = \begin{cases} \lambda e^{-\lambda x}, x \geq 0 \\ 0, x < 0 \end{cases}$$

$$F(x) = \int_{-\infty}^{x} \lambda e^{-\lambda t} dt = \begin{cases} 1 - e^{-\lambda x}, x \geq 0 \\ 0, \quad x < 0 \end{cases}$$

$$U = \begin{cases} 1 - e^{-\lambda x}, x \geq 0 \\ 0, \quad x < 0 \end{cases} \rightarrow X = -\frac{1}{\lambda} ln(1-u)$$

$(1-u)$ değeri de 0 ile 1 arasında rastgele bir sayı olacağı için bunun yerine u yazılabilir.

$$\boldsymbol{X = -\frac{1}{\lambda} ln(u)}$$

3.4. Poisson Dağılımı için Knuth Algoritması

X, poisson dağılımından rastgele değişken ve u, 0 ile 1 arasında tek düze dağılımdan rastgele bir sayı olsun.

Algoritmanın adımları aşağıda verilmiştir:

1. $L \leftarrow e^{-\lambda}$, $k \leftarrow 0$, $p \leftarrow 1$
2. $k \leftarrow k + 1$
3. $p \leftarrow p * u$
4. $p < L$ olana kadar 2. ve 3. adımları tekrarla
5. $x = k - 1$

3.5. MC Benzetim Tekniğinde Uygulanan Adımlar

3.5.1. Statik Modelin Oluşturulması

Tüm MC benzetim teknikleri gerçek senaryoyu yansıtacak, rastgele olmayan bir model oluşturmak ile başlar [4]. Bu modelde girdi değişkenlerinin kendileri arasındaki ya da çıktı değişkeni ile arasındaki ilişkiler tanımlanır. Bu ilişkileri kullanarak MC benzetim tekniği çıktı değişkenine ilişkin değerleri üretir.

3.5.2. Girdi Değişkenleri İçin Dağılımların Tanımlanması

Statik model oluşturulduktan sonra, girdi değişkenlerine ilişkin risk faktörünün de dahil edilebilmesi için, girdi değişkenlerinin her birinin hangi istatistiksel dağılma hangi parametreler ile uyum sağladığını belirlemek gerekir [3]. Bu belirlemenin yapılabilmesi için girdi değişkenlerine ilişkin geçmiş verisine ihtiyaç duyulur.

3.5.3. Rastgele Sayı Üreteçlerinin Oluşturulması

Girdi değişkenleri için dağılımlar belirlendikten sonra bu dağılımlara uygun rastgele sayı kümelerini oluşturacak RSÜ'leri oluşturulur [3]. Bu RSÜ'leri ile üretilen sayı kümeleri statik modelde girdi değerleri olarak kullanılarak, bir çıktı değeri kümesi oluşturulur. Bu işlem belirli bir sayıda tekrar edilerek çıktı değeri için bir değerler kümesi oluşturulur. Bu işlem MC benzetim tekniğinin temelini oluşturmaktadır.

3.5.4. Analiz ve Karar Verme

Çıktı değişkeni için statik model ile üretilen sayılar kümesi üzerinde istatistiksel analizler yapılarak; Bu değişkene ilişkin dağılım, parametre ve güven sınırları hesaplanarak yorumlanır [4].

4. KUYRUK TEORİSİ VE MONTE CARLO BENZETİMİNİN BİR ARADA KULLANILMASI

Kuyruk probleminin yapısı çok basit olabileceği gibi çok karmaşık seviyelere de çıkabilmektedir. Mesela tek aşamalı ve tek istasyonlu kuyruk yapısına sahip otomasyonlu bir üretim bandında, kuyruğu etkileyen faktörler; kaynaktan kuyruğa geliş sıklığı ve üretim bandının işlem süresidir. Böyle bir kuyruk sistemi çok basit bir yapıya sahiptir ve matematiksel yöntemlerle modellenebilir. Ancak tek aşamalı ve çok istasyonlu kuyruk yapısına sahip bir market kasa hattının yapısı çok karmaşıktır. Burada kuyruğu etkileyen faktörler; müşterilerin kasa hattına geliş sıklığı, müşterilerin aldığı ürün sayısı, alınan ürünlerin kasada nasıl bir işleme tabi tutulduğu (sadece barkot okutma, tartma, okutulamayan ürünlerin barkotlarının elle girilmesi vb), kasiyerlerin her işlem için performansı, kasiyerlerin çalıştığı gün ve saatler vb bu faktörler de haftanın gününden ve gün içindeki saatten çok büyük ölçüde etkilenmektedir. Bu şekilde çok karmaşık etkileşime sahip kuyruk sistemini matematiksel bir model ile çözmek çok zordur. Bu yüzden bu tip kuyruk problemlerinin çözümü için MC benzetim tekniği kullanılmaktadır.

Kuyruk problemlerinde MC benzetim tekniği ile incelenmesini zorunlu kılan bir başka neden de, kuyruk sistemlerinin genel olarak stokastik bir özellik göstermelerinden kaynaklanmaktadır. Bu özellikteki bir sistemin işleyişini belirleyebilecek bilgi ve veriler genellikle yeterli değildir. Yeterli bilgi ve veri toplamak (eğer mümkünse bile) zaman ve para kaybına neden olacaktır. Bu nedenle araştırmacılar yapay veri üretme olanağı veren MC benzetim tekniğini tercih etmektedirler. [3]

MC benzetim tekniği ile stokastik özellik gösteren değişkenlerin dağılımlarının belirlenip, bu dağılımlara uygun rastgele sayılar üretilir ve örneklem oluşturularak istenilen değişkenler için veriler üretilir. Kuyruk problemlerinde genelde kaynaktan gelişler ve istasyondaki işlem süreleri stokastik özellikte olduğu için MC benzetim tekniği sık sık başvurulan bir yöntemdir.

Kuyruk problemlerinde MC benzetim tekniğinin kullanılması, çok büyük kolaylık sağlamasının yanı sıra çoğu durumda zorunluluktur. Ancak kuyruk probleminin yapısı ve MC benzetim tekniğindeki modelin karmaşıklığı ya da üretilecek olan verilerin çok olması, MC benzetim tekniğinin sürecini çok uzatabilmektedir. Bu gibi modellerde bilgisayar kaynaklarının verimli kullanılması ve örneklem boyutunun olabildiğince küçük tutulması zorunluluk haline gelmektedir.

4.1. Kuyruk Probleminde MC Benzetim Tekniğinin Aşamaları

Kuyruk probleminde MC benzetim tekniğinin kullanılabilmesi için aşağıdaki aşamaların izlenmesi gerekmektedir.

4.1.1. Sistemin İncelenmesi

Bu aşamada kuyruk sisteminin yapısı incelenip davranışları anlaşılmaya çalışılır. Bu inceleme sırasında sistemin alt bileşenleri, bu alt bileşenlerin etkileşimleri ve sistemin çevresi ile olan ilişkisi belirlenmeye çalışılır [3]. Sistemin incelenmesi sırasında sistemdeki problemlerin belirlenmesi ve giderilmesine yönelik amaçlar geliştirilir. Problemin kusursuz olarak saptanması ve amaçların belirlenmesi için sistemin işleyişine ilişkin pek çok bilgi ve verinin toplanması, derlenmesi ve yorumlanması gerekmektedir.

4.1.2. Modelin Formülasyonu

İncelenen sistem üzerindeki problemlerin belirlenebilmesi ve bu problemlerin giderilebilmesi maksadıyla amaçların belirlenebilmesi için sistemin modellenmesi gerekmektedir. Formüle edilecek model matematiksel ve mantıksal ifadelerden oluşur [3]. Ancak unutulmamalıdır ki, iyi bir MC benzetim modelinin gerçeği temsil edebilmesi, anlaşılabilir ve kolay kullanılabilir olması gerekmektedir. Bir MC benzetimi bu kapsamda değerlendirildiğinde, sistemin tamamının modellenmesi yerine, (eğer mümkünse) sistemdeki sadece problemli ya da incelenmek istenilen sürecin modellenmesi ile model basitleştirilmelidir.

4.1.3. Modelin Mantığının Doğrulanması ve Geçerliliğinin Tespit Edilmesi

Bilgisayarda programı yazılmış modelin istenilen şekilde çalışıp çalışmadığı ve modelin gerçeği yansıtıp yansıtmadığı yani geçerli olup olmadığı tespit edilmelidir. Eğer modelin geçerli olmadığı tespit edilirse sonraki adımlara geçilmez ve modelin geçerliliğini bozanın bilgisayar kodlamasındaki hata mı yoksa modelin kuruluş mantığı mı olduğu bulunarak; bu sorun çözülmelidir [3].

4.1.4. Deneylerin Planlanması (Modelin Uygulanması)

Modelin geçerli olduğu anlaşıldıktan sonra, karar seçeneklerinin değerlendirilebilmesi için model çalıştırılır. Ancak model bir örneklem üzerinde çalışıp, sonucu da örneklem olarak ürettiği için istatistiksel hatalar barındırmaktadır [3]. Bu yüzden modelin ürettiği örneklemler üzerinden hesaplanan istatistikler yardımı ile parametreler tahmin edilmeye ve istatistiksel deney hatalarının makul bir seviyede kalması sağlanmaya çalışılır.

4.1.5. Sonuçların Analizi ve Yorumlanması

Model ile yapılan deneyler sonucunda elde edilen veriler, modelin amacına yönelik olarak değerlendirilir ve yorumlanır. Bulunan sonuçların gerçeği en iyi şekilde yansıtabilmesi için modelin yeterli sayıda tekrar edilmesi ve iyi bir örneklemin oluşturulması gerekmektedir. İyi bir örneklemin oluşturulabilmesi için gerekli tekrar sayısı modelin başından hesaplanıp sabit bir değer olarak kullanılabileceği gibi, model çalıştırılırken sonuçlara ilişkin bazı istatistiklerin belirli kriterleri sağlayacağı şekilde dinamik olarak da belirlenebilir.

5. TABU ARAMA ALGORİTMASI (TA)

Tabu arama algoritması Glover tarafından 1986 yılında geliştirilmiştir [6]. TA algoritması yerel optimum tuzaklarından kaçarak global optimumu bulan yüksek seviyeli bir sezgisel yöntemdir [7]. Birçok farklı optimizasyon probleminde kullanılabilecek olan bu algoritma, optimal ya da optimale yakın çözümler üretebilmek için esnek bir yapıya sahiptir.

TA algoritmasının genel çalışma yapısı, bir başlangıç çözümü ürettikten sonra, bu çözümü belirlenmiş dönüşüm teknikleri kullanarak optimal çözüme ulaştırmaya çalışmaktır [7]. TA algoritmasında kullanılacak dönüşüm teknikleri değişken değerini değiştirme, eleman eklemek ya da çıkartmak, iki çözüm arasında değerleri değiş tokuş etmek sayılabilir. TA algoritmasının bu esnek yapısı yeni çözümleri üretmek ve optimal çözüme ulaşmak için onu çok etkin bir hale getirmektedir ve bu esneklik sayesinde çok farklı problem türlerine uygulanabilmektedir.

TA algoritması daha ilk geliştirme aşamasında bile birçok optimizasyon probleminde kendinden önceki yöntemlere göre çok daha başarılı sonuçlar üretmiştir.

TA algoritmasının kullanıldığı bazı uygulamalar:

- Employee scheduling (Personel çizelgeleme)
- Character recognition (Karakter tanımlama)
- Space planning and architectural design (Alan planlaması ve mimari tasarım)
- Job shop scheduleing (İş akış çizelgeleme)
- Machine scheduling (Makine çizelgeleme)
- Nonlinear covering (Doğrusal olmayan kaplama)

TA algoritmasının farklı çeşitlilikte problemin optimal çözümünü ya da optimale yakın çözümünü başarılı bir şekilde bulabilmesi TA algoritmasının esnek yapısı ve hafıza yapısı sayesinde mümkün olabilmektedir [8]. TA algoritmasında genellikle iki farklı hafıza yapısı vardır. Kısa süreli hafıza ve uzun süreli hafıza.

TA algoritması, kısa süreli hafızası sayesinde, yeni çözüm üretmek için geçilebilecek en iyi hamlenin seçilmesini sağlar ya da bazı çözümlerin üretilmesini engeller. Bu engellemeler tabu olarak adlandırılır ve programın aynı çözümleri tekrar tekrar üretmesine engel olur. Tabuların öncelikli görevi yöntemin yerel optimallerden kurtularak global optimale ulaşmasını sağlamaktır. Ancak bazı durumlarda bu tabuların yıkılması gerekebilir. Bu sayede TA algoritmasının yeni çözüm üretememesi durumuna engel olunur. Programın tek bir çözümde sıkışmasının dışında, tabuların belirli bir sürede de yıkılması gerekir. Belirli bir iterasyon sayısına ulaşıldığında ilk giren ilk çıkar prensibine göre belirlenmiş tabu yıkılarak tekrar çözüm olarak seçilebilir. Bu sayede TA algoritmasının tek bir çözümde sıkışması önlenmiş olur.

TA algoritmasının temelini bu kısa süreli hafıza oluşturmaktadır. Kısa süreli hafıza sayesinde TA algoritması, mevcut çözümün çevresinde dolaşarak sanki bir tepeye tırmanıyormuş gibi yavaş yavaş optimal çözüme yaklaşmayı hedefler.

TA algoritması yerel optimalden kaçınabilmesi için belirli aralıklarla mevcut çözümden uzaklaşmaya çalışır. Bu şekilde TA algoritması tüm çözüm uzayında birçok farklı noktada detaylı arama yaparak global optimal çözüme ya da global optimum çözüme en yakın çözüme ulaşır. Arama süreci boyunca TA algoritması bulduğu en iyi çözümü uzun süreli hafızaya alır ve üretilen tüm çözümler bu en iyi çözüm ile kıyaslanarak değerlendirilir.

TA algoritmasının performansı direk olarak çözümü değiştiren formulasyona bağlıdır. Çözümde yapılacak olan değişikliklerin yönü ve boyutu bu algoritmadaki en kritik noktadır. Doğru belirlenmemiş bir değiştirme fonksiyonu TA algoritmasının optimal çözüme ulaşamamasındaki en büyük nedendir.

TA algoritmasının ne zaman duracağını belirlemek de, hem performans hem de çözüm kalitesi bakımından önemlidir. Belirli bir iterasyon sayısına ulaşıldığında, daha iyi bir çözüm bulunamadığında ya da optimal çözümü bilinen bir problem için optimal çözüme ulaşıldığında TA algoritması durdurulur.

5.1. TA Algoritmasının Adımları

1. Başlangıç çözümünü belirle. Bu çözümü mevcut çözüm ve en iyi çözüm olarak hafızaya al.
2. Belirlenen değiştirme fonksiyonu ile geçilebilecek komşu çözümleri bul.
    a. Tabu olmayan ya da tabu olsa bile tabu yıkma kriterlerini sağlayan bir komşu çözümü seç.
    b. Mevcut çözümden yeni çözüme geçişi tabu olarak belirle.
    c. Yeni çözüm o ana kadarki en iyi çözüm ise yeni çözümü en iyi çözüm olarak belirle.
3. Durdurma ölçütü sağlanana kadar 2. adımı tekrarla.

6. TABU ARAMA ALGORİTMASI VE MC BENZETİM TEKNİĞİNİN BİRLİKTE KULLANILMASI

TA algoritması, optimum sonuca ulaşmak için bir amaç fonksiyonuna gereksinim duyar. Üretilen sonuçlar bu amaç fonksiyonu ile değerlendirildikten sonra, bu sonucun iyi bir çözüm olup olmadığı kontrol edilir. Amaç fonksiyonu, TA algoritmasının ürettiği sonuçları girdi olarak kullanarak bir performans kriteri oluşturur. Ancak bazı durumlarda TA algoritmasının ürettiği sonuçları ve ulaşılmak istenen performans kriterini matematiksel olarak birbirleri ile ilişkilendirmek ve formülize etmek çok zor ya da imkansızdır. Üretilen sonuçların girdi olarak kullanılacağı modelin yapısının çok karmaşık olması ya da modelin stokastik bir yapıya sahip olması, matematiksel model kullanımı yerine MC benzetim tekniğinin tercih edilmesi işlemleri kolaylaştırmanın yanı sıra çoğu durumda da zorunluluk olmaktadır.

TA algoritması, birbirinden çok farklı sistemler için çizelgeleme problemini başarılı bir şekilde çözebilmektedir. TA algoritması tek başına çizelgeleme probleminin çözümü için kullanılabileceği gibi, MC benzetim tekniği ile birlikte de kullanılabilmektedir. Ancak yapılan çalışmalarda bulunan sonuçlar, çalışması gereken personel ya da makine sayısını belirtmişlerdir. Yani sistemlerdeki personelin ya da makinelerin birbirleri ile aynı

performansa sahip olduğu varsayımı ile sonuçlar üretilmiştir. TA algoritmasının çizelgeleme için kullanıldığı bazı çalışmalar aşağıdaki gibidir:

- The Flow Shop Scheduling Problem (Widmer and Hertz, 1989; Kim, 1993; Taillard, 1990; Reeves, 1993; Adenso-Diaz, 1992)
- The Job Scheduling Problem (Widmer, 1991; Barnes ve Chambers, 1995; Sun et al. 1995)
- The Identical Paralel Machine Scheduling Problem (Barnes ve Laguna, 1993; Hübsher ve Glover, 1994; Laguna ve Gonzalez Velarde, 1991)

Bu makaleler, TA algoritmasının ne çeşitlilikte çizelgeleme problemlerini çözebildiğini incelemektedirler [9].

İş çizelgeleme problemlerinde, yapılması gereken bir takım işler ve bu işleri yapacak bir takım makineler vardır. Bu işler makinelerde farklı süreçlere tabi tutularak tamamlanmaktadır. İş çizelgeleme problemlerinin amacı bu işlerin en kısa sürede yapılabilmesi için hangi sırada ve hangi makinede yapılması gerektiğini belirlemektir. İş çizelgeleme problemlerinde genellikle matematiksel modeller kullanılsa da MC Benzetim tekniğinin de kullanılabilmektedir.

TA algoritması ve MC benzetim tekniğin bir arada kullanıldığı en önemli çalışmalardan birisi de Beck ve Wilson'un 2007 yılında yayınladıkları "Proactive Algorithms for Job Shop Scheduling with Probabilistic Durations" isimli çalışmalarıdır. Klasik olarak çizelgeleme problemlerinde her bir aktivitenin bilinen belirli bir zamanda tamamlandığı varsayımı kullanılmaktadır. Beck ve Wilson bu makalelerinde bu varsayımı biraz esneterek, her bir aktivitenin, ortalama ve varyansı bilinen bir rastgele değişken olduğu varsayımını incelemişlerdir. Ancak bu yaklaşımda tüm işlerin aynı parametreler ile aynı dağılıma sahip olduğu varsayımı vardır. Yani işlerin tamamlanma sürelerini etkileyecek bir başka dış etken olmadığı varsayılmaktadır.

İncelenen çalışmalar sonunda, detaylı bir model ile benzetimi yapılmış bir kuyruk probleminin tabu arama algoritması ile çözümlenmesi ve sonucunun da aynı detayda üretilmesi için yeni bir çalışma yapılması gerektiği anlaşılmaktadır. Yapılan bu çalışmada, TA algoritması ve MC benzetim tekniği bir arada kullanılarak bir market kasa hattında en kısa kuyruk uzunluğuna ulaşmak hedeflenmektedir. Ancak bu sistem, performansları birbirinden farklı ve ölçülebilir olan servis istasyonlarına sahip (çağrı merkezleri, birden fazla vardiyası olan banka şubeleri vb) tüm kuyruk sistemleri için kullanılabilir.

Bir marketteki kasa kuyruk probleminin çözümünün, hem kuyruk modelinin stokastik bir yapıda olması hem de çok karmaşık olması sebebi ile MC benzetim tekniği ile değerlendirilmesi zorunluluk haline gelmektedir. MC Benzetim tekniğini amaç fonksiyonu olarak düşünürsek; bu fonksiyonun girdisi TA algoritmasının üreteceği kasiyer vardiyası ve çıktısı ise ortalama kasa kuyruk uzunluğu olacaktır. Eğer tüm kasiyerlerin birbirleri ile aynı performansa sahip olduğunu düşünüp, kasiyer vardiyasını "hangi gün hangi saatte kaç kasiyer olması gerektiği" şeklinde yazacak olursak; bu model karmaşık olsa bile matematiksel olarak modellenebilir. Ancak gerçek dünyada insanların performanslarının birbirinden farklı olması kaçınılmazdır. Bu yüzden kasiyer vardiyasının kişi bazlı olarak değerlendirilmesi gerekmektedir. Yani "hangi kasiyer, hangi gün, kaç saat çalışacak" sorusuna cevap verebilecek bir vardiya üretilmesi gerekmektedir. Bu şekilde üretilmiş bir vardiyayı girdi olarak kullanacak sisteminde aynı şekilde kasiyer bazında detaylandırılmış olması gerekmektedir. Yani her bir kasiyerin kuyruk sistemindeki işlem süresinin kendi performansına göre stokastik olarak belirlenmesi gerekmektedir. Kasiyer performanslarının da stokastik bir yapıya sahip olduğunu göz önüne aldığımızda, bu sistemin matematiksel olarak modellenmesi neredeyse imkansız olmaktadır.

Çalışmada kullanılacak programın yazılması için uygulanacak aşamalar aşağıdaki gibidir:

- Uygulanacak TA algoritmasının geliştirilmesi
    - TA algoritması için girdi verileri
    - TA algoritmasında uygulanacak kısıtlar
    - TA algoritmasının ilk vardiyayı üretmesi
    - Çözümün değiştirilmesi
    - Çözümün MC benzetim tekniğine aktarılması
- Kuyruk yapısının incelenmesi
- Kuyruk sisteminin MC benzetim tekniği ile modellenmesi
- Oluşturulan modele uygun veri yapısının hazırlanması
- Modelin ürettiği verilerin üzerinden performans kriterlerinin hesaplanması
- Performans kriterlerinin TA algoritması tarafından değerlendirilmesi

7. TA ALGORİTMASININ KUYRUK PROBLEMİNE UYGULANASI

7.1. Tabu Arama Algoritması

7.1.1. TA Algoritması İçin Girdi Verileri

- **M**ağazanın **A**çılış **Saati** (MAS)
- **M**ağazanın **K**apanış **S**aati (MKS)
- Mağazanın toplam **K**aç **S**aat **Ç**alıştığı (KSÇ)
- Mağazada **K**aç **K**asiyer **Ç**alışıyor (KKÇ)
- TA algoritması için **M**aksimum **İ**terasyon **S**ayısı (MİS)
- Kasiyerler **H**angi **G**ünler **Ç**alışıyor ve haftalık izinleri ne zaman ($HGÇ_{i,j}$)
- **P**ersonelin **E**n **E**rken vardiyaya başlanabilecek saat ($PEE_{i,j}$)
    - i=1,2,…,7 (gün indisi)
    - j=1,2,…,KKÇ (kasiyer indisi)
    - k=1,2,…,KSÇ (saat indisi)

7.1.2. TA Algoritmasında Uygulanacak Kısıtlar

Oluşturulacak sistem, bir market kasa hattı için kasiyer vardiyalarını oluşturacaktır. Marketin çalışabilmesi için gerekli bazı kısıtların yanı sıra, 4857 sayılı İş Kanunu gibi kesinlikle esnetilemez bazı kısıtlar da bulunmaktadır.

Marketin çalışabilmesi için gerekli kısıt; haftanın her günü ve mağazanın açık olduğu her saatte en az bir kasiyer çalışmalıdır.

4857 Sayılı İş Kanunu gereği kısıtlar:

- Her personelin haftada en az 24 saatlik bir haftalık izni olmalıdır.
- Bir personel üst üste en fazla 6 gün çalışabilir.
- Personelin iki vardiyası arasında en az 11 saat olmalıdır.

İlk iki kısıtın sağlanabilmesi için kasiyerlerin haftalık izinlerinin sabit bir gün olduğu ve bunun önceden belirlenmiş olduğu varsayımı yapılmıştır.

7.1.3. TA Algoritmasının İlk Vardiyayı Üretmesi ve Çözüm Değiştirme Mantığı

Kullanılacak değişkenler:

1. **K**asiyer **V**ardiyası ($KV_{i,j}$)
2. **H**angi **S**aatler **Ç**alışıyor ($HSÇ_{i,j,k}$)
3. **M**ağazada **Ç**alışan **K**asiyer **S**ayısı ($MÇKS_{i,k}$)

İlk vardiyanın üretilmesi için kullanılan algoritma:

1 Adım. $HGÇ_{i,j}$; $PEE_{1,j}$; KKÇ ve KSÇ değerlerini oku

2 Adım. $\forall\ MÇKS_{i,k} = 0$

3 Adım. $\forall\ HSÇ_{i,j,k} = 0$

4 Adım. $i = 1, j = 1$

5 Adım. Eğer $HGÇ_{i,j} = 0$ ise $KV_{i,j} = -1$ ve **15. Adıma** git

6 Adım. u, 0 ile 1 arasında tek düze dağılımdan rastgele bir sayı üret

7 Adım. $KV_{i,j} = PEE_j + (KSÇ - 8 - PEE_j) * u$

8 Adım. $k = KV_{i,j}$

9 Adım. $HSÇ_{i,j,k} = 1$

10 Adım. $MÇKS_{i,k} = MÇKS_{i,k} + 1$

11 Adım. Eğer $KV_{i,j} \leq 6$ ise $PEE_{i+1,j} = 1$

12 Adım. Eğer $KV_{i,j} > 6$ ise $PEE_{i+1,j} = KV_{i,j} - 5$

13 Adım. $k = k + 1$

14 Adım. Eğer $k < KV_{i,j} + 8$ ise **9. Adıma** git

15 Adım. $i = i + 1$

16 Adım. Eğer $i < 7$ ise **5. Adıma** git

17 Adım. $j = j + 1$

18 Adım. Eğer $j <$ KKÇ ise **5. Adıma** git

Bu algoritma üretilen kasiyer vardiyası 4857 sayılı İş Kanunu kısıtlarını sağlamaktadır ancak mağazanın çalışabilmesi için gerekli kısıdı sağlamama olasılığı vardır. Bu yüzden bu kısıdın kontrol edilip vardiyanın bu kısıdı sağlaması sağlanmalıdır. $MÇKS_{i,k}$ değerlerinden herhangi biri 0 ise bu kısıt sağlanmıyor demektir. Bu kısıtın sağlanması için 0 olan $MÇKS_{i,k}$ değerindeki k. saatte en yakın kasiyer vardiyası bulunup, k. saate kaydırılır. Daha sonra $MÇKS_{i,k}$; $PEE_{i+1,j}$ ve $HSÇ_{i,j,k}$ değerleri yeni kasiyer vardiyasına göre güncellenir.

7.1.4. Çözümün Değiştirilmesi

TA algoritmasında çözümü değiştirecek olan algoritma, optimum çözümün bulunabilmesindeki en önemli adımdır. Bu algoritmanın özelliğine göre TA algoritması optimum çözüme ulaşabilmek için üretilen çözümleri değiştirerek, her bir çözümü belirli performans kriterine göre değerlendirmektedir.

Çözüm değiştirme algoritmasında kullanılan değişkenler:

1. **K**uyruk **U**zunluğu ($KU_{i,k}$)
2. Vardiya değiştirme olasılığı (p)
3. Vardiyayı erkene alma olasılığı (p1)
4. Vardiyayı erteleme olasılığı (p2)
5. İstenen En Uzun Kuyruk Uzunluğu (n)
6. **V**ardiya başlamadan 1 saat **ö**nceki **k**uyruk **u**zunluğu (VÖKU)
7. **V**ardiya bittikten 1 saat **s**onraki **k**uyruk **u**zunluğu (VSKU)

Çözüm değiştirme algoritması aşağıdaki gibidir:

1 Adım. n değerini oku

2 Adım. $KU_{i,k}$ değerlerini MC Benzetim tekniğinden al

3 Adım. Eğer $\forall\ KU_{i,k} \leq n$ ise optimum sonuca ulaşıldı algoritmayı sonlandır

4 Adım. $i = 1$, $j = 1$

5 Adım. u, 0 ile 1 arasında tek düze dağılımdan rastgele bir sayı üret

6 Adım. p1 = VÖKU / (VÖKU + VSKU)

7 Adım. p2 = VSKU / (VÖKU + VSKU)

8 Adım. Eğer $u < 2 * p$ ise 8.1, 8.2 ve 8.3 adımlarını uygula

8.1 Adım Eğer $KV_{i,j} > PEE_{i,j}$ ve $u < p * p1$ ise $KV_{i,j} = KV_{i,j} - 1$

8.2 Adım Eğer $KV_{i,j} < KSÇ - 8$ ve $p * p1 < u < 2 * p$ ise $KV_{i,j} = KV_{i,j} + 1$

8.3 Adım $MÇKS_{i,k}$; $PEE_{i+1,j}$ ve $HSÇ_{i,j,k}$ değerlerini yeni $KV_{i,j}$ 'ye göre güncelle

9 Adım. $j = j + 1$

10 Adım. Eğer $j \leq KKÇ$ ise **5. Adıma** Git

11 Adım. $i = i + 1$

12 Adım. Eğer $i \leq 7$ ise **5. Adıma** Git

13 Adım. Eğer üretilen yeni vardiya tabu listesinde var ise **4. Adıma** git

14 Adım. Eğer üretilen yeni vardiya tabu değil ise yeni vardiya ile MC Benzetim'ini tekrar çalıştır.

7.2. Kuyruk Yapısının İncelenmesi

Bir marketteki kasa kuyruk sistemi sonsuz kaynaklı, tek aşamalı ve çok istasyonlu kuyruk yapısına sahiptir. Böyle yapıya sahip bir kuyruk sistemi çok karmaşıktır ve birçok faktörden etkilenmektedir. Bu etkileşimlerin doğru bir şekilde anlaşılması ve modele entegre edilmesi, kasiyer vardiyası için doğru bir performans kriterine ulaşabilmek için çok büyük önem arz etmektedir. Kuyruk sistemini etkileyen faktörler:

- **M**üşterilerin kasa hattına **g**eliş **s**ıklıkları. ($KGS_{i,k}$)
- Müşterilerin **a**lmış olduğu **ü**rün **s**ayıları. ($AÜS_{i,k}$)
    - Bu iki değer haftanın günü ve gün içindeki saatlere göre değişim göstermektedir.
- **K**asiyer **v**ardiyaları ($KV_{i,j}$)
- Kasiyerlerin **ü**rün **o**kutma **p**erformansları ($ÜOP_j$)
- Kasiyerlerin **ö**deme **a**lma **p**erformansları ($ÖAP_j$)

Burada kasiyer vardiyaları hariç diğer tüm faktörlerin stokastik bir yapıda olduğu ve hepsinin önceden tanımlanmış bir istatistiksel dağılıma uyum sağladığı varsayılmaktadır. Bu dağılımların tespiti için ek bir çalışma yapılmamış olup, teorik dağılımların doğru olduğu kabul edilmiştir. Faktörlerin hangi dağılımlara sahip olduğu aşağıdaki gibidir:

- $KGS_{i,k} \sim$ Üstel($\lambda 1_{i,k}$)
- $AÜS_{i,k} \sim$ Poisson($\lambda 2_{i,k}$)
- $ÜOP_j \sim$ Üstel($\lambda 3_j$)
- $ÖAP_j \sim$ Üstel($\lambda 4_j$)

7.3. Kuyruk Sisteminin MC Benzetim Tekniği İle Modellenmesi

Kuyruk sisteminin yapısı tespit edilip sistemi etkileyen faktörlere ilişkin gerekli dağılımlar belirlendikten sonra, bu yapıyı en iyi temsil edecek şekilde MC benzetim tekniği ile modellenmesi gerekmektedir. MC benzetim tekniğindeki en kritik unsurlardan birisi ise benzetimin kaç tekrar yapacağıdır. Tekrar sayısı önceden belirlenmiş sabit bir sayı olabileceği gibi, MC benzetim tekniğinin ürettiği sonuçlara göre dinamik bir yapıda da olabilir.

Çalışmada hazırlanan program düşük performanstaki bir bilgisayarda çalıştırıldığı için tekrar sayısı 10 olarak alınmıştır. 2.2GHz güce sahip bir Intel T6600 işlemci ile 10 tekrar 80 saniye sürmektedir. TA algoritmasının da ortalama olarak 50 farklı vardiya üreteceği

düşünülürse programın işlemi tamamlaması 4000 saniye yani 66 dakika sürecektir. Tekrar sayısını 100 olarak aldığımızda ise işlemin tamamlanması 11 saat sürecektir. Bu sistemin bir marketteki server üzerinde çalışacağını ve server işlemci kapasitelerinin çok çok üst seviyelerde olduğu göz önüne alındığında programın süresine bakılmaksızın tekrar sayısı belirlenebilecektir.

MC benzetim tekniğinin dinamik olarak tekrar sayısını belirlemesi için kuyruk sistemindeki en kritik değişkenin tespit edilmesi gerekmektedir. Bu çalışma için en kritik değer, kuyruğu en çok etkileyecek faktörlerden biri olması sebebi ile, müşterilerin geliş sıklıklarıdır. Üretilen rastgele müşterilerin haftanın günü ve günün saatine göre sayılarının tespit edilip, bu sayılara ilişkin güven aralıklarının belirli bir seviyenin altında kalmasını sağlayacak şekilde tekrar sayısı belirlenir.

MC benzetim tekniğinde kullanılan değişkenler:

1. L = 1,2,…,8 (müşteri değişkeni indisi)
    - L = 1 ise Müşterinin kasa hattına geliş anı
    - L = 2 ise Müşterinin aldığı ürün sayısı
    - L = 3 ise Müşterinin hangi kasiyere gideceği
    - L = 4 ise Müşterinin kasada geçirdiği süre
    - L = 5 ise Müşterinin işlemlerine başlanma anı
    - L = 6 ise Müşteriden önce kasada kaç kişi olduğu
    - L = 7 ise Müşterinin kasadan ayrılış anı
    - L = 8 ise Müşterinin önündeki müşteri
2. m = 1,2,…,10000 (Gün içinde gelen müşteri için verilen sıra numarası)
3. Döngü = 1,2,…,10 (MC Benzetim tekniğinin tekrar indisi)
4. Mağazaya **G**elen **M**üşteriler ($GM_{m,L,i}$)
5. **O**rtalama **K**uyruk **U**zunluğu ($OKU_{i,k}$)
6. **S**aatlik **M**üşteri **S**ayıları ($SMS_{i,k}$)
7. **S**aatlik **M**üşteri **S**ayıları **V**aryansı ($SMSV_{i,k}$)

MC Benzetim tekniğinin algoritması aşağıdaki gibidir:

1 Adım. $\forall\ OKU_{i,k} = 0$

2 Adım. $\forall\ SMS_{i,k} = 0$

3 Adım. $\forall\ SMSV_{i,k} = 0$

4 Adım. Döngü = 1

5 Adım. i = 1

6 Adım. m = 1

7 Adım. $GM_{m,1,i}$ = Üstel($KGS_{i,k}$) dağılımından rastgele bir sayı

8 Adım. Eğer $GM_{m,1,i}$ > MKS ise **22. Adıma** git

9 Adım. $GM_{m,2,i}$ = Poisson($AÜS_{i,k}$) dağılımından rastgele bir sayı

10 Adım. $GM_{m,3,i}$ = O anda çalışan kasiyerlerden rastgele birini seç

11 Adım. Süre = 0

12 Adım. ∀ $GM_{m,2,i}$ için Süre = Süre + Üstel($ÜOP_j$) dağılımından rastgele bir sayı

13 Adım. Süre = Süre + Üstel($ÖAP_j$) dağılımından rastgele bir sayı

14 Adım. $GM_{m,4,i}$ = Süre

15 Adım. Eğer i'inci günde gelen m'inci müşteri, $GM_{m,3,i}$ kasiyerine gelen ilk müşteri ise

15.1. $GM_{m,5,i}$ = $GM_{m,1,i}$

15.2. $GM_{m,6,i}$ = 0

15.3. $GM_{m,8,i}$ = 0

16 Adım. Eğer i'inci günde gelen m'inci müşteri, $GM_{m,3,i}$ kasiyerine gelen ilk müşteri değil ise

16.1. $GM_{m,8,i}$ = $GM_{m,3,i}$ kasiyerine m'inci müşteriden bir önceki müşterinin sıra numarası

16.2. $GM_{m,5,i}$ = $GM_{x,7,i}$

16.3. $GM_{m,6,i}$ = $GM_{m,1,i}$ değerinden büyük $GM_{m,7,i}$ değerine sahip; $GM_{m,3,i}$ kasiyerindeki müşteri sayısı

17 Adım. $GM_{m,7,i}$ = $GM_{m,5,i}$ + $GM_{m,4,i}$

18 Adım. $OKU_{i,k}$ değerini hesapla

19 Adım. $SMS_{i,k}$ değerini hesapla

20 Adım. $SMSV_{i,k}$ değerini hesapla

21 Adım. Eğer $GM_{m,1,i}$ < MKS ise m = m + 1 ve **7. Adıma** git

22 Adım. i = i + 1

23 Adım. i ≤ 7 ise **6. Adıma** git

24 Adım. döngü = döngü + 1

25 Adım. Eğer döngü ≤ 10 ise **5. Adıma** git

8. PROGRAMIN YAZILMASI

TA algoritması ve MC benzetim tekniği yapıları gereği çok fazla tekrara ve döngüye sahip bir çalışma sistemleri vardır. Hem TA algoritması art arda iterasyonlar ile yeni vardiya üretecek hem de MC benzetim tekniği üretilen vardiyalar ile kuyruk sistemini rastgele sayılar ile defalarca canlandırarak sonuca ulaşmaya çalışacaktır. Bu iki sistemin de en temel özelliği sürekli olarak tekrar üzerine çalışmalarıdır. Bu yüzden çok karmaşık sistemler için bu iki sistem bir arada kullanılmak istendiğine karşılaşılan en büyük problem, sürecin çok uzun sürmesidir. Süreci kısaltabilmek adına yazılacak kodların çok hızlı çalışabilir yapıda olmasının yanı sıra, kodların yazılacağı programlama dili de çok önemlidir. Ancak burada önemli bir diğer kısıt ise programın gerçek hayatta da kullanılabilir olmasıdır. Bu programın, herkes tarafından kolayca kullanılabilir olması için veri giriş ve çıkışının çok basit olması gerekmektedir.

Bu sistemi (zamansal olarak) en çok zorlayacak olan adım MC benzetim tekniğidir. Bu yüzden bir market kasa hattı modeli MC benzetim tekniği ile ilk olarak Matlab ile hazırlandı. Ancak Matlab ile hazırlanan bu MC benzetim tekniğinin çok uzun sürede sonuç üretebilmesi ve gerçek hayatta bir markette Matlab programı ve bunu kullanabilecek bir personelin olma ihtimalinin düşük olması nedeni ile Matlab ile sistemin kurulmasından vazgeçilmiştir. MC benzetim modeli Microsoft Excel programı ve Makro'lar kullanılarak Visual Basic kodları ile yazılarak performansı değerlendirilmiştir. Sonucu fark edilebilir bir şekilde daha hızlı üretmesi ve veri giriş ve çıkışının birçok kişi tarafından kolayca anlaşılabilir ve yönetilebilir olması sebebi ile sistem Microsoft Excel programı ve Makro'lar kullanılarak Visual Basic kodları ile yazılmıştır.

Çalışma kapsamında hazırlanan program, üç ayrı çalışma sayfasında girdi değerlerini alıp sonucunda da iki ayrı çalışma sayfasına çıktı değerlerini oluşturmaktadır. Kodlar toplamda iki farklı modül olarak yazılmıştır. Birinci modülde TA algoritması ve ikinci modülde MC benzetim tekniği olacak şekilde ayrı ayrı modüllere yazılarak kodların daha basit ve kolay anlaşılabilir olması sağlanmıştır.

9. UYGULAMA

Çalışma kapsamında hazırlanan programın gerçeğe yakın verilerle test edilmesi gerekmektedir. Bunun için rastgele üretilmiş ancak gerçek bir mağaza verilerine yakın değerler ile; günde 14 saat açık kalan, toplam 20 kasiyeri olan bir mağaza için sistemi çalıştıralım.

Programın kullanacağı genel mağaza bilgileri:

- Mağazada 20 kasiyer çalışıyor
- Mağaza saat 08:00'da açılıyor ve 22:00'da kapanıyor (14 saat açık kalıyor)

Programın kasiyerlere ilişkin girdi olarak alacağı bilgiler:

- Kasiyerler hangi günler çalışıyorlar?
- Kasiyerler pazartesi günü en erken saat kaçta vardiyaya başlayabiliyor?
- Kasiyerlerin ürün okutma performansı.
- Kasiyerlerin ödeme alma performansı.

Mağazanın genel işleyişine ilişkin girdi olarak kullanılacak bilgiler:

- Hangi gün hangi saatte mağazaya kaç müşteri gelecek?
- Hangi gün hangi saatte mağazaya gelen müşteriler kaç ürün alacak?
- Hangi gün hangi saatte alınacak ürünler hangi olasılıkla tartılacak?

9.1. Programda Kullanılan ve Rastgele Üretilmiş Veriler

Tablo 1: Kasiyerler Hangi Günler Çalışıyor?

	Pazartesi	Salı	Çarşamba	Perşembe	Cuma	Cumartesi	Pazar
Kasiyer 1	1	1	1	1	1	0	1
Kasiyer 2	1	1	0	1	1	1	1
Kasiyer 3	0	1	1	1	1	1	1
Kasiyer 4	1	0	1	1	1	1	1
Kasiyer 5	1	1	1	1	0	1	1
Kasiyer 6	1	1	1	1	0	1	1
Kasiyer 7	1	0	1	1	1	1	1
Kasiyer 8	1	0	1	1	1	1	1
Kasiyer 9	0	1	1	1	1	1	1
Kasiyer 10	1	1	0	1	1	1	1
Kasiyer 11	1	1	1	0	1	1	1
Kasiyer 12	0	1	1	1	1	1	1
Kasiyer 13	1	0	1	1	1	1	1
Kasiyer 14	1	0	1	1	1	1	1
Kasiyer 15	1	1	1	0	1	1	1
Kasiyer 16	1	1	1	1	0	1	1
Kasiyer 17	0	1	1	1	1	1	1
Kasiyer 18	0	1	1	1	1	1	1
Kasiyer 19	1	1	0	1	1	1	1
Kasiyer 20	0	1	1	1	1	1	1

Tablo 1'de 0 değerleri kasiyerlerin haftalık izinlerini 1 değerleri ise kasiyerlerin çalıştığı günleri göstermektedir.

Tablo 2: Kasiyerlerin Pazartesi Günü Gelebilecekleri En Erken Vardiya

	En Erken	Program için
Kasiyer 1	08:00	1
Kasiyer 2	08:00	1
Kasiyer 3	08:00	1
Kasiyer 4	08:00	1
Kasiyer 5	08:00	1
Kasiyer 6	08:00	1
Kasiyer 7	08:00	1
Kasiyer 8	08:00	1
Kasiyer 9	12:00	5
Kasiyer 10	12:00	5
Kasiyer 11	12:00	5
Kasiyer 12	12:00	5
Kasiyer 13	12:00	5
Kasiyer 14	12:00	5
Kasiyer 15	12:00	5
Kasiyer 16	12:00	5
Kasiyer 17	12:00	5
Kasiyer 18	12:00	5
Kasiyer 19	12:00	5
Kasiyer 20	12:00	5

Tablo 2'deki saatleri, programın anlayabilmesi için tam sayıya dönüştürülmesi gerekmektedir.

Tablo 3: Kasiyerlerin Ürün Okutma Performansları

	Okutma Süresi			
	Normal	**Tarayıcı**	**Barkod**	**PLU**
Kasiyer 1	1,000	2,000	4,000	2,000
Kasiyer 2	1,513	2,027	3,993	1,867
Kasiyer 3	1,540	1,680	3,927	2,560
Kasiyer 4	1,347	2,233	4,513	2,093
Kasiyer 5	1,067	1,680	4,513	2,467
Kasiyer 6	1,667	1,887	4,107	2,113
Kasiyer 7	1,180	2,093	4,320	2,220
Kasiyer 8	1,000	1,920	3,827	1,940
Kasiyer 9	1,040	2,367	4,113	2,653
Kasiyer 10	1,173	1,920	4,260	2,367

	Okutma Süresi			
	Normal	**Tarayıcı**	**Barkod**	**PLU**
Kasiyer 11	0,920	1,707	4,340	1,953
Kasiyer 12	0,973	2,300	3,680	2,000
Kasiyer 13	0,820	1,913	3,860	2,300
Kasiyer 14	0,867	2,420	3,953	2,487
Kasiyer 15	1,040	2,180	4,647	2,587
Kasiyer 16	1,513	2,067	3,787	1,807
Kasiyer 17	1,593	2,593	4,160	2,060
Kasiyer 18	1,360	2,367	4,420	1,833
Kasiyer 19	1,287	2,313	3,740	2,047
Kasiyer 20	1,527	2,487	3,680	2,160

Tablo 3'te her bir kasiyerin dört faklı ürün okutma şekli için saniye cinsinden performansları vardır. Ürün okutma şekilleri:

1. **Normal**: Kasa bandında bulunan normal okuyucu ile okutulabilir. (en hızlı yöntem)
2. **Tarayıcı**: Normal olarak okutulamayan ürünler tarayıcı adı verilen özel bir cihaz ile okutulabilir.
3. **Barkod**: Normal ya da Tarayıcı ile okutulamayan ürünlerin barkodları el ile sisteme girilebilir (en yavaş yöntemdir)
4. **PLU**: Eğer ürün tartılması gereken bir ürün ise PLU denilen kod girilip tartının üzerine yerleştirilir.

Tablo 4: Kasiyerin Ödeme Alma Performansları

	Ödeme Süresi	
	Nakit	**Kredi Kartı**
Kasiyer 1	10,000	12,000
Kasiyer 2	10,347	12,193
Kasiyer 3	10,307	11,740
Kasiyer 4	10,500	12,407
Kasiyer 5	9,867	12,060
Kasiyer 6	9,793	11,833
Kasiyer 7	10,373	12,487
Kasiyer 8	9,933	12,227
Kasiyer 9	9,960	11,893
Kasiyer 10	10,187	11,980
Kasiyer 11	9,833	11,893
Kasiyer 12	10,567	11,933
Kasiyer 13	10,280	12,087
Kasiyer 14	10,333	12,007
Kasiyer 15	10,667	12,567
Kasiyer 16	10,667	12,100
Kasiyer 17	9,700	12,007
Kasiyer 18	10,307	12,367
Kasiyer 19	10,207	11,673
Kasiyer 20	10,253	12,420

Kasiyerlerin saniye cinsinden ödeme alma performansları **Tablo 4**'te görüldüğü gibidir. Müşterilerin hangi ödeme yöntemini seçecekleri yine stokastik olarak belirlenmiştir. Nakit ödeme yapma olasılıkları %40 ve kredi kartı ile ödeme yapma olasılıkları %60 olarak belirlenmiştir.

Tablo 5: Pazartesi ve Salı Günü İçin Mağaza Performansı

Saat		Pazartesi			Salı		
Başlangıç	Bitiş	Müşteri	Ürün	PLU	Müşteri	Ürün	PLU
08:00	09:00	29	3	1%	29	3	1%
09:00	10:00	59	3	2%	59	3	2%
10:00	11:00	118	4	2%	118	4	2%
11:00	12:00	146	5	3%	147	5	3%
12:00	13:00	176	5	3%	176	5	3%
13:00	14:00	146	5	3%	147	5	3%
14:00	15:00	146	4	3%	147	4	3%
15:00	16:00	176	6	3%	176	6	3%
16:00	17:00	205	7	4%	205	7	4%
17:00	18:00	205	7	4%	205	7	4%
18:00	19:00	264	8	5%	264	8	5%
19:00	20:00	352	9	5%	352	9	5%
20:00	21:00	498	10	5%	499	10	5%
21:00	22:00	264	11	6%	264	11	6%

Tablo 6: Çarşamba ve Perşembe Günü İçin Mağaza Performansı

Saat		Çarşamba			Perşembe		
Başlangıç	Bitiş	Müşteri	Ürün	PLU	Müşteri	Ürün	PLU
08:00	09:00	29	3	1%	29	3	1%
09:00	10:00	59	3	2%	59	3	2%
10:00	11:00	118	4	2%	117	4	2%
11:00	12:00	147	5	3%	145	5	3%
12:00	13:00	176	5	3%	175	5	3%
13:00	14:00	147	5	3%	145	5	3%
14:00	15:00	147	4	3%	145	4	3%
15:00	16:00	176	6	3%	175	6	3%
16:00	17:00	205	7	4%	203	7	4%
17:00	18:00	205	7	4%	203	7	4%
18:00	19:00	264	8	5%	261	8	5%
19:00	20:00	352	9	5%	348	9	5%
20:00	21:00	499	10	5%	493	10	5%
21:00	22:00	264	11	6%	261	11	6%

Tablo 7: Cuma ve Cumartesi Günü İçin Mağaza Performansı

Saat		Cuma			Cumartesi		
Başlangıç	Bitiş	Müşteri	Ürün	PLU	Müşteri	Ürün	PLU
08:00	09:00	32	3	1%	40	3	1%
09:00	10:00	63	3	2%	80	3	2%
10:00	11:00	125	4	2%	159	4	2%
11:00	12:00	157	5	3%	199	5	3%
12:00	13:00	187	5	3%	239	5	3%
13:00	14:00	157	5	3%	199	5	3%
14:00	15:00	157	4	3%	199	4	3%
15:00	16:00	187	6	3%	239	6	3%
16:00	17:00	219	7	4%	279	7	4%
17:00	18:00	219	7	4%	279	7	4%
18:00	19:00	281	8	5%	358	8	5%
19:00	20:00	375	9	5%	477	9	5%
20:00	21:00	531	10	5%	676	10	5%
21:00	22:00	281	11	6%	358	11	6%

Tablo 8: Pazar Günü İçin Mağaza Performansı

Saat		Pazar		
Başlangıç	Bitiş	Müşteri	Ürün	PLU
08:00	09:00	42	3	1%
09:00	10:00	83	3	2%
10:00	11:00	166	4	2%
11:00	12:00	207	5	3%
12:00	13:00	248	5	3%
13:00	14:00	207	5	3%
14:00	15:00	207	4	3%
15:00	16:00	248	6	3%
16:00	17:00	291	7	4%
17:00	18:00	291	7	4%
18:00	19:00	373	8	5%
19:00	20:00	497	9	5%
20:00	21:00	704	10	5%
21:00	22:00	373	11	6%

Tablo 5, Tablo 6, Tablo 7 ve **Tablo 8**'de Müşteri sütunu, belirtilen gün ve saatte mağazaya ortalama kaç müşteri girdiği; ürün sütununda, giren bir müşterinin ortalama kaç ürün alacağı ve PLU sütunu, alınan ürünler hangi olasılık ile tartma işlemine tabi tutulacağıdır.

Yukarıda belirtilen değerleri girdi olarak aldıktan sonra TA ile üretilen 50 vardiya ve her vardiya için 10 MC tekrarı ile program çalıştırıldıktan sonra aşağıdaki çıktıyı üretmektedir.

Tablo 9: Programın Üretmiş Olduğu Vardiya Çıktısı

	Pazartesi	Salı	Çarşamba	Perşembe	Cuma	Cumartesi	Pazar
1. Kasiyer	6	6	5	3	5	-1	2
2. Kasiyer	5	6	-1	5	5	5	0
3. Kasiyer	-1	4	6	4	5	5	2
4. Kasiyer	4	-1	5	5	6	6	5
5. Kasiyer	5	6	0	1	-1	0	2
6. Kasiyer	0	2	4	5	-1	0	0
7. Kasiyer	4	-1	6	6	4	4	2
8. Kasiyer	3	-1	0	0	6	6	6
9. Kasiyer	-1	5	6	5	6	6	1
10. Kasiyer	4	2	-1	1	6	5	5
11. Kasiyer	4	0	5	-1	0	6	5
12. Kasiyer	-1	5	0	5	5	6	3
13. Kasiyer	4	-1	5	5	2	5	6
14. Kasiyer	6	-1	6	5	5	6	6
15. Kasiyer	4	6	5	-1	4	0	6
16. Kasiyer	6	6	1	0	-1	2	3
17. Kasiyer	-1	3	5	6	0	2	6
18. Kasiyer	-1	5	2	0	5	5	4
19. Kasiyer	4	3	-1	4	0	3	0
20. Kasiyer	-1	0	5	4	6	4	6

Tablo 9'da -1 değerleri haftalık izinleri ve diğer değerler ise mağazanın çalışmaya başlayacağı kaçıncı saatte vardiyanın başlayacağını göstermektedir. Bu tablo normal saatlerden oluşan bir sonuca dönüştürüldüğünde **Tablo 10** gibi görünmektedir.

Tablo 10: Düzenlenmiş Vardiya Çıktısı

	Pazartesi	Salı	Çarşamba	Perşembe	Cuma	Cumartesi	Pazar
1. Kasiyer	14:00-22:00	14:00-22:00	13:00-21:00	11:00-19:00	13:00-21:00	H.İ.	10:00-18:00
2. Kasiyer	13:00-21:00	14:00-22:00	H.İ.	13:00-21:00	13:00-21:00	13:00-21:00	08:00-16:00
3. Kasiyer	H.İ.	12:00-20:00	14:00-22:00	12:00-20:00	13:00-21:00	13:00-21:00	10:00-18:00
4. Kasiyer	12:00-20:00	H.İ.	13:00-21:00	13:00-21:00	14:00-22:00	14:00-22:00	13:00-21:00
5. Kasiyer	13:00-21:00	14:00-22:00	08:00-16:00	09:00-17:00	H.İ.	08:00-16:00	10:00-18:00
6. Kasiyer	08:00-16:00	10:00-18:00	12:00-20:00	13:00-21:00	H.İ.	08:00-16:00	08:00-16:00
7. Kasiyer	12:00-20:00	H.İ.	14:00-22:00	14:00-22:00	12:00-20:00	12:00-20:00	10:00-18:00
8. Kasiyer	11:00-19:00	H.İ.	08:00-16:00	08:00-16:00	14:00-22:00	14:00-22:00	14:00-22:00
9. Kasiyer	H.İ.	13:00-21:00	14:00-22:00	13:00-21:00	14:00-22:00	14:00-22:00	09:00-17:00
10. Kasiyer	12:00-20:00	10:00-18:00	H.İ.	09:00-17:00	14:00-22:00	13:00-21:00	13:00-21:00
11. Kasiyer	12:00-20:00	08:00-16:00	13:00-21:00	H.İ.	08:00-16:00	14:00-22:00	13:00-21:00
12. Kasiyer	H.İ.	13:00-21:00	08:00-16:00	13:00-21:00	13:00-21:00	14:00-22:00	11:00-19:00
13. Kasiyer	12:00-20:00	H.İ.	13:00-21:00	13:00-21:00	10:00-18:00	13:00-21:00	14:00-22:00
14. Kasiyer	14:00-22:00	H.İ.	14:00-22:00	13:00-21:00	13:00-21:00	14:00-22:00	14:00-22:00
15. Kasiyer	12:00-20:00	14:00-22:00	13:00-21:00	H.İ.	12:00-20:00	08:00-16:00	14:00-22:00
16. Kasiyer	14:00-22:00	14:00-22:00	09:00-17:00	08:00-16:00	H.İ.	10:00-18:00	11:00-19:00
17. Kasiyer	H.İ.	11:00-19:00	13:00-21:00	14:00-22:00	08:00-16:00	10:00-18:00	14:00-22:00
18. Kasiyer	H.İ.	13:00-21:00	10:00-18:00	08:00-16:00	13:00-21:00	13:00-21:00	12:00-20:00
19. Kasiyer	12:00-20:00	11:00-19:00	H.İ.	12:00-20:00	08:00-16:00	11:00-19:00	08:00-16:00
20. Kasiyer	H.İ.	08:00-16:00	13:00-21:00	12:00-20:00	14:00-22:00	12:00-20:00	14:00-22:00

Programın 50 deneme ve her deneme için MC benzetim tekniğinde 10 tekrar sonrasında bulduğu en iyi vardiya ile kuyruk sonuçları aşağıdaki gibidir:

Tablo 11: Üretilen En İyi Vardiya ile Ulaşılacak Kuyruk Uzunluğu Tahmini

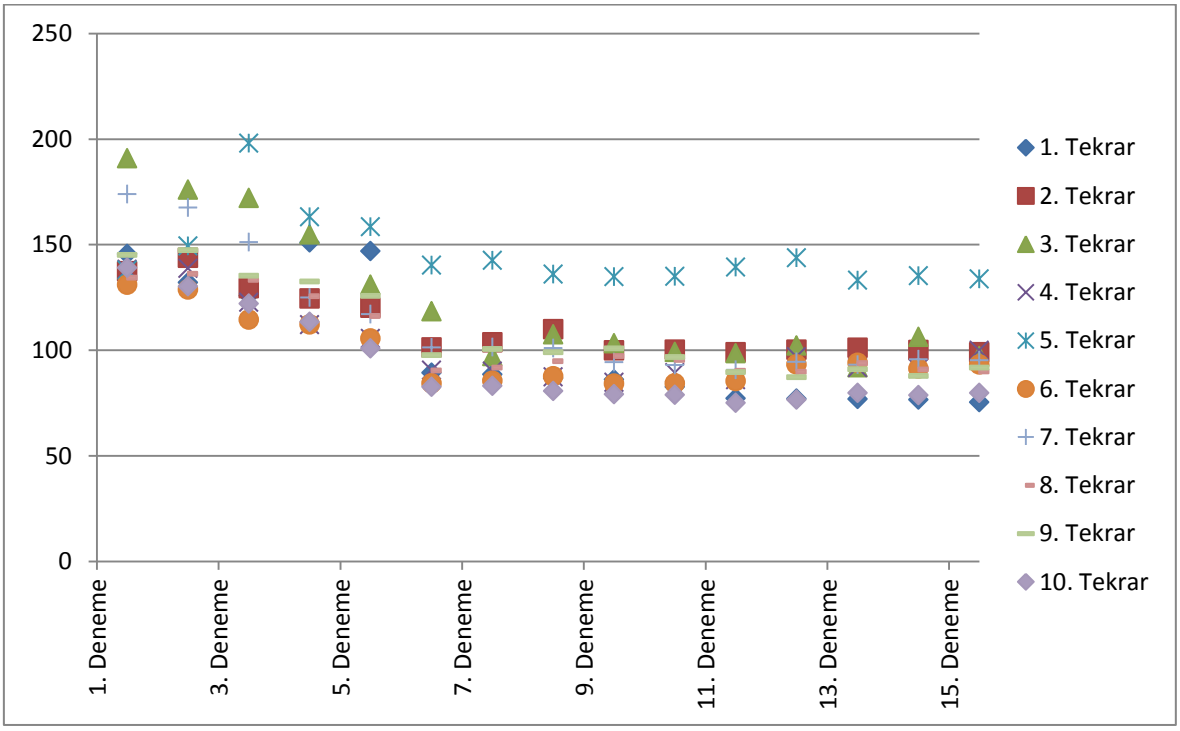
Saat Aralığı		Pazartesi	Salı	Çarşamba	Perşembe	Cuma	Cumartesi	Pazar
Başlangıç	Bitiş							
08:00	**09:00**	0,03	0,02	0,03	0,00	0,01	0,00	0,00
09:00	**10:00**	0,04	0,28	0,40	0,17	0,33	0,25	0,25
10:00	**11:00**	0,05	1,05	0,81	0,44	1,12	0,66	0,93
11:00	**12:00**	0,03	1,54	0,45	1,25	0,68	0,78	3,52
12:00	**13:00**	0,08	1,07	0,55	0,71	0,81	0,67	0,68
13:00	**14:00**	0,72	0,56	0,44	0,38	0,42	0,34	0,29
14:00	**15:00**	0,21	0,42	0,19	0,28	0,26	0,22	0,21
15:00	**16:00**	0,16	0,21	0,18	0,14	0,17	0,23	0,17
16:00	**17:00**	0,25	0,29	0,22	0,25	0,27	0,31	0,24
17:00	**18:00**	0,26	0,32	0,22	0,22	0,24	0,39	0,32
18:00	**19:00**	0,52	0,50	0,48	0,43	0,51	0,60	0,57
19:00	**20:00**	0,82	1,36	1,06	0,83	1,22	1,85	1,03
20:00	**21:00**	1,92	3,43	2,61	2,24	2,52	3,16	3,53
21:00	**22:00**	1,82	2,89	1,54	1,27	1,34	2,97	**3,65**

Tablo 11'de görüldüğü üzere Pazartesi günü saat 08:00 ile 09:00 arasında kasaya gelen müşteriler ortalama 0,03 kişi kuyruk bekleyecekleri tahmin edilmektedir. Aynı şekilde Pazar günü saat 21:00 ile 22:00 arasında kasaya gelen müşteriler ortalama 3,65 kişi kuyruk bekleyecekleri tahmin edilmektedir.

Programın ürettiği kuyruk sonuçlarını incelediğimizde, toplam 98 gün saat kombinasyonundan, 10 tanesinin kritik değer olarak belirlenen 2 kişilik kuyruk uzunluğundan daha yüksek olduğu görülmektedir. Bu kuyruk uzunluklarının kritik değerleri aştığı noktaların nedeninin, programın içindeki bir mantık hatası mı yoksa deneme sayısındaki yetersizlik mi olduğunu tespit etmek gerekmektedir. Bu tespitin yapılabilmesi için donanımsal yetersizlik nedeni ile denenecek vardiya sayısının arttırılması mümkün olmadığından; program 15 farklı vardiya deneyecek ve MC benzetim tekniğinde 10 tekrar yapacak şekilde 10'ar defa çalıştırılıp kuyruk uzunluklarındaki değerleri üretilmiştir. Her vardiya da üretilmiş kuyruk değerlerinin; her saat ve her güne ilişkin değerleri toplanarak, tek bir performans kriteri oluşturulmuştur. Bulunan bu performans kriteri ile deneme sayıları, aralarında ilişki olup olmadığının anlaşılması için SPSS 15.0 for Windows Evaluation Version programı ile incelenmiştir.

Tablo 12: TA Algoritası ile Üretilen 15 Vardiyanın 10 Kere Tekrarlanması İle Elde Edilen Kuyruk Sonuçları

Peş Peşe Üretilen Vardiya Sayısı	Program Ayarları Sıfırlanarak Yapılan Tekrar Sayısı									
	1. Tekrar	2. Tekrar	3. Tekrar	4. Tekrar	5. Tekrar	6. Tekrar	7. Tekrar	8. Tekrar	9. Tekrar	10. Tekrar
1. Deneme	146	137	191	138	138	131	174	134	145	139
2. Deneme	132	144	176	139	149	129	168	136	147	130
3. Deneme	126	129	172	123	198	114	151	133	135	122
4. Deneme	151	125	155	112	163	112	125	126	132	113
5. Deneme	147	120	131	106	158	105	117	116	126	101
6. Deneme	89	101	118	91	140	84	101	90	98	83
7. Deneme	89	104	97	90	143	86	101	92	101	83
8. Deneme	87	110	107	87	136	88	101	95	99	81
9. Deneme	86	100	103	85	135	84	94	97	101	79
10. Deneme	83	100	99	89	135	84	93	95	97	79
11. Deneme	77	99	99	86	139	85	91	90	90	75
12. Deneme	77	100	102	97	144	93	95	90	87	77
13. Deneme	77	101	92	92	133	94	93	94	91	80
14. Deneme	77	100	106	91	135	91	96	91	88	79
15. Deneme	75	99	97	100	134	93	95	90	92	80



Şekil 5: Deneme Sayısı ve Yapılan Tekrarlara Göre Kuyruk Uzunluklarının Dağılımı

Şekil 5'teki grafikte deneme, TA algoritmasının ürettiği vardiya sayısı; tekrar ise TA algoritmasının kaç kere çalıştırıldığını göstermektedir. **Tablo 12** ve **Şekil 5** incelendiğinde deneme sayısı arttıkça kuyruk uzunluğunun azaldığı görülmektedir. Deneme ile kuyruk uzunluğu arasındaki ilişkinin tespit edilebilmesi için bu iki değişken arasında (deneme bağımsız ve kuyruk bağımlı değişken olacak şekilde) regresyon analizi yapıldığında:

Tablo 13: Model Özet Tablosu

Model Özeti

Model	R	R Kare	Düzeltirmiş R Kare	Tahmin Std. Hatası
1	0,663	0,439	0,436	19,824

Kestirici: (Sabit), deneme

Tablo 14: Anova Tablosu

ANOVA(b)

Model	Kareler Toplamı	Sd	Kareler Ortalaması	F	Sig.
Regresyon	45579,61	1	45579,61	115,99	0,00
Artık	58160,54	148	392,98		
Toplam	103740,15	149			

Kestirici: (Sabit), deneme

Bağımlı Değişken: kuyruk

Tablo 15: Katsayılar Tablosu

Katsayılar(a)

	Standartlaştırılmamış Katsayılar		Standartlaştırılmış Katsayılar		
Model	B	Std. Hata	Beta	t	Sig.
(Sabit)	142,337	3,406		41,788	0,000
deneme	-4,035	0,375	-0,663	-10,770	0,000

Bağımlı Değişken: kuyruk

Regresyon sonuçları incelendiğinde, kuyruk uzunluğu değerindeki değişimlerin %43,9'unun deneme sayısı ile açıklanabildiği görülmektedir. Kurulan regresyon modeli ve katsayıların da istatistiksel olarak anlamlı olduğu anlaşılmaktadır. Yani programda kurulan modelin ve TA algoritmasının doğru olduğu ve deneme sayısı arttıkça TA algoritmasının daha başarılı sonuçlar üretebileceği görülmektedir.

10.SONUÇ

Özellikle perakende sektöründeki karışık kuyruk yapısını detaylı bir şekilde modelleyip, bu kuyruk problemini çözecek bir araştırmanın daha önceden yapılmamış olması nedeni ile yapılan bu çalışmada: MC benzetim tekniği ve TA algoritması bir arada kullanılarak, kuyruk problemi çözülmeye çalışılmıştır.

TA algoritması esnek yapısı sayesinde, bir marketteki kuyruk yapısı kadar karışık bir sistem için çözüm üretebilmiş ve MC benzetim tekniği sayesinde bu karışık yapı, modellenerek performans kriterleri hesaplanabilmiştir. Bu kadar detaylandırılmış şekilde iki tekniğin beraber kullanımını güçleştiren en büyük etken, iki tekniğin de çözüme ulaşma sürelerinin çok uzun olmasıdır. Deneme sayısına ve benzetim içindeki tekrar sayısına bağlı olarak çözüm süresi doğrudan etkilenmektedir. MC benzetim modeli ve TA algoritması üzerinde yapılan optimizasyonlar sayesinde, bu iki tekniğin bir arada kullanılması mümkün hale gelmiştir.

TA ve MC benzetim modeli kurulduktan sonra, bu ikili sistemin çalışma performansları, rastgele üretilmiş mağaza ve kasiyer performansları üzerinde çalıştırılmıştır. Bulunan sonuçlar değerlendirildiğinde, kurulan sistemin başarılı bir şekilde çalıştığı, ancak sonuç performansının, deneme sayısından çok büyük ölçüde etkilendiği tespit edilmiştir.

Kurulan bu sistemin iki ana zayıf noktası vardır. Birincisi, deneme sayısının performansı çok büyük ölçüde etkilemesidir. Düşük performanslı bir bilgisayarda bu sistem çalıştırıldığında, ya çok uzun süre beklenmesi gerekecek ya da çözüm kalitesinden ödün verilmesi gerekecektir. İkinci zayıf nokta ise, mağaza performans parametreleridir. MC benzetim tekniği bu parametrelere göre rastgele sayılar üreterek, vardiya üretilmek istenen haftayı modelleyecektir. Bu parametrelerin doğru bir şekilde girilmemesi; vardiya üretilmek istenen haftaya ilişkin müşteri ve ürün sayılarının yanlış tahmin edilmesine sebep olacaktır. Bu hatalar ise MC benzetim tekniğinin gerçek dünyadaki değerleri yansıtamamasına dolayısıyla da vardiyaların yanlış tahmin edilmesine sebep olacaktır.

Günümüz teknolojisinde kullanılan veri ambarları sayesinde; mağaza ve kasiyer performansları gün, saat hatta müşteri bazında uzun dönemler boyunca kayıt altında tutulabilmektedir. Ayrıca güncel server sistemleri ile programın yeterli sayıda çalıştırılıp makul bir sürede sonuca ulaşılabilecektir. Bu sayede doğru verilerin kullanılması ve yeterli sayıda iterasyonun yapılması sayesinde sistem, optimal çözüme ya da optimale en yakın çözüme hatasız bir şekilde ulaşabilecektir.

11.PROGRAM KODLARI

11.1. TA Algoritması için Değişkenlerin Tanımlanması ve Okutulması

```
Public Sonuc() As Integer
Dim toplam As Double
Dim Kisa_Sureli_Hafiza() As Integer
Dim Uzun_Sureli_Hafiza() As Integer
Dim Kisa_Sureli_Kuyruk() As Double
Dim Uzun_Sureli_Kuyruk() As Double
Dim Kisa_Sureli_Kuyruk_Var() As Double
Dim Uzun_Sureli_Kuyruk_Var() As Double
Dim Kuyruk_Uzunlugu() As Double
Dim Kasiyer_Sayisi As Integer
Dim En_Erken() As Integer
Dim Kac_Gun_Calisiyor() As Integer
Dim Kac_Gun_Calisti() As Integer
Dim Hangi_Gunler_Calisiyor() As Integer
Dim Hangi_Saatler_Calisiyor() As Integer
Dim Magaza_Kapanisi As Integer
Dim Magaza_Calisma_Saatleri() As Integer
Dim Kac_Kasiyer_Calisiyor() As Integer
Dim Saat_Dilimi As Integer
Dim Kasiyer_Vardiyasi() As Integer
Dim Olasi_Saatler As Integer
Dim Secili_Kasiyer_Vardiyasi() As Integer
Dim Vardiya_Degisikligi() As Integer
Dim Deneme_Sayisi, Simulator_Sayisi As Integer

Sub Verileri_Oku()
Dim i, j, k As Integer
Dim n As Integer
n = Sheets("Genel_Bilgiler").Cells(10, 2)
Saat_Dilimi = Sheets("genel_bilgiler").Cells(5, 2)
Kasiyer_Sayisi = Sheets("Genel_Bilgiler").Cells(6, 2)
Magaza_Kapanisi = Sheets("Genel_Bilgiler").Cells(7, 2)
ReDim En_Erken(7, Kasiyer_Sayisi) As Integer
ReDim Kac_Gun_Calisiyor(Kasiyer_Sayisi) As Integer
ReDim Kac_Gun_Calisti(Kasiyer_Sayisi) As Integer
ReDim Hangi_Gunler_Calisiyor(7, Kasiyer_Sayisi) As Integer
ReDim Hangi_Saatler_Calisiyor(7, Kasiyer_Sayisi, Magaza_Kapanisi) As Integer
ReDim Magaza_Calisma_Saatleri(7, Magaza_Kapanisi) As Integer
ReDim Kac_Kasiyer_Calisiyor(7, Magaza_Kapanisi) As Integer
ReDim Secili_Kasiyer_Vardiyasi(7, Kasiyer_Sayisi) As Integer
ReDim Vardiya_Degisikligi(7, Kasiyer_Sayisi) As Integer
Olasi_Saatler = Magaza_Kapanisi - 8 * Saat_Dilimi
ReDim Kasiyer_Vardiyasi(7, Kasiyer_Sayisi, Magaza_Kapanisi) As Integer
ReDim Kisa_Sureli_Hafiza(n, 7, Kasiyer_Sayisi) As Integer
ReDim Uzun_Sureli_Hafiza(10, 7, Kasiyer_Sayisi) As Integer
ReDim Kisa_Sureli_Kuyruk(n, 7, Magaza_Kapanisi) As Double
ReDim Uzun_Sureli_Kuyruk(10, 7, Magaza_Kapanisi) As Double
ReDim Kisa_Sureli_Kuyruk_Var(n, 7, Magaza_Kapanisi) As Double
ReDim Uzun_Sureli_Kuyruk_Var(10, 7, Magaza_Kapanisi) As Double
Deneme_Sayisi = Sheets("Genel_Bilgiler").Cells(8, 2)
Simulator_Sayisi = Sheets("Genel_Bilgiler").Cells(9, 2)
For j = 1 To Kasiyer_Sayisi
   En_Erken(1, j) = Sheets("Kasiyer_Perf").Cells(j + 2, 19)
   Kac_Gun_Calisiyor(j) = Sheets("Kasiyer_Perf").Cells(j + 2, 17)
   For i = 1 To 7
```

```
      Hangi_Gunler_Calisiyor(i, j) = Sheets("Kasiyer_Perf").Cells(j + 2, i + 9)
   Next i
Next j
End Sub
```

11.2. İlk Vardiyanın Oluşturulması

```
Sub Vardiya_Olustur()
Dim i, j, k, n As Integer
Dim Is_Basi As Integer
Dim Rastgele As Integer
Dim Rastgele2 As Double
n = 1
Randomize
For j = 1 To Kasiyer_Sayisi
   For i = 1 To 7
      'i. gün için j. kasiyere vardiya ata
      If Hangi_Gunler_Calisiyor(i, j) = 0 Then
         Secili_Kasiyer_Vardiyasi(i, j) = -1
      Else
         Secili_Kasiyer_Vardiyasi(i, j) = Rastgele_Vardiya_Bul(i, j)
      End If
      'i+1'inci gün için j. kasiyerin en_erken değerini bul
      If i < 7 Then
         If Secili_Kasiyer_Vardiyasi(i, j) + (8 + 11) * Saat_Dilimi <= 24 * Saat_Dilimi Then
            En_Erken(i + 1, j) = 0
         Else
            En_Erken(i + 1, j) = Secili_Kasiyer_Vardiyasi(i, j) + (8 + 11) * Saat_Dilimi - (24 *
Saat_Dilimi)
         End If
      End If
      For k = 0 To Magaza_Kapanisi - 1
         If Secili_Kasiyer_Vardiyasi(i, j) <> -1 Then
            If k >= Secili_Kasiyer_Vardiyasi(i, j) And k <= Secili_Kasiyer_Vardiyasi(i, j) + 8 *
Saat_Dilimi Then
               Kasiyer_Vardiyasi(i, j, k) = 1
            Else
               Kasiyer_Vardiyasi(i, j, k) = 0
            End If
         Else
            Kasiyer_Vardiyasi(i, j, k) = 0
         End If
      Next k
   Next i
Next j
For i = 1 To 7
   For k = 0 To Magaza_Kapanisi
      For j = 1 To Kasiyer_Sayisi
         Kac_Kasiyer_Calisiyor(i, k) = Kac_Kasiyer_Calisiyor(i, k) + Kasiyer_Vardiyasi(i, j, k)
      Next j
   Next k
Next i
End Sub
```

11.3. Tabu Arama Algoritması

```
Sub Tabu_Arama()
Dim i, j, k, n As Integer
n = Sheets("Genel_Bilgiler").Cells(10, 2)
DoEvents
Verileri_Oku
Vardiya_Olustur
Kasiyer_Vardiyasi_Kontrol
Secili_Kasiyer_Vardiyasini_Hafizaya_Al 1, 1
Secili_Kasiyer_Vardiyasini_Hafizaya_Al 2, 1
Benzetim Kasiyer_Sayisi, Kasiyer_Vardiyasi, Simulator_Sayisi, Kac_Kasiyer_Calisiyor
Kuyruk_Uzunlugunu_Hafizaya_Al 1, 1
Kuyruk_Uzunlugunu_Hafizaya_Al 2, 1
i = 2
j = 2
k = 1
While k <= Deneme_Sayisi
   If k > 1 Then
      i = 1
      Degisiklik 0.4
      Secili_Kasiyer_Vardiyasini_Hafizaya_Al 1, i
      Benzetim Kasiyer_Sayisi, Kasiyer_Vardiyasi, Simulator_Sayisi, Kac_Kasiyer_Calisiyor
      If Kuyruk_Kiyasla Then
         Secili_Kasiyer_Vardiyasini_Hafizaya_Al 2, 1
         Kuyruk_Uzunlugunu_Hafizaya_Al 2, 1
      End If
      i = i + 1
   End If
   While i <= n
      DoEvents
      Degisiklik 0.1
      Secili_Kasiyer_Vardiyasini_Hafizaya_Al 1, i
      DoEvents
      Benzetim Kasiyer_Sayisi, Kasiyer_Vardiyasi, Simulator_Sayisi, Kac_Kasiyer_Calisiyor
      If Kuyruk_Kiyasla Then
         Secili_Kasiyer_Vardiyasini_Hafizaya_Al 2, 1
         Kuyruk_Uzunlugunu_Hafizaya_Al 2, 1
      End If
      i = i + 1
   Wend
   k = k + 1
Wend
Sonuc_Yaz:
DoEvents
ReDim En_Iyi_Vardiya(7, Kasiyer_Sayisi)
ReDim En_Kisa_Kuyruk(7, Magaza_Kapanisi)
For i = 1 To 7
   For j = 1 To Kasiyer_Sayisi
      En_Iyi_Vardiya(i, j) = Uzun_Sureli_Hafiza(1, i, j)
   Next
   For k = 0 To Magaza_Kapanisi - 1
      En_Kisa_Kuyruk(i, k + 1) = Uzun_Sureli_Kuyruk(1, i, k)
   Next k
Next
Sheets("Kuyruk").Range("B2:V9") = En_Iyi_Vardiya
Sheets("Kuyruk").Range("B14:P21") = En_Kisa_Kuyruk
End Sub
```

11.4. Üretilen Vardiyalar Kısıtlara Uygun mu Kontrolü

```
Sub Kasiyer_Vardiyasi_Kontrol()
Dim i, j, k As Integer
Dim Basla, Bitis As Boolean
For i = 1 To 7
   Basla = True
   Bitis = True
   For j = 1 To Kasiyer_Sayisi
      If Secili_Kasiyer_Vardiyasi(i, j) = 0 Then Basla = False
      If Secili_Kasiyer_Vardiyasi(i, j) = 6 Then Bitis = False
   Next
   If Basla Then
      For j = 1 To Kasiyer_Sayisi
         If Secili_Kasiyer_Vardiyasi(i, j) = 1 Then
            Secili_Kasiyer_Vardiyasi(i, j) = 0
            Kac_Kasiyer_Calisiyor(i, 1) = Kac_Kasiyer_Calisiyor(i, 1) + 1
            Kac_Kasiyer_Calisiyor(i, 9) = Kac_Kasiyer_Calisiyor(i, 9) - 1
            Kasiyer_Vardiyasi(i, j, 0) = 1
            Kasiyer_Vardiyasi(i, j, 8) = 0
         End If
      Next
   End If

   If Bitis Then
      For j = 1 To Kasiyer_Sayisi
         If Secili_Kasiyer_Vardiyasi(i, j) = 5 Then
            Secili_Kasiyer_Vardiyasi(i, j) = 6
            Kac_Kasiyer_Calisiyor(i, 6) = Kac_Kasiyer_Calisiyor(i, 6) - 1
            Kac_Kasiyer_Calisiyor(i, 14) = Kac_Kasiyer_Calisiyor(i, 14) + 1
            Kasiyer_Vardiyasi(i, j, 13) = 1
            Kasiyer_Vardiyasi(i, j, 5) = 0
         End If
      Next
   End If
Next
End Sub
```

11.5. Kasiyer Vardiyasını Hafızaya Al

```
Sub Secili_Kasiyer_Vardiyasini_Hafizaya_Al(Hafiza, Indeks)
Dim i, j As Integer
For i = 1 To 7
   For j = 1 To Kasiyer_Sayisi
      If Hafiza = 1 Then
         Kisa_Sureli_Hafiza(Indeks, i, j) = Secili_Kasiyer_Vardiyasi(i, j)
      ElseIf Hafiza = 2 Then
         Uzun_Sureli_Hafiza(Indeks, i, j) = Secili_Kasiyer_Vardiyasi(i, j)
      End If
   Next j
Next i
End Sub
```

11.6. Vardiyayı Değiştirmek

```
Private Sub Degisiklik(p As Double)
Dim i, j, k As Integer
Dim Rastgele As Double
```

```
Dim p1, p2 As Double
Dim Degistimi, Tabu_Mu As Boolean
Degisiklik_Basi:
Tabu_Mu = True
For i = 1 To 7
   For j = 1 To Kasiyer_Sayisi
      Randomize
      Rastgele = Rnd()
      Degistimi = False
      If Secili_Kasiyer_Vardiyasi(i, j) <> -1 Then
         If Secili_Kasiyer_Vardiyasi(i, j) > 0 And Secili_Kasiyer_Vardiyasi(i, j) < 6 * Saat_Dilimi Then
            p1 = Module2.Ort_Kuyruk(i, Secili_Kasiyer_Vardiyasi(i, j)) / (Module2.Ort_Kuyruk(i,
Secili_Kasiyer_Vardiyasi(i, j)) + Module2.Ort_Kuyruk(i, Secili_Kasiyer_Vardiyasi(i, j) + 9))
            p2 = Module2.Ort_Kuyruk(i, Secili_Kasiyer_Vardiyasi(i, j) + 9) / (Module2.Ort_Kuyruk(i,
Secili_Kasiyer_Vardiyasi(i, j)) + Module2.Ort_Kuyruk(i, Secili_Kasiyer_Vardiyasi(i, j) + 9))
         ElseIf Secili_Kasiyer_Vardiyasi(i, j) = 6 * Saat_Dilimi Then
            If Module2.Ort_Kuyruk(i, Secili_Kasiyer_Vardiyasi(i, j) + 8) > Module2.Ort_Kuyruk(i,
Secili_Kasiyer_Vardiyasi(i, j) + 1) Then
               p1 = 1
            Else
               p1 = 0
            End If
         ElseIf Secili_Kasiyer_Vardiyasi(i, j) = 0 Then
            If Module2.Ort_Kuyruk(i, Secili_Kasiyer_Vardiyasi(i, j) + 1) < Module2.Ort_Kuyruk(i,
Secili_Kasiyer_Vardiyasi(i, j) + 9) Then
               p2 = 1
            Else
               p2 = 0
            End If
         End If
         If Secili_Kasiyer_Vardiyasi(i, j) < (Magaza_Kapanisi - 8) * Saat_Dilimi And
Secili_Kasiyer_Vardiyasi(i, j) > En_Erken(i, j) Then
            If Rastgele <= p * p1 * 2 Then
               Vardiya_Degisikligi(i, j) = -1
               Degistimi = True
               Tabu_Mu = False
            ElseIf Rastgele <= 2 * p And Rastgele > p * p1 * 2 Then
               Vardiya_Degisikligi(i, j) = 1
               Degistimi = True
               Tabu_Mu = False
            End If
         ElseIf Secili_Kasiyer_Vardiyasi(i, j) = (Magaza_Kapanisi - 8) * Saat_Dilimi And
Kac_Kasiyer_Calisiyor(i, Magaza_Kapanisi) > 1 Then
            If Rastgele <= 2 * p * p2 Then
               Vardiya_Degisikligi(i, j) = -1
               Degistimi = True
               Tabu_Mu = False
            End If
         ElseIf Secili_Kasiyer_Vardiyasi(i, j) = En_Erken(i, j) Then
            If Kac_Kasiyer_Calisiyor(i, 1) > 1 Then
               If Rastgele <= 2 * p * p1 Then
                  Vardiya_Degisikligi(i, j) = 1
                  Degistimi = True
                  Tabu_Mu = False
               End If
            End If
         End If
         If Not Degistimi Then
            Vardiya_Degisikligi(i, j) = 0
         End If
```

```
      End If
   Next j
Next i
If Tabu_Mu Then GoTo Degisiklik_Basi
For i = 1 To 7
   For j = 1 To Kasiyer_Sayisi
      If Secili_Kasiyer_Vardiyasi(i, j) <> -1 Then
         Secili_Kasiyer_Vardiyasi(i, j) = Secili_Kasiyer_Vardiyasi(i, j) + Vardiya_Degisikligi(i, j)
         Kasiyer_Vardiyasi(i, j, Secili_Kasiyer_Vardiyasi(i, j) - Vardiya_Degisikligi(i, j)) = 0
         Kasiyer_Vardiyasi(i, j, Secili_Kasiyer_Vardiyasi(i, j) + 8 * Saat_Dilimi) = 1
         Kac_Kasiyer_Calisiyor(i, Secili_Kasiyer_Vardiyasi(i, j) - Vardiya_Degisikligi(i, j) + 1) =
Kac_Kasiyer_Calisiyor(i, Secili_Kasiyer_Vardiyasi(i, j) - Vardiya_Degisikligi(i, j) + 1) - 1
         Kac_Kasiyer_Calisiyor(i, Secili_Kasiyer_Vardiyasi(i, j) + 8 * Saat_Dilimi) =
Kac_Kasiyer_Calisiyor(i, Secili_Kasiyer_Vardiyasi(i, j) + 8 * Saat_Dilimi) + 1
      End If
   Next j
Next i
End Sub
```

11.7. Poisson Dağılımından Rastgele Sayı Üretmek

```
Public Function Poisson_Rastgele(lamda)
Dim k, L, p
p = 1
k = 0
L = Exp(-lamda)
Do While p > L
   k = k + 1
   p = p * Rnd()
Loop
Poisson_Rastgele = k - 1
End Function
```

11.8. Üstel Dağılımından Rastgele Sayı Üretmek

```
Public Function Ustel_Rastgele(lamda)
Dim u As Double
Baslangic:
u = Rnd()
If u = 0 Then GoTo Baslangic
u = Log(u)
Ustel_Rastgele = (-lamda) * u
End Function
```

11.9. MC Benzetim Tekniği Kodları

```
Public Ort_Kuyruk(), Var_Kuyruk() As Double
Dim Saatlik_Musteri() As Integer

Public Function Benzetim(Kasiyer_Sayisi, Kasiyer_Vardiyasi, Tekrar_Sayisi,
Kac_Kasiyer_Calisiyor)
Dim Okutma As Integer
Dim i, j, k, n, m As Integer
Dim Sure, Saat As Double
Dim Magaza_Kapanisi, Kasiyer_Sec As Integer
Dim Saat_Dilimi As Integer
Dim Musteriler(10000, 8, 7)
```

```
Dim Musteriler2(10000, 8)
Dim lamda As Double
Dim Musteri_Sayisi(7) As Integer
Dim Urun_Sayisi As Integer
Saat_Dilimi = Sheets("genel_bilgiler").Cells(7, 2) + 1
ReDim Musteri_Sayisi_Ort(Saat_Dilimi)
Dim Gecerli_Saat As Integer
Dim Oncemi As Boolean
Magaza_Kapanisi = Sheets("Genel_Bilgiler").Cells(7, 2)
ReDim Ort_Kuyruk(7, Magaza_Kapanisi)
ReDim Var_Kuyruk(7, Magaza_Kapanisi)
ReDim Saatlik_Musteri(7, Magaza_Kapanisi)
Dim Dongu As Integer
For i = 1 To 7
   For j = 1 To Magaza_Kapanisi
      Ort_Kuyruk(i, j) = 0
      Var_Kuyruk(i, j) = 0
      Saatlik_Musteri(i, j) = 0
   Next j
Next i
For Dongu = 1 To Tekrar_Sayisi
   Randomize
   For i = 1 To 7
      j = 1
      Saat = 0
      Sure = Ustel_Rastgele(Ustel_Lamda(Sheets("mağaza_perf").Cells(3, i * 3)))
      Saat = Saat + Sure
      Musteriler(1, 1, i) = Saat
      Urun_Sayisi = Poisson_Rastgele(Sheets("mağaza_perf").Cells(3, i * 3 + 1))
      Musteriler(1, 2, i) = Urun_Sayisi
      m = 0
      Kasiyer_Sec = Round(Rnd() * Kac_Kasiyer_Calisiyor(i, 1), 0)
      UserForm1.ListBox1.Clear
      For n = 1 To Kasiyer_Sayisi
         If Kasiyer_Vardiyasi(i, n, Int(Musteriler(j, 1, i))) = 1 Then
            UserForm1.ListBox1.AddItem n
         End If
      Next n
      Gecerli_Saat = Int(Musteriler(j, 1, i))
      If UserForm1.ListBox1.ListCount <> 0 Then Musteriler(j, 3, i) =
UserForm1.ListBox1.List(Int(UserForm1.ListBox1.ListCount * Rnd()))
      For n = 1 To Musteriler(j, 2, i)
         kkk = Rnd()
         If kkk < 0.84 Then
            Okutma = 1
         ElseIf kkk < 0.89 Then
            Okutma = 2
         ElseIf kkk < 0.9 Then
            Okutma = 3
         Else
            Okutma = 4
         End If
         If Musteriler(j, 3, i) > 0 Then
            Musteriler(j, 4, i) = Musteriler(j, 4, i) +
Ustel_Rastgele(Sheets("Kasiyer_Perf").Cells(Musteriler(j, 3, i) + 2, Okutma + 2))
         End If
      Next n
      Musteriler(j, 4, i) = Musteriler(j, 4, i) / 600
      Musteriler(j, 7, i) = Musteriler(j, 1, i) + Musteriler(j, 4, i)
      Gecerli_Saat = -1
```

```
        Musteriler(j, 6, i) = 0 'ilk müşterinden önce kasada kaç kişi var
        Musteriler(j, 5, i) = Musteriler(j, 1, i) 'ilk müşterinin işleme başlama anı
        While Saat <= Saat_Dilimi - 1
            Sure = Ustel_Rastgele(Ustel_Lamda(Sheets("mağaza_perf").Cells(Int(Saat) + 3, i * 3)))
            Saat = Saat + Sure
            If Saat > 14 Then GoTo gun_sonu
            j = j + 1
            Musteriler(j, 1, i) = Saat 'müşterinin kasaya geliş anı
            Urun_Sayisi = Poisson_Rastgele(Sheets("mağaza_perf").Cells(Int(Saat) + 3, i * 3 + 1))
            If Urun_Sayisi <= 0 Then Urun_Sayisi = 1
            Musteriler(j, 2, i) = Urun_Sayisi
            If Gecerli_Saat <> Int(Musteriler(j, 1, i)) Then
                UserForm1.ListBox1.Clear
                For n = 1 To Kasiyer_Sayisi
                    If Kasiyer_Vardiyasi(i, n, Int(Musteriler(j, 1, i))) = 1 Then
                        UserForm1.ListBox1.AddItem n
                    End If
                Next n
                Gecerli_Saat = Int(Musteriler(j, 1, i))
            End If
            If UserForm1.ListBox1.ListCount <> 0 Then Musteriler(j, 3, i) =
UserForm1.ListBox1.List(Int(UserForm1.ListBox1.ListCount * Rnd()))
            Musteriler(j, 4, i) = 0
            For n = 1 To Musteriler(j, 2, i)
                kkk = Rnd()
                If kkk < 0.84 Then
                    Okutma = 1
                ElseIf kkk < 0.89 Then
                    Okutma = 2
                ElseIf kkk < 0.9 Then
                    Okutma = 3
                Else
                    Okutma = 4
                End If
                If Musteriler(j, 3, i) > 0 Then
                    Musteriler(j, 4, i) = Musteriler(j, 4, i) +
Ustel_Rastgele(Sheets("Kasiyer_Perf").Cells(Musteriler(j, 3, i) + 2, Okutma + 2))
                End If
            Next n
            Musteriler(j, 4, i) = Musteriler(j, 4, i) / 600
            Musteriler(j, 6, i) = 0
            Oncemi = True
            For n = 1 To j - 1
                If Musteriler(j, 3, i) = Musteriler(j - n, 3, i) And Musteriler(j, 1, i) < Musteriler(j - n, 7, i)
Then
                    Musteriler(j, 6, i) = Musteriler(j, 6, i) + 1
                    If Oncemi Then
                        Musteriler(j, 8, i) = j - n
                        Oncemi = False
                    End If
                End If
                If n > 50 Then
                    Exit For
                End If
            Next n
            If Musteriler(j, 6, i) = 0 Then 'müşterinin önünde kimse yoksa
                Musteriler(j, 5, i) = Musteriler(j, 1, i) 'işleme başlama anı = kasaya geliş anı
            Else 'müşterinin önünde başkaları varsa
                Musteriler(j, 5, i) = Musteriler(Musteriler(j, 8, i), 7, i)
            End If
```

```
            Musteriler(j, 7, i) = Musteriler(j, 5, i) + Musteriler(j, 4, i)
            If j Mod 300 = 0 Then DoEvents
        Wend
gun_sonu:
        Musteri_Sayisi(i) = j
    Next
    For k = 1 To 7
        For j = 1 To Musteri_Sayisi(k)
            Ort_Kuyruk(k, Int(Musteriler(j, 1, k))) = Ort_Kuyruk(k, Int(Musteriler(j, 1, k))) + Musteriler(j,
6, k)
            Var_Kuyruk(k, Int(Musteriler(j, 1, k))) = Var_Kuyruk(k, Int(Musteriler(j, 1, k))) + Musteriler(j,
6, k) ^ 2
            Saatlik_Musteri(k, Int(Musteriler(j, 1, k))) = Saatlik_Musteri(k, Int(Musteriler(j, 1, k))) + 1
        Next j
    Next k
Sheets("çıktı").Label3.Caption = Dongu & ". Simulasyon"
Next Dongu
For k = 1 To 7
    For i = 0 To Magaza_Kapanisi
        If Saatlik_Musteri(k, i) <> 0 Then
            Ort_Kuyruk(k, i) = Ort_Kuyruk(k, i) / Saatlik_Musteri(k, i)
            Var_Kuyruk(k, i) = Var_Kuyruk(k, i) / Saatlik_Musteri(k, i) - Ort_Kuyruk(k, i) ^ 2
        End If
    Next i
Next k
End Function
```

KAYNAK

[1] Nafees, A., *Queuein Theory And Its Application: Analysis Of The Sales Checkout Operation In ICA Supermarket.* Lisansüstü Tezi, University of Dalarna Department of Economics and Society **2007.**

[2] Giffin, & W.C., Queueing: Basic Theory and Applications. *Columbus, Ohio: Grid Publishing Company, 356p* , 3., **1978**

[3] Sarıaslan, D., *Sıra Bekleme Sistemlerinde Simulasyon (Benzetim) Tekniği.* Ankara: Ankara Üniversitesi Siyasal Bilgiler Fakültesi Yayınları, Ankara, **1986**.

[4] Raychaudhuri, S., Introduction to Monte Carlo Simulation. *Winter Simulation Conference*, (s. 1). Miami, **2008**.

[5] Kochanski, G., Monte Carlo Simulation,. http://kochanski.org /gpk/teaching/0501Oxford/MonteCarlo.pdf (Aralık **2014**)

[6] Lopez, L., Carter, M. W., & Gendreau, M., The hot strip mill production scheduling problem: A tabu g problem: A tabu. *European Journal of Operational Research* , 317-335, **1998**.

[7] Chelouah, R., & Siarry, P., Tabu Search applied to global optimization. *European Journal of Operational Researc* , 256-270, **2000**.

[8] Glover, F., Tabu Search: A Tutorial http://www.ida.liu.se/~zebpe/ heuristic/papers/TS_tutorial.pdf (Kasım **2014**)

[9] James, R., & Buchanan, J., Performance Enhancements to Tabu Search for the Early/Tardy Scheduling Problem. *European Journal Of Operational Research* , 254-265, **1998**.

[10] Dodin, B., Elimam, A., & Rolland, E., Tabu search in audit scheduling. *European Journal of Operational Research* , 373-392, **1998**.

[11] Gupta, J., Das, S., & Khumawala, B., A Savings Index Heuristic Algorithm for Flowshop Scheduling with Sequence Dependent Set-up Times. *The Journal of the Operational Research Society* , 1365-1373, **1995**.

[12] Logendran, R., & Sonthinen, A., A Tabu Search-Based Approach for Scheduling Job-Shop Type Flexible Manufacturing Systems. *The Journal of the Operational Research Society* , 264-277, **1997**.

[13] Nowicki, E., & Smutnicki, C., A Fast Taboo Search Algorithm for the Job Shop Problem. *Maangemen Science* , 797-813, **1996**.

[14] Sztrik, D. J., *Basic Queueing Theory*. University of Debrecen, Faculty of Informatics, **2012**.

ÖZGEÇMİŞ

Kimlik Bilgileri

Adı Soyadı : Ömür GÜRBÜZ

Doğum Yeri : Ankara

Medeni Hali : Evli

E-Posta : omurgurbuz@gmail.com

Adresi : Atilla Mah. 460. Sok. No:25 Daire 4 Konak İzmir

Eğitim

Lise : Ankara Atatürk Lisesi

Lisans : Hacettepe Üniversitesi İstatistik Bölümü

Yabancı Dil ve Düzeyi

İngilizce (ÜDS notu 65)

İş Deneyimi

Tesco Kipa Kitle Pazarlama Ticaret Lojistik ve Gıda Sanayi A.Ş.

Small Format Range Analyst (2010 - 2014)

Deneyim Alanları

-

Tezden Üretilmiş Projeler ve Bütçesi

-

Tezden Üretilmiş Yayınlar

-

Tezden Üretilmiş Tebliğ ve/veya Poster Sunumu ile Katıldığı Toplantılar

-