

T.C.
GEBZE TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YENİ NESİL DİZİLEMEDE OPTİMAL İŞ AKIŞI
ÇİZELGELEME

GÜLÇİN YILDIZ
YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

GEBZE
2023

T.C.
GEBZE TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YENİ NESİL DİZİLEMEDE OPTİMAL İŞ
AKIŞI ÇİZELGELEME

GÜLÇİN YILDIZ
YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

DANIŞMANI
PROF. DR. FATİH ERDOĞAN SEVİLGİN

GEBZE
2023

T.R.
GEBZE TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED
SCIENCES

OPTIMAL WORKFLOW SCHEDULING
FOR NEXT GENERATION SEQUENCING

GÜLÇİN YILDIZ
A THESIS SUBMITTED FOR THE DEGREE OF
MASTER OF SCIENCE
DEPARTMENT OF COMPUTER ENGINEERING

THESIS SUPERVISOR
PROF. DR. FATİH ERDOĞAN SEVİLGİN

GEBZE
2023



YÜKSEK LİSANS JÜRİ ONAY FORMU

GTÜ Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 10/07/2023 tarih ve 2023/38 sayılı kararıyla oluşturulan jüri tarafından 02/08/2023 tarihinde tez savunma sınavı yapılan Gülçin Yıldız'ın tez çalışması Bilgisayar Mühendisliği Anabilim Dalında YÜKSEK LİSANS tezi olarak kabul edilmiştir.

JÜRİ

ÜYE

(TEZ DANIŞMANI) : Prof. Dr. Fatih Erdoğan SEVİLGİN

ÜYE

: Prof. Dr. Haluk Rahmi TOPÇUOĞLU

ÜYE

: Dr. Öğr. Üyesi Yakup GENÇ

ONAY

Gebze Teknik Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun
...../...../..... tarih ve/..... sayılı kararı.

İMZA/MÜHÜR

ÖZET

Çizelgeleme, işlemci kaynağı üzerinde bir veya birden fazla uygulamanın maksimum tamamlanma süresini etkileyen önemli bir unsurdur. Çok sayıda araştırma, uygulamaların veri bağımlılığı olan işlemlerini yönlü döngüsüz çizge (DAG) ile modelleyerek çizelgeleme problemini ele almıştır. Uygulamanın bir işlemi, birden fazla işlemci üzerinde yürütülebiliyorsa veri paralel ismini alır. DAG ile modellenen ve veri paralel işlemler içeren uygulamaların çizelgelemesi problemi için birçok çalışmada veri paralel işlemlere tahsis edilecek işlemci sayısının belirlenmesi (işlemci tahsisi) ve yürütme sırasının belirlenmesi ile işlemlerin uygun işlemcilere atanması (çizelgeleme) prosedürleri için çözüm geliştirilir. Bu tez çalışmasında da veri paralel işlemler içeren DAG ile modellenen uygulamaların homojen işlemci kaynağında statik ve dinamik çizelgeleme problemi ele alınmıştır.

Statik çizelgeleme problemi için geliştirilmiş CPA, MCPA, MCPA2, IAES gibi açgözlü algoritmalar vardır. Tez çalışmasında işlemci tahsisi prosedürü için açgözlü algoritmaların işlemci tahsisi sezgisellerini kullanan hiper-sezgisel bir yöntem geliştirilmiştir. Hiper-sezgisel yöntem ile açgözlü algoritmaların işlemci tahsisindeki güçlü yönlerinden faydalanılmıştır.

Dinamik çizelgeleme problemi için literatürde az sayıda çalışma olduğu gözlemlenmiştir. Dinamik çizelgeleme probleminde iş akışlarının sisteme varış zamanı bilinemez. Tez çalışması kapsamında, iş akışlarının sisteme varış zamanında işlemci tahsisi ve çizelgelemeyi tekrar tetiklemesi ile çözüme ulaşılr. İşlemci tahsisinde geliştirilen hiper-sezgisel yöntem kullanılmıştır. Çizelgeleme prosedürü için de genetik algoritma (meta-sezgisel yöntem) ile çözüm geliştirilmiştir.

Deneyssel çalışmalarda statik çizelgelemede önerilen algoritma CPA algoritmasına göre ortalama %14, MCPA algoritmasına göre %7, MCPA2 algoritmasına göre %9, IAES algoritmasına göre %5 daha başarılı sonuç vermiştir. Dinamik çizelgelemede ise önerilen algoritma ile hiper-sezgisel işlemci tahsisi CPA ile işlemci tahsisine göre %16, IAES ile işlemci tahsisine göre %14, MOWS algoritmasına göre %20 daha başarılı sonuç vermiştir.

Anahtar Kelimeler: Dinamik Çizelgeleme, Statik Çizelgeleme, İşlemci Tahsisi, Veri Paralel, Hiper-sezgisel.

SUMMARY

Scheduling is an important factor affecting the maximum completion time of one or more applications on the processor resource. In many studies, the scheduling problem has been expressed by modeling the data dependencies of applications with a directed acyclic graph (DAG). If a task of the application can be executed on more than one processor, it is called data parallel. In the scheduling problem of the application represented by DAG and containing data parallel tasks, the stages of determining the number of processors to be allocated to the data parallel tasks (processor allocation) and determining the execution order (scheduling) are considered. In this thesis, the problem of static and dynamic scheduling of applications including data parallel tasks and modeled by DAG on homogeneous processor resource is solved.

In the static scheduling problem, there are greedy algorithms such as CPA, MCPA, MCPA2, IAES. In this thesis, a hyper-heuristic method using heuristics of greedy algorithms is developed for the processor allocation. The hyper-heuristic method takes advantage of the processor allocation strengths of heuristics.

It has been observed that there are few studies in the literature for workflows in the dynamic scheduling problem. In the dynamic scheduling problem, the arrival time of the workflows in the system are not known. In the thesis study, a genetic algorithm (meta-heuristic method) solution has been developed for the scheduling procedure of the dynamic scheduling problem. For the processor allocation, the developed hyper-heuristic method is used.

In experimental studies, the algorithm recommended for static scheduling has yielded an average of 14% better results compared to the CPA algorithm, 7% better results compared to the MCPA algorithm, 9% better results compared to the MCPA2 algorithm, and 5% better results compared to the IAES algorithm. In dynamic scheduling, the proposed algorithm has resulted in 16% better processor allocation compared to processor allocation with CPA, 14% better processor allocation compared to IAES, and 20% better results compared to the MOWS algorithm.

Key Words: Dynamic Scheduling, Static Scheduling, Processor Allocation, Data Parallelism, Hyper-heuristic.

TEŞEKKÜR

Başta, yüksek lisans eğitimimde ve akademik hayatımda desteğini ve yardımlarını hiçbir zaman esirgemeyip bilgisi ile bu çalışmanın oluşmasının yolunu açan danışmanım Prof. Dr. Fatih Erdoğan Sevilgen'e,

Bütün çalışmam boyunca yanımda olan, göstermiş olduğu desteklerinden dolayı sevgili annem Emine Özbay'a, ablam Elvin Bolat'a, eşim Mehmet Ali Yıldız'a ve tüm değerli aile üyelerime en içten teşekkürlerimi sunarım.



İÇİNDEKİLER

	Sayfa
ÖZET	v
SUMMARY	vi
TEŞEKKÜR	vii
İÇİNDEKİLER	viii
SİMGELER ve KISALTMALAR DİZİNİ	x
ŞEKİLLER DİZİNİ	xi
TABLolar DİZİNİ	xiv
1. GİRİŞ	1
1.1. Tezin Amacı ve Katkısı	4
1.2. Tezin İçeriği	6
2. PROBLEM TANIMI VE İLGİLİ ÇALIŞMALAR	7
2.1. Çizelgeleme	8
2.1.1. Statik Çizelgeleme	9
2.1.1.1. CPA	9
2.1.1.2. MCPA	10
2.1.1.3. MCPA2	12
2.1.1.4. IAES	13
2.1.1.5. Diğer Çalışmalar	13
2.1.2. Dinamik Çizelgeleme	16
2.1.2.1. GA-ECT	16
2.1.2.2. HGA	17
2.1.2.3. MOWS	18
2.1.2.4. Diğer Çalışmalar	18
3. YENİ NESİL DİZİLEME	21
3.1. Yeni Nesil Dizilemede Çalışma Zamanı Modeli	23
4. HİPER SEZGİSEL YÖNTEM İLE İŞLEMCİ TAHSİSİ	32
4.1. Genetik Algoritma	32
4.2. Genetik Algoritma Kullanımı	33
4.2.1. Amaç Fonksiyonu	35

4.2.2. Çaprazlama	35
4.2.3. Mutasyon	36
4.3. Statik Çizelgeleme Deneysel Sonuçları	37
4.3.1. Standart İşlem Çizgeleri	38
4.3.2. Uygulama Çizgeleri	46
4.3.3. Yeni Nesil Dizileme İş Akışı Çizgesi	52
5. GELİŞTİRİLEN DİNAMİK ÇİZELGELEME MODELİ	58
5.1. Genetik Algoritma Kullanımı	60
5.1.1. Çaprazlama	62
5.1.2. Mutasyon	62
5.2. Dinamik Çizelgeleme Deneysel Sonuçları	63
5.2.1. Yeni Nesil Dizileme İş Akışı Çizgesi	64
5.2.2. Standart İşlem Çizgeleri	67
5.2.3. Uygulama Çizgeleri	68
6. SONUÇLAR	70
KAYNAKLAR	72
ÖZGEÇMİŞ	78
EKLER	79

SİMGELER VE KISALTMALAR DİZİNİ

Simgeler ve Açıklamalar Kısaltmalar

BAM	: Binary Alignment/Map
BWA	: Burrows-Wheeler Aligner Wheeler
BWT	: Burrows-Wheeler Transform
DAG	: Directed Acyclic Graph
DNA	: Deoksiribonükleik asit
GA	: Genetik Algoritma
GATK	: Genome Analysis ToolKit
GB	: Gigabayt
GTÜ	: Gebze Teknik Üniversitesi
SAM	: Sequence Alignment/Map
SLR	: Schedule Length Ratio
Sn	: Saniye
STG	: Standart Task Graph
YND	: Yeni Nesil Dizileme

ŞEKİLLER DİZİNİ

<u>Sekil No:</u>	<u>Sayfa</u>
1.1: DAG.	2
1.2: Örnek bir DNA dizisi biyoinformatik analiz iş akışı.	4
2.1: CPA işlemci tahsisi sözde kodu.	10
2.2: MCPA DAG.	11
2.3: MCPA işlemci tahsisi sözde kodu.	12
2.4: a) Kromozom, b) Yeni gelen işlemler ile henüz yürütülmeye başlamamış işlemler ve yürütülmeye başlamış işlemlerin atandığı işlemciler, c) Yeni kromozom.	17
3.1: Kırpma işleminin çalışma süresi tahmini için doğrusal model.	24
3.2: Çalışma zamanı tahmini için doğrusal model ve Karp-Flatt metriği.	26
3.3: Hizalama işleminde bir işlemcide çalışma süresi tahmini için doğrusal model.	26
3.4: Dosya dönüştürme işleminin çalışma süresi tahmini için doğrusal model.	29
3.5: Sıralama işleminin çalışma süresi tahmini için doğrusal model.	29
3.6: Genotip olasılıkların üretilmesi işleminin çalışma süresi tahmini için logaritmik model.	30
3.7: Varyant arama işleminin çalışma süresi tahmini için logaritmik model.	31
4.1: Kromozom.	34
4.2: Liste Çizelgeleme Sözde Kodu.	35
4.3: Hiper-sezgisel GA Çaprazlama.	36
4.4: Hiper-sezgisel GA Mutasyon.	36
4.5: 50 işlem içeren 30 çizge ile 8 işlemcili kaynakta ağgözlü algoritmalar ile makespan karşılaştırması.	39
4.6: 50 işlem içeren 30 çizge ile ağgözlü algoritmalar ile SLR karşılaştırması.	40
4.7: 50 işlem içeren 30 çizge ile 8 işlemcili kaynakta rastgele ve permütasyon çözümleri ile makespan karşılaştırması.	41
4.8: 50 işlem içeren 30 çizge ile rastgele ve permütasyon çözümleri ile SLR karşılaştırması.	42
4.9: 100 işlem içeren 30 çizge ile 8 işlemcili kaynakta ağgözlü algoritmalar ile makespan karşılaştırması.	43

4.10:	100 işlem içeren 30 çizge ile ağgözlü algoritmalar ile SLR karşılaştırması.	44
4.11:	100 işlem içeren 30 çizge 8 işlemcili kaynakta rastgele ve permütasyon çözümleri ile makespan karşılaştırması.	45
4.12:	100 işlem içeren 30 çizge rastgele ve permütasyon çözümleri ile SLR karşılaştırması.	46
4.13:	Matmul için ağgözlü algoritmalar ile makespan ve SLR karşılaştırması.	47
4.14:	Matmul için rastgele ve permütasyon çözümleri ile makespan ve SLR karşılaştırması.	47
4.15:	Strassen için ağgözlü algoritmalar ile makespan ve SLR karşılaştırması.	48
4.16:	Strassen için rastgele ve permütasyon çözümleri ile makespan ve SLR karşılaştırması.	49
4.17:	Spec fppp için ağgözlü algoritmalar ile makespan ve SLR karşılaştırması.	50
4.18:	Spec fppp için rastgele ve permütasyon çözümleri ile makespan ve SLR karşılaştırması.	50
4.19:	Robot Kontrol için ağgözlü algoritmalar ile makespan ve SLR karşılaştırması.	51
4.20:	Robot Kontrol için rastgele ve permütasyon çözümleri ile makespan ve SLR karşılaştırması.	52
4.21:	YND analizi iş akışı yönlü döngüsüz çizgesi.	53
4.22:	YND analizi 8 iş akışı için ağgözlü algoritmalar ile makespan ve SLR karşılaştırması.	54
4.23:	YND analizi 8 iş akışı için rastgele ve permütasyon çözümleri ile makespan ve SLR karşılaştırması.	55
4.24:	YND analizi 20 iş akışı için ağgözlü algoritmalar ile makespan ve SLR karşılaştırması.	56
4.25:	YND analizi 20 iş akışı için rastgele ve permütasyon çözümleri ile makespan ve SLR karşılaştırması.	57
5.1:	Dinamik Çizelgeleme için önerilen akış şeması.	59
5.2:	Genetik Algoritma Sözde Kodu.	60
5.3:	a) İşlemler ve bağımlılıkları, b) Kromozom.	61
5.4:	Çaprazlama.	62
5.5:	Mutasyon.	63
5.6:	YND analizi 8 iş akışı için ağgözlü algoritmalar ile makespan karşılaştırması.	65
5.7:	YND analizi 8 iş akışı için rastgele ve permütasyon çözümleri ile makespan karşılaştırması.	65

5.8:	YND analizi 20 iş akışı için açgözlü algoritmalar ile makespan karşılaştırması.	66
5.9:	YND analizi 20 iş akışı için rastgele ve permütasyon çözümleri ile makespan karşılaştırması.	66
5.10:	50 işlem içeren 5 iş akışı için açgözlü algoritmalar ile makespan karşılaştırması.	67
5.11:	50 işlem içeren 5 iş akışı için rastgele ve permütasyon çözümleri ile makespan karşılaştırması.	68
5.12:	4 uygulama için açgözlü algoritmalar ile makespan karşılaştırması.	69
5.13:	4 uygulama için rastgele ve permütasyon çözümleri ile makespan karşılaştırması.	69



TABLÖLAR DİZİNİ

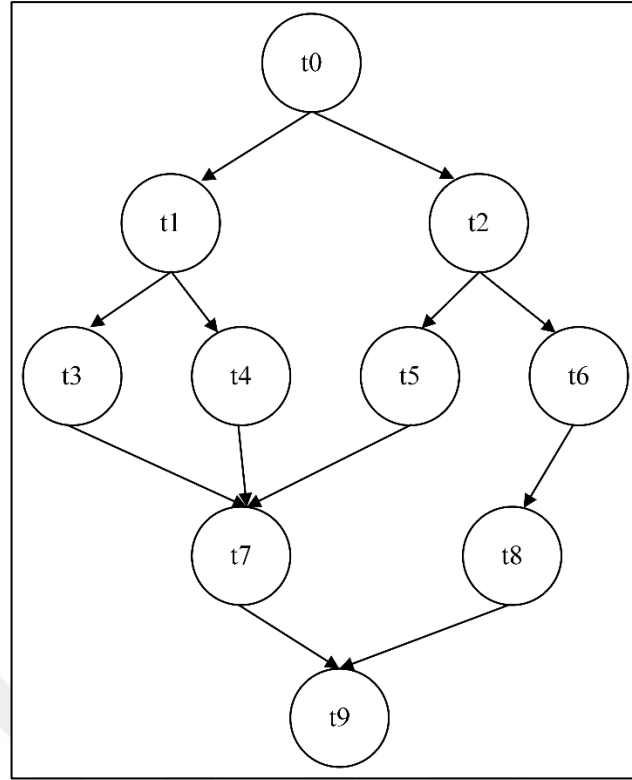
<u>Tablo No:</u>	<u>Sayfa</u>
3.1: Dosya Boyutuna göre Paralellik Değerleri.	27
3.2: Dosya Boyutuna (GB) ve İşlemci Sayısına göre gerçek çalışma süresi(sn).	28
3.3: Dosya Boyutuna (GB) ve İşlemci Sayısına göre F_{est} değeri ile hesaplanan tahmini çalışma süresi (sn).	28
B1.1: Tez çalışmasında Türkçe olarak kullanılan İngilizce terimler.	79

1. GİRİŞ

Çizelgeleme, kısıtlı kaynakların amaç fonksiyonuna uygun olarak işlemlere tahsis edilmesidir. Geliştirilmekte ve mevcut olan bilimsel uygulamaların artan hesaplama kaynağı gereksinimini karşılamak için paralel işlemlerin çok işlemcili sistemler üzerinde çizelgelenmesi genel yürütme süresini etkileyen güçlü bir araçtır. Günümüze kadar yapılan çalışmalarda görüntü işlemeden iklim modellenmesine kadar birçok bilimsel uygulamada çizelgelemenin faydaları görülmektedir. Bu çalışmada genel çizelgeleme probleminin çözülmesi yanında özellikle biyoinformatik alanındaki yeni nesil dizileme analizi iş akışları hedeflenmiştir.

Çizelgeleme probleminde paralel işlemler görev paralel ve veri paralel olmak üzere iki modelde tanımlanır. Farklı işlemciler üzerinde eşzamanlı yürütülen işlemler görev paraleldir. Birden fazla işlemci üzerinde yürütülebilen bir işlem ise veri paraleldir. Veri paralel bir işlemin her bir veri parçası farklı işlemcide yürütülebilir. Veri paralel işlemler katı, kalıplanabilir, biçimlendirilebilir olmak üzere üç çeşittir. Çizelgeleme sürecinden önce her bir işleme atanan işlemci sayısı belirli ise katı işlemdir. İşleme atanacak işlemci sayısı çizelgeleme sürecinde belirleniyor ise kalıplanabilir işlemdir. Biçimlenebilir işlemler ise atanacak işlemci sayısı çalışma zamanında belirlenen işlemlerdir.

Birçok araştırmada, bilimsel uygulamaların çizelgeleme problemi yönlü döngüsüz çizge (DAG) ile modellenmiştir. Yönlü döngüsüz çizgede her bir düğüm uygulamanın bir işlemini ifade eder. Düğüm maliyeti, düğümün temsil ettiği işlemin bir veya birden fazla işlemci üzerinde çalışma maliyetini gösterir. İşlemler arasındaki veri aktarımı çizgedeki kenarlar ile temsil edilir. Şekil 1.1’de yönlü döngüsüz çizge ile bir uygulamanın işlemleri ve işlem bağımlılıkları gösterilmiştir. Bu çizgede t_0 başlangıç işlemi iken t_9 son işlemdir. İşlem t_0 ’ın veri çıktısı t_1 ve t_2 işlemleri için giriş verisini oluşturur. Uygun işlemcilerin bulunması halinde t_1 ve t_2 eşzamanlı olarak farklı işlemcilerde yürütülebilir. Çizgedeki tüm işlemler veri bağımlılığı bulunan işlem veya işlemlerin veri çıktısını giriş verisi olarak kullanır. Bu veri bağımlılıkları ile tüm işlemler tamamlanır.



Şekil 1.1: DAG.

İşlemci kaynağının türü çizelgeleme problemlerinde önemli bir faktördür. Çok işlemcili sistemde kaynak türü homojen ve heterojen kaynak olarak ikiye ayrılır. HOMOJEN kaynaklı sistemlerde tüm işlemciler aynı işlem kapasitesine sahiptir. Heterojen kaynaklı sistemlerde ise farklı işlem kapasitesine sahip işlemciler bulunmaktadır.

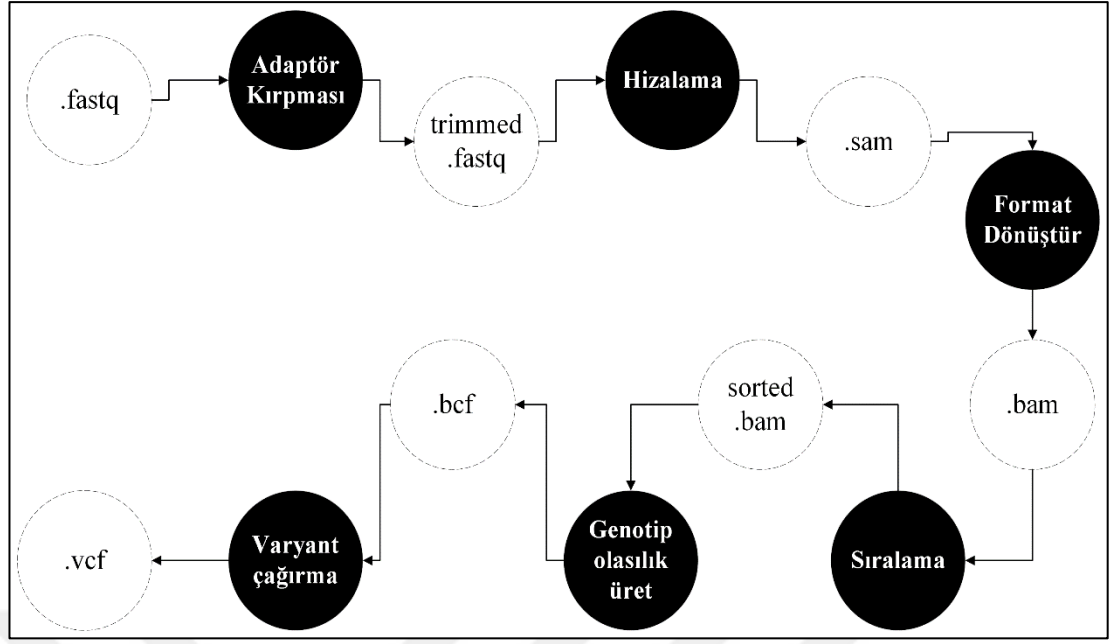
Çizelgeleme problemi statik ve dinamik çizelgeleme olarak ikiye ayrılır. Statik çizelgelemede tüm işlemlerin işlemci kaynağına varış zamanı bellidir. İşlem bağımlılıkları, işlem sayısı, işlemci kaynağın kapasitesi bilinir ve zaman içinde değişiklik göstermez. Dinamik çizelgeleme probleminde ise problemin doğası zaman içinde değişebilmektedir. İşlemlerin işlemci kaynağına varış zamanı bilinmez. İşlemci kaynağında hata, kaza vs. gibi durumlardan kaynaklı işlemci sayısı değişebilir. Donanımsal ya da yazılımsal hatalara bağlı olarak işlem sayısı değişkenlik gösterebilir.

DAG ile modellenen ve veri paralel işlemler içeren problemlerin çizelgeleme çözümü için günümüze kadar birçok çalışma yapılmıştır. Bazı çalışmalarda veri paralel işlemler için tahsis edilecek işlemci sayısının belirlenmesinde işlemci tahsisi algoritmaları ile çözüm bulunurken çizelgeleme çözümü için ayrı bir süreç iletirmektedir. Çok işlemcili bir hesaplama kaynağı üzerinde bir işlemin sadece bir

işlemci kullanarak öncelik kısıtları ile çizelgeleme probleminin NP-Zor olduğu gösterilmiştir [1]. Veri paralel işlemler içeren iş akışına sahip uygulamaların öncelik kısıtları ile çizelgeleme probleminin daha zor bir problem olduğu görülmektedir.

Yeni nesil dizileme verisinin analizi de hesaplama kaynağı gereksinimi yüksek işlemler içeren bir problemdir. Dizileme işlemi DNA'nın nükleik asit moleküllerinin sıralamasını belirlemek için kullanılan bir yöntemdir. Dizileme biyolojik çalışmalar için büyük önem taşır. Bu yüzden dizileme işlemi için birçok farklı yöntem ortaya çıkmıştır. İnsan Genom projesi DNA dizileme analizinde yapılan ilk büyük çalışmadır. Yaklaşık 13 yıl süren çalışma 2003 yılında tamamlanmıştır. Bu proje birinci nesil dizileme yöntemi olan Sanger dizileme ile gerçekleştirilmiştir. Projenin yüksek maliyeti ve hesaplama süresinin kısaltılması amacıyla yeni çözüm arayışlarına girilmesi yeni nesil dizileme teknolojisi ortaya çıkmasını sağlamıştır. Yeni nesil dizilemede milyonlarca DNA parçası için paralel olarak dizileme yapılırken Sanger dizilemede ise tek seferde bir DNA parçası için dizileme yapılabilmektedir. Bu durum yeni nesil teknoloji ile çok daha düşük maliyet ile ve çok yüksek hızlarda dizileme yapılabilmesini sağlamıştır.

Yeni nesil dizileme genel kapsamda DNA parçalama, kütüphane oluşturma, dizileme, dizilerin birleştirilmesi ve biyoinformatik analiz aşamalarından oluşmaktadır. Biyoinformatik analiz aşamasında elde edilen büyük boyuttaki DNA dizisi verisinin analiz süreci genel olarak okumaların kırılması, hizalama, sıralama, varyant çağırma gibi işlemlerden oluşur. Şekil 1.2'de DNA dizileme iş akışı ve işlemleri görülmektedir. Analiz iş akışında düşük kalitede olan okumalar için kırma işlemi gerçekleşir. Hizalama işlemi ile okumalar referans genoma eşlenir. Hizalama işlemi çıktıları sıralanarak varyantların çıkarılması için varyant çağırma işlemi gerçekleştirilir. Biyoinformatik analiz süreci bir işlemin çıktısı diğer işlemin girdisi olacak şekilde tasarlanmış bir iş zinciridir. Bu işlemler çeşitli yazılım paketleri ile gerçekleştirilir. Bu işlemlerden bazıları veri paralel işlem özelliği gösterirken bazı işlemler ise tek bir işlemci üzerinde yürütülür.



Şekil 1.2: Örnek bir DNA dizisi biyoinformatik analiz iş akışı.

1.1. Tezin Amacı ve Katkısı

Yeni nesil dizilemede bir örnekten elde edilen DNA dizilerinin biyoinformatik analizi saatlerce sürmektedir. Klinik araştırma hedefli çalışmalarda yüzlerce DNA dizileme verisi için benzer analiz çalışmalarının yapılması gerekir. Bu durum oldukça yüksek hesaplama kaynağı gereksinimi ve zaman maliyeti oluşturmaktadır. Bu tez çalışmasında statik ve dinamik çizelgeleme problemleri için önerilen algoritmalar ile yeni nesil dizileme analizi iş akışları için de çözüm sunulmuştur. Sistem aynı kapasiteye sahip işlemcilerden oluşan homojen hesaplama kaynağıdır. Veri paralel işlemler kalıplanabilir işlemlerdir.

Yeni nesil dizileme ile bir DNA dizileme analizi, veri paralel işlemler içermesi ve işlemler arasında veri aktarım bağımlılıkları içermesi yönünden statik çizelgeleme problemi olarak ele alındığında yönlü döngüsüz çizge ile modellenenabilir. Birden fazla iş akışı, sıfır maliyetli giriş ve çıkış düğümlerine bağlandığında yönlü döngüsüz çizge oluştururlar. Yapılan çalışmalarda yönlü döngüsüz çizge ile modelleme yapılmış gerçek dünya uygulamaları için çizelgeleme çözüm yaklaşımları, işlemci kaynağı verimliliğini artırırken zaman maliyetlerini azaltılmıştır. Ancak, bildiğimiz kadarıyla, literatürde yeni nesil dizileme iş akışı için çalışma zamanı modelleyerek çizelgeleme iyileştirmesi yapmış çalışma bulunmamaktadır.

Yönlü döngüsüz çizge ile modellenen statik çizelgeleme problemleri için yapılan çalışmalarda işlemci tahsisi ve çizelgeleme süreci iki aşamada ya da tek aşamada geliştirilmiş algoritmalar ile çözümlenmiştir. Tek aşamada çözüm geliştiren çalışmalarda algoritma her iterasyonda işlemci tahsisi ve çizelgeleme için bir çözüm geliştirip en iyi çözümde iterasyonu sona erdirir. İki aşamalı çalışmalarda ise işlemci tahsisi ve çizelgeleme için ayrı çözüm elde edilir. CPA [2], MCPA [3], MCPA2 [4] algoritmaları iki aşamada çözüm üreten açgözlü yaklaşımlardır. IAES [5], CPR [6] algoritmaları ise tek aşamada çözüm üreten açgözlü algoritmalar. İşlemci tahsisi ve çizelgeleme prosedürleri için farklı çalışmalarda genetik algoritma (meta-sezgisel) ile yapılan çalışmalar da bulunmaktadır [7], [8].

Bu tez çalışmasında hiper-sezgisel bir yaklaşım ile işlemci tahsisinde özgün bir çözüm önerilmektedir. Hiper-sezgisel yöntemler sezgisellerin nasıl kullanılacağı ile ilgili çözüm uzayında arama yaparak çözümün iyileşmesini hedefler. Hiper-sezgisel yöntem ile kalıplanabilir veri paralel işlemlerin işlemci tahsisi gerçekleşir. Literatürdeki çalışmalarda işlemci tahsisi için önerilmiş bir hiper-sezgisel yaklaşım bulunmamaktadır. Ayrıca hiper-sezgisel yöntem ile statik çizelgeleme problemi de ele alınarak gerçek dünya uygulamaları ve standart işlem çizgeleri için deneysel çalışmalar yapılmıştır.

Yeni nesil dizileme analizi iş akışının çizelgelemesi, dinamik çizelgeleme problemi olarak da ele alınmıştır. DNA dizileme analizi yapılacak örneklerin ve dolayısı ile analizi yapılacak DNA verisinin hesaplama kaynağına varış zamanı bilinemez. DNA verisinin sadece sisteme varış zamanının bilinmemesi dinamikliğin oluşmasına neden olmuştur.

Veri paralel işlem içeren ya da içermeyen dinamik çizelgeleme problemleri için literatürde farklı yöntemler ortaya konmuştur. MOWS [9] açgözlü algoritması veri paralel işlemler içeren dinamik çizelgeleme problemi için geliştirilmiştir. Ayrıca veri paralel işlem içermeyen problemler için geliştirilmiş genetik algoritma ve karınca sürüsü optimizasyonu gibi meta-sezgisel yaklaşımlarda bulunmaktadır [10], [11].

Bu tez çalışmasında veri paralel işlemler içeren ayrıca işlemler arasında veri bağımlılığı bulunan yeni nesil dizileme iş akışları için meta-sezgisel bir yöntem ile homojen işlemci kaynağında dinamik çizelgeleme problemine çözüm geliştirilmiştir. İşlemci tahsisi aşamasında hiper-sezgisel yöntem kullanılmıştır.

Statik ve dinamik çizelgeleme problemleri için yapılan bu çalışmalar sadece yeni nesil dizileme iş akışı için değil veri paralel işlemler içeren ve veri bağımlılığı bulunan

benzer tüm problem kümeleri için çözüm yöntemi olacaktır. Deneysel çalışmalarda yeni nesil dizileme iş akışı haricinde standart işlem çizgesi ve çeşitli gerçek dünya uygulamaları ile elde edilen sonuçlar geliştirilen yöntemin çeşitli çizge türlerinde fayda sağlayacağını göstermektedir.

1.2. Tezin İçeriği

Giriş bölümünde; veri paralel işlemler içeren ve veri bağımlılığı bulunan iş akışlarının çizelgeleme problemine dair genel bilgileri; tez çalışmasının amacı ve katkılarından bahsedilmektedir. İkinci bölümde; statik ve dinamik çizelgeleme problemi ayrı ayrı ele alınarak problemler için çözümler sunan literatür çalışmalarından bahsedilmektedir. Üçüncü bölümde yeni nesil dizileme ve yeni nesil dizileme analizindeki işlemlerin çalışma zamanının modellenmesi anlatılmaktadır. Dördüncü bölümde ise hiper-sezgisel yöntem ile işlemci tahsisi detayları ile anlatılmaktadır. Ayrıca statik çizelgeleme problemi olarak ele alınarak çeşitli gerçek dünya problemleri, standart işlem çizgeleri ve yeni nesil dizileme analizi iş akışı ile yapılan deneysel çalışmaların sonuçları verilmiştir. Beşinci bölümde ise meta-sezgisel yöntem ile dinamik çizelgeleme problem çözümü detayları ile anlatılmaktadır. Ayrıca dinamik çizelgeleme problemi olarak ele alınan yeni nesil dizileme iş akışı, gerçek dünya problemleri, standart işlem çizgeleri ile yapılan deneysel çalışmaların sonuçları verilmiştir. Son olarak altıncı bölümde ise tez çalışmasından çıkarılan sonuçlar bulunmaktadır.

2. PROBLEM TANIMI VE İLGİLİ ÇALIŞMALAR

Çizelgeleme probleminin çözümünde veri paralel işlemler için tahsis edilen işlemci sayısının belirlenmesinde işlemci tahsisi prosedürü, işlemlerin yürütme sırasının belirlenerek uygun işlemcilere atanması içinde çizelgeleme prosedürü uygulanır. Bu bölümde işlemci tahsisi ve çizelgeleme prosedürü için geliştirilen algoritmalar anlatılmadan önce problem tanımı verilmiştir. Literatür de veri paralel işlemlere tahsis edilecek işlemci sayısı çalışma zamanında veya önceden belirlenmemiş ise bu işlemler kalıplanabilir işlemler olarak tanımlanır. Bu tez çalışmasında kalıplanabilir işlemlerin homojen sistem üzerinde çizelgelenmesi problemi ele alınmıştır. Çok sayıda işlemci içeren homojen işlemci kaynağı $P = \{m_k : k = 1, 2, \dots, p\}$ ile ifade edilir. İşlemci kaynağı p tane işlemci içerir ve m_k bir işlemciyi ifade etmektedir. İşlemler kümesi $T = \{t_i : i = 1, 2, \dots, n\}$ ile ifade edilir. İşlemler kümesi n tane işlem içerir ve t_i bir işlemi ifade etmektedir. İşlemler arasında veri bağımlılıkları bulunur. $\text{Öncül}(t_i)$, t_i işleminden önce tamamlanması gereken işlemleri ifade eder. Yönlü döngüsüz çizge modelinde $\text{Öncül}(t_i)$ işlemler kümesinden t_i işlemine yönlü kenar ile veri bağımlılıkları ve veri akışı gösterilir. $\text{Ardıl}(t_i)$, t_i işleminden sonra başlaması gereken işlemleri ifade eder. Dinamik çizelgeleme probleminde işlemlerin varış zamanı $V = \{v_i : i = 1, 2, \dots, n\}$ kümesi ile gösterilir.

Eşitlik (2.1)'de gösterilen $T_{uygun}(t_i)$, t_i işlemine tahsis edilecek olan tüm işlemcilerin hazır olma zamanıdır. İşlem t_i 'ye tahsis edilen tüm işlemciler kümesi Q_i ile ifade edilir. İşlem t_i 'den önce Q_i işlemci kümesindeki m işlemcisinde tamamlanan son işlem t_x^m ile ifade edilir. $F(t_x^m)$, t_x^m işleminin bitiş zamanıdır.

$$T_{uygun}(t_i) = \max_{m \in Q_i} \{F(t_x^m)\} \quad (2.1)$$

$EBZ(t_i)$, t_i işleminin en erken başlama zamanıdır. Eşitlik (2.2)'de gösterildiği gibi öncül işlemlerin maksimum bitiş zamanıdır. İş akışlarında başlangıç işlemlerinin $EBZ(t_i)$ değeri ile v_i değeri aynıdır.

$$EBZ(t_i) = \max_{t_j \in \text{Öncül}(t_i)} \{F(t_j)\} \quad (2.2)$$

Eşitlik (2.3)'de gösterilen $S(t_i)$, t_i işleminin başlangıç zamanıdır. t_i 'nin öncül işlemlerinin bitiş zamanının ve tahsis edilen işlemci sayısı kadar işlemcinin uygun olduğu zamanın maksimumudur.

$$S(t_i) = \max\{EBZ(t_i), T_{uygun}(t_i)\} \quad (2.3)$$

Eşitlik (2.4)'da gösterilen $F(t_i)$, t_i işleminin bitiş zamanıdır. İşlem t_i 'nin başlangıç zamanı ve çalışma süresinin toplamıdır. $Np(t_i)$, t_i işlemine tahsis edilen işlemci sayısını ifade eder. $C(t_i, Np(t_i))$, t_i işleminin $Np(t_i)$ işlemcisindeki çalışma süresidir.

$$F(t_i) = S(t_i) + C(t_i, Np(t_i)) \quad (2.4)$$

Eşitlik (2.5)'de maksimum tamamlanma süresi (makespan) verilmiştir. Amaç fonksiyonumuz maksimum tamamlanma süresinin minimize edilmesidir.

$$makespan = \max(F(t_i)) \quad (2.5)$$

Bir işlemin öncül işlemi ile farklı işlemciler üzerinde yürütülmesi işlemciler arasında veri aktarımını gerektirir. Bu durum işlemler arasında iletişim maliyetinin oluşmasına neden olmaktadır. Günümüze kadar yapılmış çalışmalarda eğer bir işlem, öncül işlemi ile aynı işlemci üzerinde yürütülürse iletişim maliyeti sıfır olarak hesaplanır [12]. Yönlü döngüsüz çizgede kenar ağırlığı ile iletişim maliyeti ifade edilir. Bu tez çalışmasında işlemler arasındaki iletişim maliyeti sıfır kabul edilmiştir.

2.1. Çizelgeleme

Çizelgeleme problemi ile ilgili günümüze kadar yapılmış birçok çalışma bulunmaktadır. Yapılan çalışmaları statik ve dinamik çizelgeleme problemi olarak iki ana başlıkta inceleyeceğiz.

2.1.1. Statik Çizelgeleme

Veri paralel işlemler içeren ve yönlü döngüsüz çizge ile modellenen statik çizelgeleme problemlerinde açgözlü veya meta-sezgisel yöntemler ile çözüm geliştiren çalışmalar bulunmaktadır. Bu çalışmalarda işlemci tahsisi ve çizelgeleme prosedürleri bazı çalışmalarda tek aşamada çözüm oluştururken bazı çalışmalarda ise iki aşamada çözüm oluşturur. Tek aşamada çözüm geliştiren çalışmalarda algoritma her iterasyonda işlemci tahsisi ve çizelgeleme için bir çözüm geliştirir. İterasyonlar geliştirilebilen en iyi çözüme kadar devam eder. IAES [5], CPR [6] tek aşamadan oluşan algoritmalarıdır. İki aşamada çözüm geliştiren algoritmalar ise işlemci tahsisi ve çizelgeleme prosedürü ayrı ele alınır. İşlemci tahsisi prosedürü tamamlandıktan sonra işlemlere tahsis edilmiş işlemci sayılarına göre çizelgeleme gerçekleşir. CPA [2], MCPA [3], MCPA2 [4] iki aşamadan oluşan algoritmalarıdır. Heterojen işlemci kaynağı olan problemlere uygun açgözlü algoritmalar geliştirilmiştir [13]. Ayrıca statik çizelgeleme problemi için yapılan çalışmalarda sadece işlemci tahsis prosedürü ya da çizelgeleme prosedürü için geliştirilmiş açgözlü veya meta-sezgisel algoritmalar bulunmaktadır [7], [8], [14].

2.1.1.1. CPA

CPA statik çizelgeleme problemi için homojen işlemci kümesi üzerinde çizelgeleme çözümü oluşturan bir algoritmadır [2]. Veri ve işlem paralel işlemler yönlü döngüsüz çizge ile modellenir. CPA algoritmasında çizelgeleme süreci iki aşamada gerçekleşir. İlk aşama da veri paralel işlemlerin yürütüleceği işlemci sayısını belirlemede işlemci tahsisi prosedürü yürütülür. İşlemci tahsisi prosedüründe iki alt sınır etkilidir. Sınır değişkenlerinden T_{CP} 'yi çizelgeleme uzunluğu belirler. Veri paralel işlemlerin işlemci sayısının artırılması ile bu alt sınır azalır. Diğer bir alt sınır ise T_A 'dır. T_A değeri işlemci kullanım ölçüsüdür ve işlemlere atanan işlemci sayısı arttıkça bu değer azalmaktadır. Algoritmada bu iki alt sınırın maksimumunun minimize edilmesi hedeflenir. Şekil 2.1'de CPA işlemci tahsisi sözde kodu verilmiştir. Başlangıç durumunda işlemlere birer işlemci tahsis edilir. Tüm işlemlerin birer işlemci ile başlaması T_A değerinin minimum değer olmasını sağlar. Eğer $T_{CP} < T_A$ ise işlemlere tahsis edilen işlemci sayısı artırılmaya devam eder. T_{CP} değerinin azalması kritik

yoldaki işlemlerin işlemci sayısının artırılması ile sağlanır. Her bir iterasyonda işlemci sayısının arttırılmasından en çok etkilenen işlemin işlemci sayısı artar. $T_{CP} > T_A$ olduğunda algoritma sona erer. T_{CP} ve T_A değerinin maksimumunu minimize eden işlemci tahsisine ulaşılmıştır.

CPA işlemci tahsisi
1. İşlemler kümesindeki her bir t işlemi için: 2. t işlemine 1 işlemci tahsis et 3. while ($T_{CP} > T_A$) 4. Eğer t işlemi kritik yolda ve t işlemine tahsis edilen işlemci sayısı sistemin işlemci sayısından küçük ve ((t işlemine tahsis edilen işlemci ile çalışma süresi) – (t işlemine tahsis edilmiş işlemci sayısının bir fazlası ile çalışma süresi)) maksimum ise 5. t işlemine tahsis edilmiş işlemci sayısının 1 artır 6. T_{CP} ve T_A güncelle 7. bitir

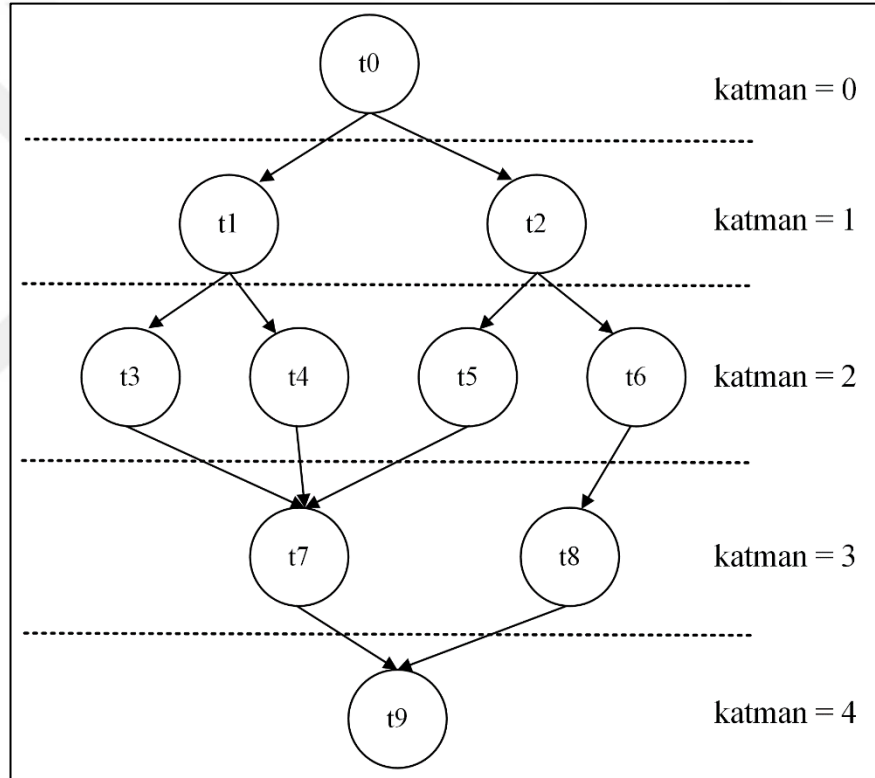
Şekil 2.1: CPA işlemci tahsisi sözde kodu.

İkinci aşamada liste çizelgeleme prosedürü yürütülür. Her iterasyonda hazır olan işlemlerden en yüksek öncelik değerine sahip olan işlem seçilir. Bir işlemin öncelik değeri yönlü döngüsüz çizgede çıkış düğümüne olan yol uzaklığıdır. Seçilen işlem tahsis edilmiş işlemci sayısı kadar uygun sayıda olan işlemcilere atanır. Tüm işlemler işlemcilere atanana kadar iterasyonlar devam eder. Farklı çizelgeleme prosedürlerinde işlemlerin öncelik belirleme prosedürleri değişebilir. Örneğin TSAS algoritmasında işlemin en erken başlama zamanına göre öncelik değeri belirlenir [15].

2.1.1.2. MCPA

CPA algoritması veri paralelliğinden yararlanarak mevcut tüm işlemciyi belirli bir göreve tahsis edebilir. CPA algoritmasının işlemci tahsisi aşamasında veri paralelliğine çok fazla odaklanması görev paralelliğinden faydalanabilme olasılığını bir kenara atar. Bu durum MCPA algoritmasının geliştirilmesini sağlamıştır [3]. MCPA algoritmasında CPA algoritması gibi işlemci tahsisi ve çizelgeleme iki

aşamada gerçekleşir. İlk aşama olan işlemci tahsisinde CPA algoritmasına benzerdir. CPA algoritmasından farklı olarak fazladan bir koşul daha ekleyerek işleme işlemci tahsis edilebilmesi için izin verir. Bu koşul ise işlemin bulunduğu katmandaki işlemlere tahsis edilen toplam işlemci sayısının sistemdeki toplam işlemci sayısından küçük olmasıdır. Şekil 2.2’de yönlü döngüsüz çizgede oluşan katmanlar gösterilmiştir. Şekil 2.3’de MCPA işlemci tahsisi sözde kodu verilmiştir. Algoritmada CPA algoritmasından farklı olarak t işleminin bulunduğu katmandaki işlemlere tahsis edilmiş toplam işlemci sayısı bulunmuştur. Tahsis edilen işlemci sayısının artırılması için katmandaki toplam işlemci sayısının sistemin işlemci sayısından küçük olması gerekir.



Şekil 2.2: MCPA DAG.

MCPA işlemci tahsisi
1. İşlemler kümesindeki her bir t işlemi için: 2. t işlemine 1 işlemci tahsis et 3. while ($T_{CP} > T_A$) 4. $N_i \leftarrow t$ işleminin bulunduğu katmandaki işlemlere tahsis edilmiş toplam işlemci sayısı 5. Eğer t işlemi kritik yolda ve N_i sistemin işlemci sayısından küçük ve ((t işlemine tahsis edilen işlemci ile çalışma süresi) – (t işlemine tahsis edilmiş işlemci sayısının bir fazlası ile çalışma süresi)) maksimum ise 6. t işlemine tahsis edilmiş işlemci sayısının 1 artır 7. T_{CP} ve T_A güncelle 8. bitir

Şekil 2.3: MCPA işlemci tahsisi sözde kodu.

MCPA algoritmasının çizelgeleme aşaması CPA algoritması ile aynıdır. Liste çizelgeleme prosedürü yürütülür.

2.1.1.3. MCPA2

MCPA2 algoritması da iki aşamadan oluşur [4]. İlk aşama işlemci tahsisi sürecinde, MCPA işlemci tahsisi sürecindeki bir katmandaki işlemlere tahsis edilen işlemci sayısının sistemdeki toplam işlemci sayısını aşmaması gerektiği kısıtı değerlendirilir. Eğer katman daha fazla işlemci tahsisine uygunsa katmandaki işlemlerin toplam işlemci sayısı, sistemdeki toplam işlemci sayısının iki katına kadar artırılabilir.

MCPA2 algoritmasında ikinci aşamada CPA algoritmasına benzer liste çizelgeleme prosedürü uygulanır. CPA algoritmasından farklı olarak işlemci kullanımında faydayı artırmak için paketleme ve geri doldurma yöntemleri uygulanır. Paketleme yöntemi bir işlemin yürütülmeye hazır olduğu (öncül işlemlerinin tamamlandığı) anda yeterli sayıda boşta işlemcinin olmaması durumunda gerçekleştirilir. İşleme tahsis edilecek işlemci sayısı azaltılarak daha erken zamanda işlemin yürütülmesi sağlanır. Ancak işlemin tahmini bitiş zamanı işlemci sayısı azaltılmadan önceki tahmini bitiş zamanından küçük olmalıdır. Uygulanan geri

doldurma yönteminde ise yüksek öncelikli bir işlem yeterli sayıda boşta işlemci bulamadığında yeterli işlemci sayısına ulaşılan kadar bekler. Bu durum çizelgelemede işlemcilerin boş kaldığı zamanların oluşmasına neden olur. İşlemci kaynakların daha verimli kullanılması için bu boşluk zamanlarda yürütülmeye uygun olan daha düşük öncelikli işlemler bu işlemcilere atanır.

2.1.1.4. IAES

IAES algoritması işlemci tahsisi ve çizelgeleme prosedürlerini iki aşamada değil tek aşamada yürütür [5]. Algoritmada başlangıçta her bir işleme birer işlemci tahsis edilir. Tahsis edilen işlemci sayısına göre liste çizelgeleme prosedüründe olduğu gibi işlemlerin önceliklerine göre işlemcilere atanır. Genel yürütme süresi hesaplanır. Sonrasında her iterasyonda işlemlere tahsis edilen işlemci sayısı artırılarak ya da azaltılarak genel yürütme süresi hesaplanır. İşlemlere atanan işlemci sayısı statik kritik yola göre değil tahsis edilen kritik yola göre belirlenir. İşlemci sayısının azaltılması aşamasında ise öncelikle işlemci sayısı artırılan işlemin başlangıç ve bitiş zaman aralığı periyodunda bulunan işlemler bulunur. Birden fazla işlemcisi bulunan işlemler arasında işlemci sayısının işlemin çalışma süresine en az etkisi olan işlem seçilir. Seçilen işlemin işlemci sayısı bir azaltılır. Genel yürütme süresi değişmeden kalana kadar iterasyonlar devam eder.

2.1.1.5. Diğer Çalışmalar

CPR algoritması [6] işlemci tahsisi ve çizelgeleme prosedürlerini tek aşamada yürüten algoritmalarından biridir. Bir iterasyonda belirli bir işleme tahsis edilen işlemci sayısı artırılır ve liste çizelgeleme prosedürü ile genel yürütme süresi hesaplanır. Genel yürütme süresi azalıp azalmaması kontrol edilerek azaldıysa işlemci tahsisi gerçekleşir. İşlemci sayısının artması ile genel yürütme süresini azaltacak bir işlem bulunmadığında iterasyon sona erer.

Tek aşamada yürütülen diğer bir algoritma LOC-MPS' dir [16]. Bu algoritmada da kritik yol üzerindeki işlemler seçilir. İşlemci sayısının artırılmasında sadece çalışma süresinde sağlanan fayda yeterli değildir. Çünkü bir işleme atanan işlemci sayısının artırılması eşzamanlı çalışabileceği işlemleri azaltmaktadır. Bu yüzden algoritmada

işlemlerin eşzamanlılık oranı da işlemci tahsisini etkiler. Algoritma genel yürütme süresini artıracak işlemci tahsisine izin verir. Böylece işlem paralelliğinden veri paralelliğine dönüşümü sağlanır. Geri doldurma tekniğiyle de düşük öncelikli işlemlerin yüksek öncelikli işlemleri geciktirmeden işlemcilerin kullanılmasını sağlar.

Hesaplama kaynağı heterojen ve çoklu kümeleme yapısında olduğu durumda çizelgeleme problemi için önerilen çizelgeleme algoritmalarından biri MTD'dir [13]. AEST, bir işlemin uygun heterojen platformda en erken başlama zamanıdır. ALST, işleminin uygun heterojen platformda en geç başlama zamanıdır. İşlem seçimi için bu iki değerin farkı alınır. Minimum olması durumuna göre seçilir. Çizelgeleme sürecinde kümenin durumuna bağlı olarak işlem çoğaltma ve göç gibi teknikler uygulanır.

CPA13 algoritmasında teoride ve pratikte işlem çalışma sürelerinin farklı olması durumu ele alınmıştır [17]. Pratikte farklı sebeplerden dolayı çalışma süresi farklılıklar gösterir. Bu farklılıkların giderilmesi ve hesaplamaların teorik değerlere yakınlaştırılması için çalışma süresinde artışa neden olan işlemci tahsisleri kaldırılarak çözüm bulunmuştur. MCPA algoritmasına benzer ancak çizelgeleme işlemi optimize edilmiştir. Belirli bir zamanda mevcut bir işlem için işlem daha erken başlayabilir mi, işlemin tamamlanma süresini ertelemeyebilir mi ve işlemci sayısını azaltılabilir mi soruları için ikili arama algoritması uygulanır.

İşlemci tahsis prosedürü için meta-sezgisel yöntemler de kullanılmıştır [7]. Bu çalışmada sadece işlemci tahsis prosedürü için genetik algoritma uygulanmıştır. Kromozomlar işlemlere tahsis edilen işlemci kümesinden oluşur. Başlangıç çözümleri elde etmek için MCPA algoritması kullanılır. Yeni kromozomların oluşturulması için sadece mutasyon uygulanır.

İşlemci tahsis prosedürü haricinde sadece çizelgeleme prosedürü öneren çalışmalar da bulunmaktadır. Liste çizelgeleme de işlem önceliğini belirleyen üç kriter belirlenmiştir [14]. Bu kriterler:

- P: Eğer bir işlemin veri paralelliği diğer işleme göre daha fazla ise bu işlem daha yüksek önceliklidir.
- C: iki işlem aynı veri paralelliğine sahipse kritik yol değerlerine bakılır.
- S: iki işlem aynı veri paralelliğe ve kritik yol değerine sahipse ardıl işlem sayısına göre öncelik atanır.

Öncelik belirleme kriterine göre değişen sıralarla 6 farklı algoritma oluşturulur. Algoritma ismindeki harf sıralaması öncelik belirlenmesinde uygulanan sırayı göstermektedir. Bu algoritmalar PCS, CPS, CSP, SCP, PSC, SPC'dir.

Çizelgeleme prosedürü için meta-sezgisel yöntemlerle de çözüm geliştirilmiştir [8]. İşlemci tahsis sayılarının belli olduğu varsayılır. Problem çözümünde genetik algoritma kullanılır. Kromozomlar problem için potansiyel problem çözümünü gösterir. Kromozomda bir gen bir işlemi ifade eder. Kromozomun geçerli olması için işlemlerin veri bağımlılığı kısıtlarını sağlıyor olması gerekir. Uygunluk fonksiyonu kromozomun kalitesini gösterir. Çalışmada uygunluk fonksiyonu kromozom için en uzun çizelgeleme uzunluğudur. Klasik bir seçim tekniği olan rulet kullanır. Çaprazlama ve mutasyon uygulanır. Mutasyonda kromozomdan rastgele bir gen seçilir ve öncül ve ardıl işlemleri arasında rastgele yerleştirilir.

Veri paralel işlem içermeyen yönlü döngüsüz çizge ile modellenmiş statik çizelgeleme problemleri için meta-sezgisel yöntemler geliştirilmiştir. Karınca sürüsü optimizasyonu da bu meta-sezgisel yöntemlerden biridir [18]. Karınca sürüsü optimizasyon yönteminde başlangıçta feromon değerleri aynıdır. $N \times N$ bir matriste gösterilir. N işlem sayısıdır. N_i işleminden sonra seçilecek işleme olasılıksal karar değerine bağlı olarak feromon miktarına ve işlem önceliği sezgiseline göre karar verilir. İşlemlerin birbirine veri bağımlılıklarının bulunmasından dolayı algoritmada yürütülmeye hazır olan işlemler listesi oluşturulur. Başlangıçta liste boştur. Karınca sürü optimizasyonu algoritmasında her bir iterasyonda bu liste güncellenir. Seçilen işlemler karınca listesine eklenir. Hazır listesinden çıkarılır. Tüm işlemler tamamlandıktan sonra karınca listesindeki işlemlere en erken başlama zamanına göre uygun olan işlemci atanır. En iyi sonucu veren karıncanın çözümü için feromon eklemesi yapılır. Son adımda yerel minimumdan kaçmak için buharlaştırma işlemi yapılır. Buharlaştırma oranına deneysel olarak karar verilir. Diğer bir meta-sezgisel yöntem öneren çalışmada ise genetik algoritma kullanılır [12]. Genetik algoritmada kromozomda işlem ve işlemci bilgisi yer alır Turnuva seçimi, çaprazlama, mutasyon uygulanır. İşlemci seçimi rastgele gerçekleştirilmez. Eğer işlem kritik yol üzerinde ise önceki kritik yol üzerindeki işlem ile aynı işlemci tahsis edilir. Eğer kritik yol üzerinde değil ise iletişim maliyetinin yüksek olduğu işleme atanan işlemci tahsis edilir. Eğer birden fazla iletişim maliyeti olan işlem var ise bu işlemlerin atandığı işlemciler arasından rastgele bir işlemci seçilir. Popülasyona rastgele oluşturulan birkaç üyede eklenerek erken yakınsamadan kaçınılır.

2.1.2. Dinamik Çizelgeleme

Dinamik çizelgelemede heterojen işlemci kaynaklarında yapılan çalışmalar ön plana çıkmaktadır. Veri paralel işlemler içeren iş akışları için literatürdeki çalışma sayısı oldukça kısıtlıdır. Yapılan çalışmalarda veri paralel işlemler katı işlemlerdir. Katı işlemlere tahsis edilecek işlemci sayısı çizelgelemeden önce belirlenmektedir. Dinamik çizelgelemede veri paralel işlem içermeyen iş akışları için açgözlü algoritmalar ve meta-sezgisel algoritmalarda bulunmaktadır.

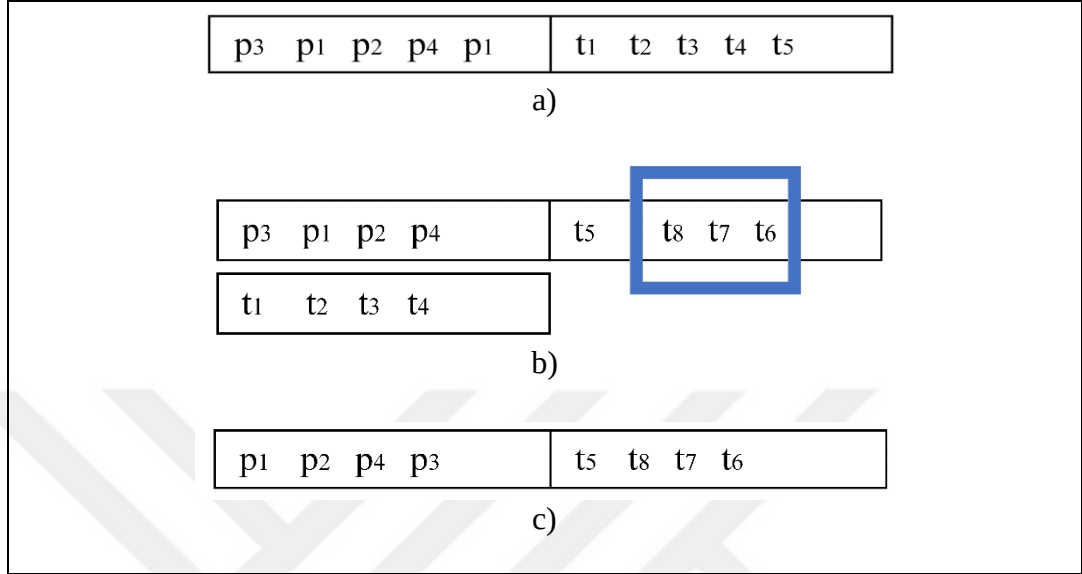
2.1.2.1. GA-ECT

Veri paralel işlem içermeyen yönlü döngüsüz çizgeler ile problem modellenmiştir [10]. Heterojen işlemci kaynağı üzerinde yürütülür. İşlemci kaynağı farklı hızlara sahip olan işlemcilerden oluşur. Bu yüzden aynı işlem farklı çalışma sürelerinde yürütülebilir. Her bir işlemin son teslim tarihi vardır. İşlemlerin sisteme varış zamanı bilinmez. Sistemde çizelge bilgisini tutan merkezi çizelgeleme birimi belirlenmiştir. Yeni iş geldiği zaman merkezi çizelgeleme birimi, henüz başlamamış olan işlemler ile yeni işlemler için çizelgeleme işlemini gerçekleştirilir. Yürütülmesi başlamış olan işlemler ise yeni çizelgelemeye dahil edilmezler. Yeni çizelgeleme sürecinde tüm problem kısıtları değerlendirilmelidir.

Merkezi çizelgeleme birimi, işlemcilerin durum bilgisini tutan yapıyı günceller. Biten işlemleri tutan kuyruk yapısını da bitmiş işlem bilgileri ile günceller. Merkezi işlemci birimi ayrıca sıralanmış işlem kuyruğunu da beslemektedir. Bu kuyrukta her bir işlemin son tamamlanma zamanı, çalışma süresi, iletişim maliyeti, işlem öncelik kısıtları tutulur. Kuyruktaki işlem bağlı bir işlem ise merkezi çizelgeleme birimi öncül işlemlerin tamamlanma durumunu kontrol eder. Eğer tamamlanmamışsa sıralanmış işlem kuyruğun da işlem beklemeye devam eder.

Çizelgeleme için meta-sezgisel yöntem olan genetik algoritma ile çözüm geliştirilir. Genetik algoritmada başlangıç popülasyonları rastgele belirlenir. Kromozomda işlem ve işlemci bilgileri yer alır. Çizelgelemede yeni gelen işlemler ile kromozomun uzunluğu değişebilir. Şekil 2.4 a)' da bir kromozom örneği verilmiştir. p_3, p_1, p_2, p_4, p_1 işlemcileri ifade ederken t_1, t_2, t_3, t_4, t_5 işlemleri ifade eder. Şekil 2.4 b)' de t_5 işlemi henüz yürütülmeye başlamamıştır. Bu yüzden yeni kromozomda

da yer alır. Diğer işlemlerin atandığı işlemciler gösterilmektedir. Şekil 2.4 c)' de çizelgeleme yeni işlemler ile yeni kromozom oluşmuştur. Algoritmada rulet seçimi, mutasyon ve çaprazlama uygulanmıştır.



Şekil 2.4: a) Kromozom, b) Yeni gelen işlemler ile henüz yürütülmeye başlamamış işlemler ve yürütülmeye başlamış işlemlerin atandığı işlemciler, c) Yeni kromozom.

2.1.2.2. HGA

Bulut ortamında birden fazla iş akışının çizelgelenmesi problemi üzerine çalışır [19]. Veri paralel işlem içermeyen iş akışları yönlü döngüsüz çizge ile modellenir. Her T anında yeni gelen işlemlerle beraber henüz bitmemiş işlemler tekrar çizelgeleme algoritmasına eklenir. Önerilen genetik algortmada kromozomlar çeşitli sezgisel kurallardan oluşur. Önerilen sezgisel kurallar;

- MINSK: Her bir işlemin bekleme zamanını hesaplar. Bu hesaplama verinin transfer süresini de ekler. Öncelik sıralaması, bekleme süresinin küçükten büyüğe sıralanması ile belirlenir.
- MIN LFT: En erken biten işlemlere öncelik verilir.
- MIN SPT: Çalışma süresi ve veri transfer süresi en küçük olan işlemlere öncelik verilir.
- MIN LST: Uzun süre ertelenebilecek işlemlere küçük bir öncelik verilecektir.

- MIN EST: En erken başlama stratejisine dayanır. İlk başlayan işlemlere öncelik verir.

Kromozomlar işlem sayısı kadar genden oluşur. Her bir gen MINSK, MIN LFT, MIN SPT, MIN LST, MIN EST sezgisellerinden birini ifade eder. Çaprazlamada iki noktalı çaprazlama yöntemi kullanılır. Mutasyonda ise rastgele belirlenen bir gen değiştirilir. Uygunluk fonksiyonu çizelgelemenin genel yürütme süresidir.

2.1.2.3. MOWS

Veri paralel işlemler içeren iş akışları ile çalışılır [9]. Veri paralel işlemler katı işlemlerdir. Katı işlemlerde çizelgeleme öncesinde işlemlere atanan işlemci sayısı belirlenir. İş akışları yönlü döngüsüz çizge (DAG) ile temsil edilir. İş akışlarının varış zamanı bilinmemektedir. Önerilen yöntemde her bir hazır işlem kuyruğa alınır. Yeni bir iş akışı geldiğinde kalan tüm iş akışlarının çalışma süresini hesaplanır. Bekleme kuyruğundaki işlemlerin kalan çalışma sürelerine göre küçükten büyüğe sıralanır. Aynı iş akışından gelen işlemler sıralama değerine göre büyükten küçüğe sıralanır. Geri doldurma tekniği uygulanır.

2.1.2.4. Diğer Çalışmalar

Dinamik çizelgeleme de veri paralel işlemler içeren ancak işlemler arasında veri bağımlılığı olmayan problemlere açgözlü algoritma ile çözüm geliştirilmiştir [20]. Bu çalışmada işlemlerin varış zamanları bilinmemektedir. Veri paralel işlemler kalıplanabilir işlemlerdir. Problem işlem öncelik sıralaması ve işlemci tahsisi için iki alt prosedüre ayrılmıştır. İşlemin öncelik sıralamasının belirlenmesinde FCFS, SJF ve OvH kullanılır. FCFS, işlemin geliş zamanına göre öncelik sırası tahsis eder. SJF, en küçük yürütme süresine sahip olan işlem önceliklidir. OvH da ise işlemlerin paralellik ek yüküne göre öncelik verilir. İşleme atanan işlemci sayısına ise Single, Max, Fit, Fair ile karar verilir. Single’da her işleme tek işlemci tahsis edilir. Max’da her işleme maksimum sayıda işlemci tahsis edilir. Fit’de işlem seçimi öncelikli işleme maksimum işlemci tahsis edilir. Fair’de her işleme adil sayıda işlemci tahsis edilir.

Heterojen kaynaklar üzerinde yönlü döngüsüz çizge (DAG) ile temsil edilen veri bağımlılıkları olan veri paralel işlemlerin çizelgelenmesi problemi üzerine çalışmalarda bulunmaktadır [21]. Bu çalışmanın temel amacı, işlemleri öncül işlemlere atanan işlemcilerle eşlemeye çalışarak iletişim ve yeniden dağıtım maliyetlerini azaltmaktır. Algoritma da öncelikle hazır işlemler hesaplama maliyetine ve çizgedeki derin uzunluğuna göre sıralanır. Hazır listesindeki tüm işlemler işlemcilerle atanana kadar iterasyon devam eder. Diğer bir çalışmada ise hesaplama kaynağı birden fazla kümeden oluşur [22]. Algoritma birinci adımda işlem yürütme süresinden ve tüm yeniden veri dağıtım (iletişim) maliyetlerinden oluşan en küçük toplam süreye sahip kümeyi seçer. En çok derinliğe sahip olan iş akışları en erken bitebileceği kümeye atanır. Ayrıca algoritmada işlemlerin erteleme zamanı da dikkate alınır. Heterojen kaynak kullanılan diğer bir çalışmada ise bir iş akışı tamamlandığında, boş işlemciler varsa bekleyen iş akışlarına ve ayrıca halihazırda yürütülmekte olan iş akışlarına dağıtılır [23]. Yeni gelen bir iş akışı, yürütmeye başlamak için yeterli işlemciye sahip olmadığında birkaç işlemcinin iş akışı yürütmesini devre dışı bırakılır ve yeni iş akışlarına tahsis edilir. Çizelgelemede gelen iş akışı için minimum, optimal ve maksimum işlemci sayısı, gerçek tahsis edilen işlemciler listesi, iş akışının başlangıç zamanı ve iş akışının beklenen bitiş zamanı gibi bilgileri tutulur. Çizelgeleme aşamasını iki adımda yürüten çalışma da bulunmaktadır [24]. Bu çalışmada da heterojen kaynakta birden fazla DAG için dinamik çizelgeleme problemi ele alınır. Veri paralel işlemler içermeyen tek iş akışı için HEFT [25] algoritması geliştirilerek P-HEFT [26] algoritması oluşturulmuştur. HEFT algoritması ise veri paralel olmayan DAG ile modellenen iş akışının heterojen kaynak üzerinde statik çizelgeleme problemi için önerilmiştir. Çoklu iş akışları için ise HPTS algoritması önerilmiştir.

Dinamik çizelgeleme problemi için meta-sezgisel yöntemler ile de çalışmalar yapılmıştır. Veri paralel işlemler içermeyen DAG ile işlemlerin modellendiği çalışmada genetik algoritma ile çözüm geliştirilmiştir [27]. Yeni iş akışlarının gelmesi durumunda çizelgeleme işlemi tekrardan gerçekleştirilir. Başlangıç popülasyonu rastgele belirlenir. Kromozomdaki her bir gen işlemler ve işlemlerin atandığı işlemcileri içerir. Uygunluk fonksiyonu bulut maliyetine ve işlemlerin bitiş zamanına bağlı olarak belirlenir. Çaprazlama işlemi tek noktalı, iki noktalı ve tekdüze olarak gerçekleştirilebilir. Mutasyon işlemi kromozomda rastgele belirlenen iki gen arasındaki işlemlerin yer değiştirmesi ile gerçekleşir. Genetik algoritma ile çözüm geliştirilen farklı bir çalışmada ise algoritmada bir gen, çizelgeleme ve haritalama

olmak üzere iki bilgiyi içerir [28]. Çizelgeleme, işlemlerin yürütülme sıralamasıdır. Haritalama ise işlemlerin çalıştırılabileceği işlemcileri gösterir. Seçim için turnuva seçimi kullanılır. 0 ile 1 arasında rastgele bir değerle çizelgeleme işlemine karar verilir. Rastgele belirlenen bir değer 0,5'ten küçükse haritalama kısmında değilse çizelgeleme kısmında çizelgeleme yapılır. Mutasyonda ise sadece haritalama kısmında gerçekleştirilir. Meta-sezgisel yöntemlerden karınca sürüsü optimizasyonu ile çözüm geliştirilen çalışmada bulunmaktadır [11]. Dinamik çizelgeleme, işlemlerin iletişim sürelerinin, yürütme süresinin ve öncelik kısıtlarının çalışma anında değişmesi olarak tanımlanmıştır. Heterojen sistemde her bir işlem bir karıncadır. Feromon miktarı ise işlemci yüküdür. Başlangıç feromon değeri 0 atanır. Böylece ilk aramada bir karınca bir noktayı sabitleyecektir. Sonraki karıncalar aynı noktayı ve en iyi işlemciyi takip edecektir. Algoritma da bitiş süresinden önce bir işlem yüksek hızlı bir işlemcide tamamlanırsa çizelgeleme prosedürü bekleyen işleme aynı işlemciyi tahsis eder ve feromon güncellenir. İşlemci seçimi işlemin en erken başlama zamanı ve en erken bitiş zamanına dayanır. Algoritma iki adıma yürütülür. İlk adımda sıralama metodu kullanılarak DAG'daki işlemlerin öncelikleri belirlenir. İkinci adımda ACO metodu kullanılarak işlemler işlemcilere atanır.

Bulut ortamında bağımsız işlemlerin dinamik çizelgelenmesi problem üzerine meta-sezgisel yöntemlerle çözüm geliştirilmiş çalışmalarda bulunmaktadır [29]. Tavuk sürü optimizasyon algoritması (CSO) ve erken yakınsamayı önleyen dinamik kuzgun optimizasyon (IRRO) algoritması beraber kullanılmıştır.

3. YENİ NESİL DİZİLEME

Yeni nesil dizileme milyonlarca parçaya ayrılmış DNA molekülünün her bir parçasının düşük maliyetle ve yüksek güvenilirlikle paralel olarak ölçülmesini sağlar. Günümüzde yeni nesil dizileme yöntemi, hastalığa sebep olan mikroorganizmaların genomunun dizileme ve bitki genomunun dizileme gibi birçok amaç için kullanılır [30], [31]. Yeni nesil dizileme analizi, kanser hastalığının erken tanısı, bireysel tedavi oluşturulması ve hastaların tedaviye yanıtının izlenmesi gibi süreçlerin yürütülmesine, mutasyonların tespiti ile nadir hastalıkların tanısına, gen keşfine, atasal DNA keşfine de olanak sağlar.

Islak laboratuvar (kimyasal reaksiyonlar) adımı kuru laboratuvar adımı olarak tanımlanan yeni nesil dizileme verisi analiz süreci için ham veri oluşturulur. Islak laboratuvar çalışmasında biyolojik materyalin elde edilmesi, elde edilen biyolojik materyallerden genomik DNA'nın izolasyonu, izole edilen DNA'daki hedef bölgelerin seçilmesi, DNA'nın enzimatik reaksiyonla kesilerek DNA kütüphanesinin oluşturulması, kütüphaneyi oluşturan DNA parçalarının çoğaltılması, DNA parçalarının dizileme, dizileme sonrası ham verinin oluşturulması süreçleri gerçekleşir. DNA parçalarının porlardan geçerken iletken ortamda yarattığı pikoamper (pA) boyutundaki elektriksel değişiklikler por etrafında bulunan sensörler tarafından okunup eş zamanlı olarak kaydedilir. Dizileme cihazlarından elde edilen ham verilerde ph veya elektrik akımı değişimi gibi verilerdir.

Yeni nesil dizilme, pirodizileme sisteminin geliştirilmesi ile başlamış sentez ile dizileme, ligasyon ile dizileme, iyon ile dizileme sistemleri ile devam etmiştir [32].

Baz çağırma aşamasında ham verilerin nükleotit bilgilerine çevrilmesi gerekmektedir. Baz çağırma aşamasının ardından elde edilen veri formatı FASTQ olacaktır. [33] FASTQ dosyasında nükleotit dizileri ve her bir nükleotit dizisini kalite değeri yer alır. Kalite değeri ASCII formatında gösterilir.

Yeni nesil dizilemede baz çağırma işleminin doğru olması biyoinformatik analiz sürecini kritik oranda etkiler. Yapılan hatalar hizalama ve varyant çağırmayı ciddi oranda etkileyeceği için öncelikle kalite kontrolü yapılır. Düşük kaliteye sahip olan okumalar kırpılır. Adaptör dizileri de hizalama aşamasında yanlış sonuçların elde edilmesi veya varyant çağırmada varyant tespiti hatalarına sebep olması gibi nedenlerle kırılabilir. FASTQC aracı kırpma işlemi için kullanılabilir.

DNA yeni nesil dizileme analizinde bir sonraki aşama hizalamadır. Hizalama, kısa nükleotit okumalarının bir referans genomla haritalanması işlemidir. Milyonlarca kısa okumanın her birinin insan genomundaki 3 milyar olası konumla karşılaştırılması gerektiğinden, bu hesaplama adımı oldukça karmaşıktır. Dizileme de üretilen kısa okumalar tek uçlu okuma ya da çift uçlu okuma olabilir. Bu okumaların referans genom üzerinde uygun konumlara hizalanması gerekir. Milyonlarca okumayı verimli bir şekilde referans genomla hizalamak için yazılımlar geliştirilmiştir. Her bir okumanın referans genomdaki başlangıcının tespit edilmesi, benzersiz eşleme yapılması, kalite değişkeni gibi etkenlerden dolayı hizalama yazılımları hesaplama açısından zaman alıcıdır. Dizilerin referans genomdan eksik olması durumunda hizalanamayan ya da yanlış hizalanmış okumalara sebep olur. Hizalama da kalite skoru bir okumanın yanlış eşleştirilmiş olma olasılığını tahmin eder. Bir okuma eğer referans genomda birden fazla lokasyona eşleniyorsa bazı yazılımlar tespit edilen tüm lokasyonlara yerleştirirken bazıları ise rastgele yerleştirir. Bu gibi sebepler dolayı yazılımların kalite skorları farklılık gösterir.

Günümüzde en yaygın kullanılan hizalama aracı Burrows-Wheeler Aligner Wheeler (BWA)'dır [34]. BWA, Burrows-Wheeler Transform (BWT) sıkıştırması ve Smith-Waterman algoritmasını kullanır. Burrows-Wheeler Transform tabanlı yöntemler referans genomla ait indeks oluşturarak dizi eşleme yaklaşımı kullanır. BWT tabanlı yöntemler hızlıdır. Bowtie [35] ve SOAP2 [36] BWT tabanlı diğer hizalama araçlarıdır. Sequence Alignment/Map (SAM) ve Binary Alignment/Map (BAM), okuma hizalamalarının depolanması için kullanılan standart dosya formatlarıdır [37]. SAMTools, SAM dosyasındaki bilgilerin okunabilir hale dönüşmesi sağlayarak BAM dosyasının oluşturulmasını sağlayan yazılım aracıdır. Çıktı dosyası okuma sırası, kalite puanı, hizalama konumu, referans sırasına göre farklılıkları, haritalama kalite puanlarını (MAPQ) içerir. Hizalama işleminden sonra BAM dosyasında sıralama gerçekleştirilir.

Varyant çağırma aşamasında referans genomla göre farklı olan ham verinin kalite değerlendirme aşamasından geçmiş varyantları tespit edilir [38], [39]. Tespit edilen varyantlar küçük insersiyon ve delesyonlar (INDELS), tek nükleotit varyasyonları (SNVs), veya büyük yapısal farklılıklar olabilir. Varyant çağırma işleminde gerçek varyantların tespit edilmesini hizalama ve sıralama hataları zorlaştırır. İşlemin yüksek hassasiyet ile yapılması ise hastalıklarla ilgili dizileme varyantlarını belirlemede, nadir hastalıkların varyantının tespitinde, popülasyondaki alel frekanslarını

değerlendirmede önemli bir başarı sağlar. Varyantların değerlendirilmesinde dizi verisinin referans genomu ne kadar kapladığı dikkat edilmesi gereken önemli bir konudur. SAMTools içerisinde analiz araçları bulunduran bir araç kümesidir. Varyant çağırma için de mpileup aracı kullanılır. Genome Analysis ToolKit (GATK) [40] varyant çağırmada kullanılan diğer popüler yöntemlerden biridir. GATK modüler bir şekilde tasarlanmıştır ve MapReduce fonksiyonel programlama yaklaşımına dayanmaktadır [41]. Varyant çağırma çıktıları VCF dosya formatında oluşturulur.

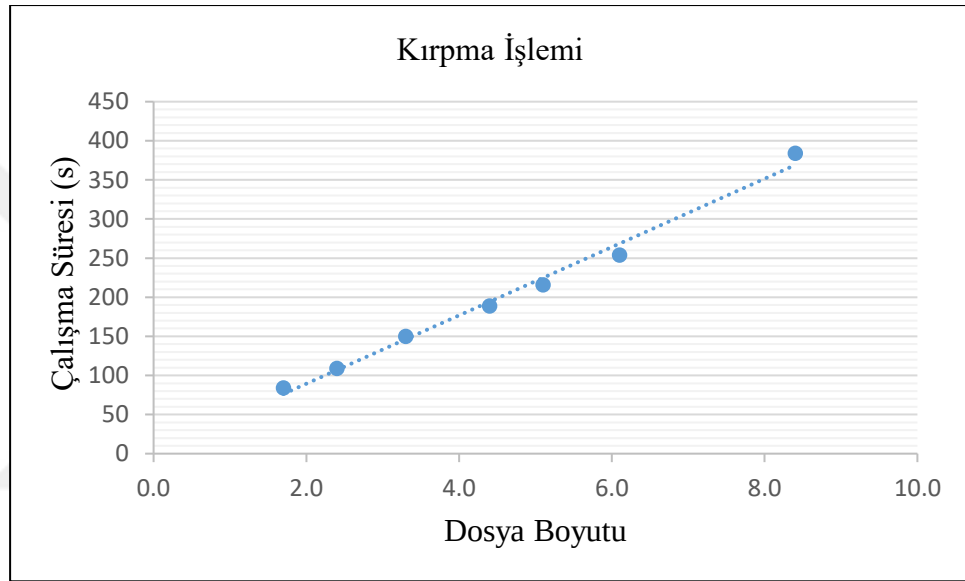
Varyant anatasyonu, varyantların işlevsel etkilerini tahmin edilmesi işlemidir. Varyant anatasyonu için kullanılan araçlar farklı metodoloji ve veritabanı kullanırlar. Araçların çoğu tek nükleotit varyasyonları (SNVs) ve delesyonları (INDELS) açıklayabilir. Yapısal varyasyon (SV) ve kopya sayısı farklılıklarının (CNV) daha karmaşık olmasından dolayı açıklama gerçekleştirilemeyebilirler. Yaygın olarak kullanılan araçlar; ANNOVAR [42], VEP [43], snpEff [44], SeattleSeq [45]'dir. ANNOVAR aracı SNP, INDEL, CNV'leri tanımlayabilir. Varyantların genlere ve diğer genomik öğelere göre etkilerini açıklar. Varyasyon veri tabanı ile karşılaştırır. VEP, ANNOVAR gibi genomik açıklama sağlar. Varyantın genomik yapıda nereye karşılık geldiğini belirler ve buna göre bir sonuç tahmininde bulunur. SNP, INDEL, CNV veya SV'leri tanımlayabilir. Bu alanda sıklıkla kullanılan veri tabanları ise Ensembl [46], RefSeq [47], UCSC' dir [48]. Her veri tabanı farklı özelliklere sahiptir. Açıklamalar veri tabanlarına göre farklılık gösterebilir. Varyantları anlamlandırmak için bazı filtreleme stratejilerine gerek duyulmaktadır. Yeni nesil dizilemede yaygın kullanılan filtrelerden biri popülasyon frekansı filtersidir. Bu alanda sıklıkla kullanılan veri tabanları ExAC [49], gnomAD'dır [50].

3.1. Yeni Nesil Dizilemede Çalışma Zamanı Modeli

Yeni nesil dizileme analizinde FASTQ dosyası sisteme vardığında çizelgeleme gerçekleştirilebilmesi için iş akışındaki işlemlerin çalışma süresinin bilinmesi gerekir. İş akışındaki işlemlerin çalışma süresi hesaplaması için FASTQ dosyasının büyüklük farklılıklarından faydalanılır. Bu tez çalışmasında dosya büyüklüğüne göre çalışma zamanı modellenmesi gerçekleştirilmiştir. İş akışlarının yürütülmesi için kullanılan sistem 8 fiziksel, 8 sanal çekirdekli i7 4.6 Ghz işlemci, 16 GB ana bellek, 1 TB saklama alanına ve Ubuntu 22.04 işletim sistemine sahiptir. 1,7, 2,4, 3,3, 4,4, 5,1, 6,1, 8,4 GB

büyükte FASTQ dosya formatındaki dizi okuma dosyaları kullanılmıştır [51]. Her dosya için tüm işlemler birden fazla kez çalıştırılmıştır. Çalışma süresi tahminlemesi için her dosyanın ilgili işlemdeki çalışma sürelerinin ortalaması alınmıştır.

Yeni nesil dizilemede biyoinformatik iş akışındaki ilk işlem kırpma işlemidir. Bu işlem için FASTQC kullanılmıştır. Girdi dosyası FASTQ dosyasıdır. Şekil 3.1’ de yürütülen farklı boyuttaki girdi dosyalarının çalışma süresi ile doğrusal modellemesi görülmektedir. Eşitlik (3.1)’de kırpma işlemi için çalışma süresi tahminlemede doğrusal modelin denklemi verilmiştir.



Şekil 3.1: Kırpma işleminin çalışma süresi tahmini için doğrusal model.

$$y = 43.68 x + 2.046 \quad (3.1)$$

Biyoinformatik iş akışındaki ikinci işlem olan hizalama işlemi veri paraleldir. Veri paralel işlemler birden fazla işlemci üzerinde paralel yürütülebilir. Paralel çalışma süresi tahmini için sıklıkla Amdahl yasası kullanılır [3]. Amdahl yasasına göre çok sayıda işlemci üzerinde veri paralel işlem, verinin bir bölümünü paralel yürütürken diğer bölümünü ise seri olarak yürütür. Literatür çalışmalarında Yeni nesil dizilme analizi gibi gerçek dünya uygulamaları için çalışma zamanı ve paralel performans tahmin modeli öneren çalışma bulunmaktadır [52]. Çalışmada gerçek iş akışlarında genellikle hızlanmaların üst sınırlarını tahmin edilmesinde Amdahl Yasasını kullanıldığı belirtilir. Eşitlik (3.2)’de Amdahl Yasasının s_x hızlanma değerini

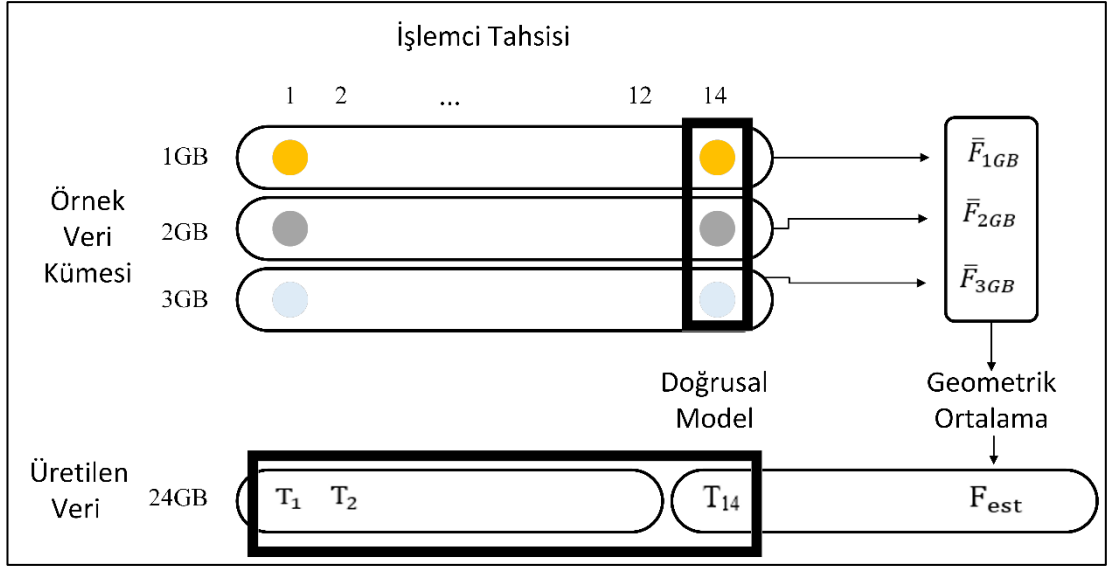
gösterirken bir işlemcide çalışma zamanı T_1 ile x işlemci üzerinde çalışma zamanı ise T_x ile gösterilir. Eşitlik (3.3)'de gösterilen $(1 - F)$, işlemin seri yürütülen parçasını ifade ederken F ise işlemin paralel yürütülen parçasını ifade eder. Amdahl yasasında işlemci sayısının artmasından paralel parçalar doğrusal faydalanırken seri parçaların faydalanmadığı varsayılır. Bu yüzden işlemci sayısı artarken çalışma zamanı doğrusal olarak azalmaz. Paralel parça F 'nin değerinin bilinmesi zordur. Bazı işlemler için iletişim gibi iş yükleri tahmini zayıflatır. Eşitlik (3.4)'de hızlanma değeri ile Karp-Flatt metriği kullanılarak F paralel değeri tahmin edilebilir [53].

$$s_x = \frac{T_1}{T_x} = \frac{T_1}{(1 - F)T_1 + T_1 F/x} \quad (3.2)$$

$$T_x = \left((1 - F) + \frac{F}{x} \right) \times T_1 \quad (3.3)$$

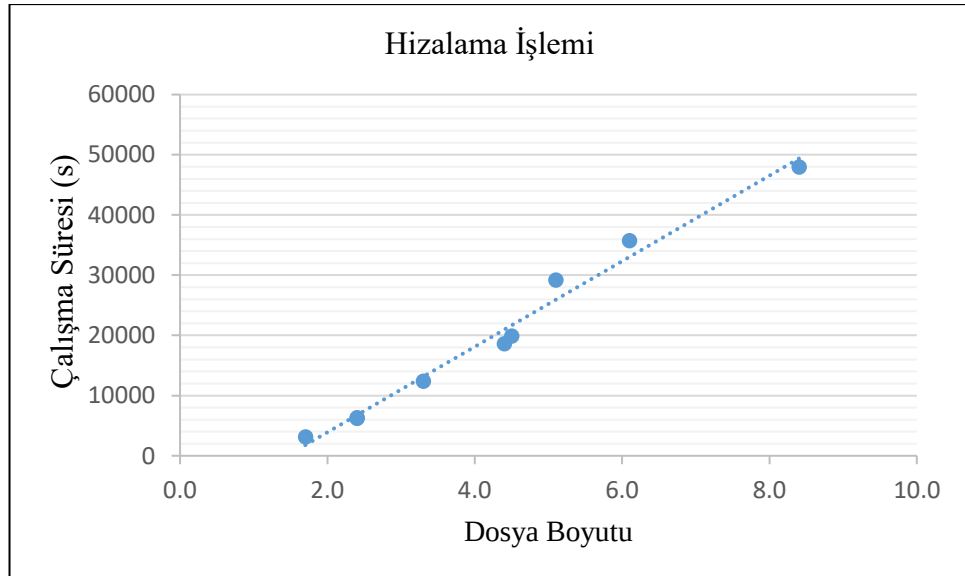
$$F = \left(1 - \frac{1}{s_x} \right) \left(1 - \frac{1}{x} \right)^{-1} \quad (3.4)$$

Girdi dosyasının işlemci sayısı sabit iken dosya boyutu ile çalışma süresi doğrusal model ile tahminlenir. İşlemci sayısının etkisini tahmin etmek için ise Karp-Flatt metriği kullanılır. Şekil 3.2' de gösterildiği üzere veri paralel işlemlerin paralellik performansının tahmini için Karp-Flatt metriği ile doğrusal model birleştirilir. Örnek verilerdeki her dosya boyutu için F paralellik değeri hesaplanarak ortalaması alınır.



Şekil 3.2: Çalışma zamanı tahmini için doğrusal model ve Karp-Flatt metriği.

Hizalama işlemi için örnek veri dosyaları ile bir işlemci üzerinde yürütülmesi sonucu oluşan doğrusal model Şekil 3.3’ de gösterilmiştir. Eşitlik (3.5)’ de hizalama işlemi için bir işlemcide çalışma süresi tahmini doğrusal modelinin denklemi verilmiştir. Hizalama işleminde BWA yazılım paketindeki BWA-MEM algoritması kullanılır. İşlemde Genom Referans Konsorsiyumu tarafından sunulan insan referans genomu kullanılır [54].



Şekil 3.3: Hizalama işleminde bir işlemcide çalışma süresi tahmini için doğrusal model.

$$y = 7106.2 x + 10318 \quad (3.5)$$

Hizalama işlemi, bir işlemci haricinde 2, 4, 6, 8, 10, 12, 14 işlemci üzerinde örnek veri dosyaları ile yürütülmüştür. Her bir dosya boyutu için farklı işlemci sayıları ile Karp-Flatt metriği kullanılarak F paralellik değeri hesaplanmıştır. F paralellik değeri hesaplanırken öncelikle Eşitlik (3.2) kullanılarak hızlanma değeri hesaplanmıştır. Eşitlik (3.4) kullanılarak da hızlanma değeri ve işlemci sayısına göre F paralellik değeri hesaplanmıştır. Tablo 3.1’de her bir dosya boyutunda işlemci sayına göre değişen F paralellik değerlerinin ortalaması verilmiştir. Tablo sonunda F_{est} değeri ile dosya boyutuna göre değişen F paralellik değerlerinin ortalamasının geometrik ortalaması verilmiştir. Bu değer işlemci tahsisinde işlemci sayısının artırılmasıyla çalışma zamanı tahmini ve işlemci sayısının artmasındaki faydanın ölçülmesinde kullanılacaktır. Tablo 3.2’de dosya boyutu ve işlemci sayısına göre gerçek çalışma süresi verilmiştir. Tablo 3.3’de ise Eşitlik (3.3)’de verilen çalışma süresi denklemine göre F_{est} değeri ile dosya boyutu ve işlemci sayısına bağlı olarak hesaplanan tahmini çalışma süreleri verilmiştir. Ölçülen değer ve tahmin edilen değerler arasındaki farkların mutlak değeri alınarak ölçülen değere göre normalize edildiğinde ortalama %5 oranında değiştiği görülmektedir. Tahmin edilen değer ölçülen değere göre yaklaşık %5 oranında daha fazla yada daha az hesaplanmıştır.

Tablo 3.1: Dosya Boyutuna göre Paralellik Değerleri.

Dosya Boyutu (GB)	$\bar{F}$
1,7	0.948284
2,4	0.961734
2,4	0.957323
3,3	0.970742
4,4	0.957524
4,5	0.95892
5,1	0.962358
6,1	0.959148
8,4	0.959295
F_{est}	0.960558

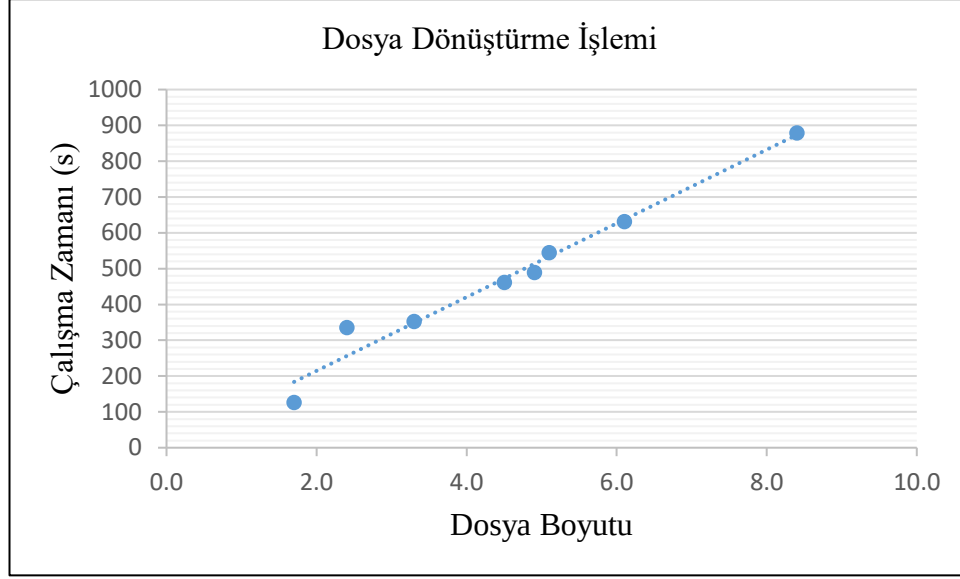
Tablo 3.2: Dosya Boyutuna (GB) ve İşlemci Sayısına göre gerçek çalışma süresi(sn).

	2	4	6	8	10	12	14
1,7	1575	844	622	516	456	403	388
2,4	3165	1657	1182	967	882	812	742
2,4	3270	1739	1199	981	892	819	774
3,3	5997	2618	1930	1727	1640	1569	1512
4,4	9529	5298	3950	2520	2399	2283	2188
4,5	9886	5638	3990	2768	2561	2393	2241
5,1	14712	7936	5630	4464	4072	3724	3499
6,1	17979	9455	6865	5431	5038	4614	4308
8,4	24131	12898	9049	7170	6896	6159	5659

Tablo 3.3: Dosya Boyutuna (GB) ve İşlemci Sayısına göre F_{est} değeri ile hesaplanan tahmini çalışma süresi (sn).

	2	4	6	8	10	12	14
1,7	1616	869	620	496	421	371	336
2,4	3269	1758	1255	1003	852	751	680
2,4	3228	1736	1239	991	842	742	671
3,3	6438	3463	2472	1976	1678	1480	1338
4,4	9354	5032	3591	2871	2439	2151	1945
4,5	10337	5561	3969	3173	2695	2377	2149
5,1	15166	8159	5823	4655	3954	3487	3153
6,1	18560	9984	7126	5696	4839	4267	3859
8,4	24908	13399	9563	7645	6494	5726	5178

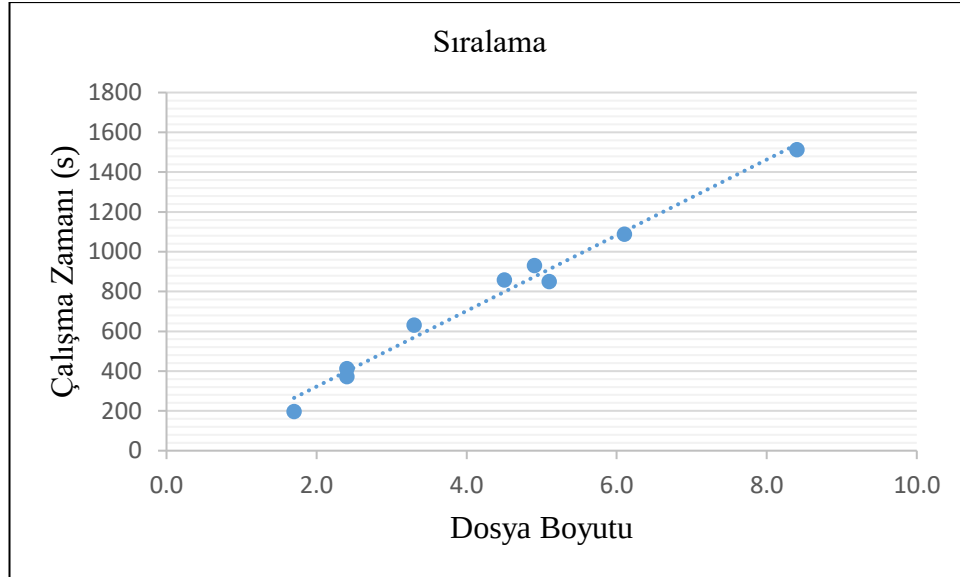
Hizalama işleminden sonraki işlem SAMTools aracı ile hizalama çıktısı olan Sam formatındaki dosyanın BAM formatına dönüştürülmesidir. Şekil 3.4’ de yürütülen farklı boyuttaki girdi dosyalarının çalışma süresi ile doğrusal modellemesi görülmektedir. Eşitlik (3.6)’de dosya dönüştürme işlemi için çalışma süresi tahminlemede doğrusal modelin denklemi verilmiştir.



Şekil 3.4: Dosya dönüştürme işleminin çalışma süresi tahmini için doğrusal model.

$$y = 102.95 x + 8.7 \quad (3.6)$$

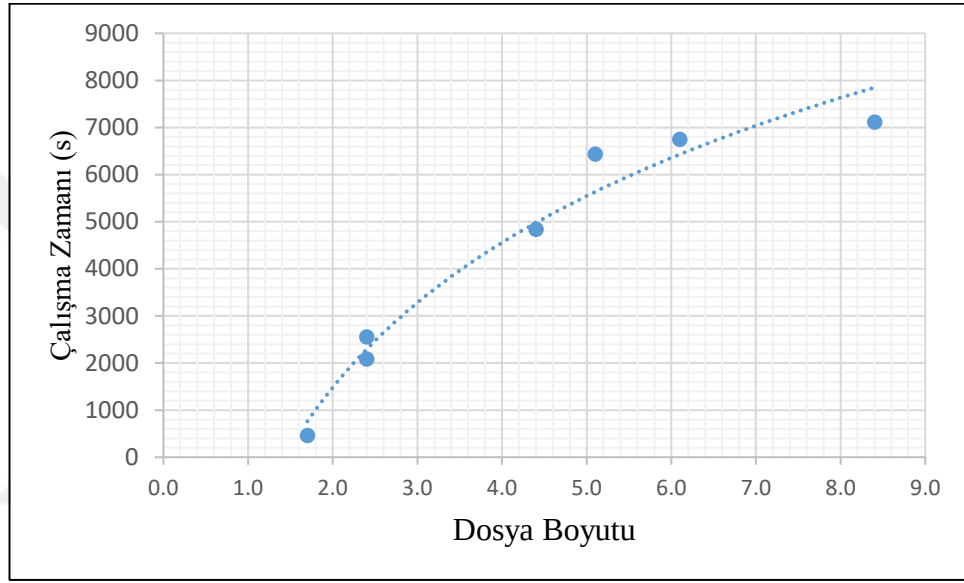
SAMTools aracı ile iş akışındaki bir sonraki işlem olan sıralama işlemi gerçekleştirilir. Sıralama işlemi için Şekil 3.5’ de yürütülen farklı boyuttaki girdi dosyalarının çalışma süresi ile doğrusal modellemesi görülmektedir. Eşitlik (3.7)’de çalışma süresi tahminlemede doğrusal modelin denklemi verilmiştir.



Şekil 3.5: Sıralama işleminin çalışma süresi tahmini için doğrusal model.

$$y = 776.68 \ln(x) - 285.51 \quad (3.7)$$

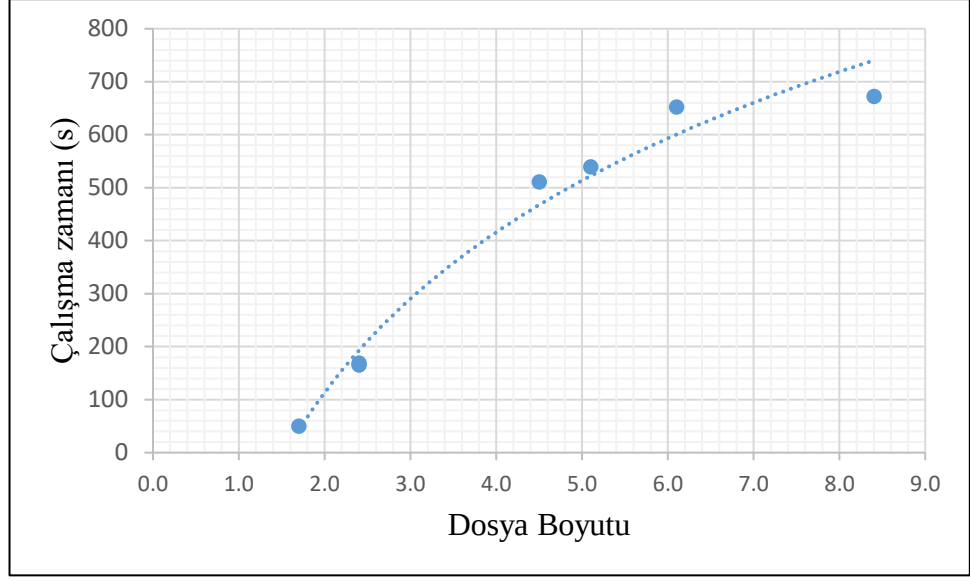
Bcftools aracı SNP ve kısa indel dizi varyantlarını çağırma kullanılır. Bcftools mpileup komutu ile her bir genomik pozisyondaki genotip olasılıklarının üretilmesi işlemi yapılır. Bu işlemi için Şekil 3.6’ de yürütülen farklı boyuttaki girdi dosyalarının çalışma süresi ile modellenmesi görülmektedir. Eşitlik (3.8)’de çalışma süresi tahminlemede logaritmik modelin denklemi verilmiştir.



Şekil 3.6: Genotip olasılıkların üretilmesi işleminin çalışma süresi tahmini için logaritmik model.

$$y = 4436.4 \ln(x) + 1594 \quad (3.8)$$

Bir sonraki işlemde ise Bcftools call komutu ile varyant araması yapılır. Bu işlemi için Şekil 3.7’ de yürütülen farklı boyuttaki girdi dosyalarının çalışma süresi ile modellenmesi görülmektedir. Eşitlik (3.9)’de çalışma süresi tahminlemede logaritmik modelin denklemi verilmiştir.



Şekil 3.7: Varyant arama işleminin çalışma süresi tahmini için logaritmik model.

$$y = 436.96 \ln(x) + 189.7 \quad (3.9)$$

4.HİPER SEZGİSEL YÖNTEM İLE İŞLEMCİ TAHSİSİ

Hiper sezgisel yöntemler, sezgisellerden oluşan uzayda arama gerçekleştirerek optimizasyon problemlerine çözüm oluştururlar. Bir problemin etkili çözümünü oluşturmak için farklı sezgiselleri veya sezgisel bileşenleri seçebilir, üretebilir ya da birleştirilebilirler.

Hiper-sezgisel terimi ilk olarak bir dizi yapay zekâ yöntemini seçmek ve birleştiren bir sistemi tanımlamak için kullanıldı [55]. Ancak 1960'lı yılların başında Atölye tipi çizelgeleme problemi çözümü için tek kural yerine kuralların birleştirilmesinin daha iyi sonuçlar verdiği gösterilmiştir [56]. Cowling, G. Kendall, E. Soubeiga [57] tarafından hiper-sezgisel yöntem, sezgiselleri seçmek için sezgisel olarak tanımlanmıştır. Son çalışmalarda hiper-sezgisel yöntem var olan sezgisellerden yeni sezgisel üreten arama metodu veya öğrenme mekanizması olarak tanımlanmıştır [58].

Hiper-sezgisel yaklaşımı sezgisel seçim metodolojisi ve sezgisel üretim metodolojisi olmak üzere iki ana kategoride sınıflandırılır [59], [60]. Sezgisel seçim metodolojileri, belirli bir noktaya uygulamak için arama uzayında düşük seviyeli bir buluşsal yöntem seçer. Sezgisel üretim metodolojileri, bir dizi düşük seviye sezgiselden otomatik olarak yeni buluşsal yöntemler üretir. Çözüm uzayında arama yapmak yerine sezgisel uzayında arama yapan hiper sezgisel yöntemler NP-Zor problemler için kaliteli çözümler oluşturmaktadır.

4.1. Genetik Algoritma

Genetik algoritma doğal seleksiyon ve genetik kavramından esinlenerek optimizasyon problemlerine uygun çözüm bulmayı amaçlar. Algoritma, global çözüme ulaşmak için çok sayıda birbirinden bağımsız çözüm üreterek arama uzayında birden fazla yerde arama yapar. Evrim teorisinden esinlenerek Genetik algoritma konusunda ilk çalışma yapan kişi John Honald' dır. Honald genetik algoritmayı yapay sistemlere uygulayarak geliştirmiştir [61].

Genetik algoritma çözüm uzayında amaç fonksiyonuna göre en iyileştirme amacıyla iterasyonlar gerçekleştirir. Yeni üretilcek çözümler amaç fonksiyonuna (uygunluk değeri) göre olasılıksal olarak değerlendirilir. Yüksek değere sahip olan çözümler yeni çözümler üretmesi için seçilir. Düşük değere sahip çözümler ise bir süre sonra seçilme olasılığı azaldığı için popülasyon dışında kalır. İterasyonlar tamamlandığında en iyi çözüm optimum ya da optimuma yakındır.

Gen, canlının kalıtsal molekülünde bulunan kalıtsal birime denir. Genetik algoritmada en küçük anlamlı birim olarak tanımlanır. Kromozom ise birden fazla genden oluşan bir çözümü ifade eden dizidir. Problem için olası çözüm kümesine popülasyon denir. Genetik algoritmada ilk adım başlangıç popülasyonun belirlenmesidir. Başlangıç popülasyonundaki ata (ebeveyn) kromozomlar sonraki jenerasyondaki çocuk kromozomları oluşturur. Uygunluk değeri yüksek olan kromozomların çaprazlama ile çocuk kromozomları oluşturma olasılığı yüksek olacaktır. Bu yüzden kromozomların uygunluk değeri hesaplanır. Ata kromozomların seçilimi için rulet tekerleği tekniği veya turnuva seçimi kullanılır. Rulet tekerleği tekniğinde uygunluk değeri kalitesi ile doğru orantılı yüzdesel pay dağıtılır. Uygunluk değeri yüksek olan kromozomlar daha yüksek pay aldıkları için seçilme olasılığı daha yüksek olacaktır. Turnuva seçiminde ise popülasyon içerisinde rastgele seçilen belirli sayıdaki kromozom arasında en iyi uygunluk değerine sahip olan kromozom seçilir. Çaprazlama iki ata kromozomun birbiri arasında gen değişimi yaparak yeni iki çocuk kromozom oluşturmaktır. Çaprazlamanın amacı seçilmiş olan uygunluk değeri yüksek iki ata kromozomdan uygunluk değeri yüksek çocuk kromozomlar oluşturarak daha iyi nesiller elde etmektir. Mutasyon işleminde ise çaprazlamadan sonra oluşan kromozomların gen bilgisinin değişmesi ile gerçekleşir. Buradaki amaç gen dizilimlerinin tekrar etmesini önleyerek genetik çeşitliliği sağlamaktır. Belirli bir iterasyon sayısına ulaşılması veya istenilen sonuca ulaşılması sonucunda algoritma sonlanır.

4.2. Genetik Algoritmanın Kullanımı

Bu tez çalışmasında işlemci tahsisi için genetik algoritma tabanlı açgözlü algoritma sezgisellerini kullanan hiper-sezgisel bir yöntem uygulanmıştır. Genetik algoritmanın en küçük anlamlı birim olan gen, açgözlü algoritmanın sezgiselini temsil

eder. Kromozomlarda işlemci tahsis edilecek işlem sayısı kadar gen bulunur. Her işlem bir gen ile eşleştirilir. İşleme tahsis edilecek işlemci sayısının belirlenmesinde genin temsil ettiği sezgisel kullanılır. CPA, MCPA, MCPA2, IAES algoritmalarında uygulanan işlemci tahsisi sezgiselleri kullanılmıştır. Şekil 4.1’de 6 işlem için kromozom örneği gösterilmektedir. Sırayla 6 işlem birer gene eşlenmiştir. Genin temsil ettiği sezgiseller 1- CPA sezgiseli, 2- MCPA sezgiseli, 3- MCPA2 sezgiseli, 4- IAES sezgiseli olarak kodlanmıştır. Şekil 4.1’de birinci işlem için işlemci tahsisinin MCPA sezgiseli ile yapılacağını ilk gen göstermektedir. İkinci gen ikinci işlem için işlemci tahsisinde CPA sezgiselinin kullanılacağını gösterir. Diğer genlerde benzer şekilde işlemler ile eşleştirilmiş ve açgözlü sezgisellerin temsili ile kodlanmıştır.

2	1	4	3	1	4
---	---	---	---	---	---

Şekil 4.1: Kromozom.

CPA sezgiseli işlemci tahsisinde veri paralelliğinin avantajlarından faydalanır. MCPA veri ve işlem paralelliğinin dengeli olduğu iş akışlarında iyi sonuçlar verir. MCPA2 düzensiz iş akışlarında MCPA algoritmasına göre daha iyi sonuçlar üretir. IAES algoritması bir iş akışında aynı katmanda bulunan işlemlerin iş yüklerinin eşit olmadığı durumda diğer üç algoritmaya göre daha iyi sonuçlar vermektedir. Hiper-sezgisel yöntem ile farklı iş akışlarında ve iş yüklerinde bu dört sezgisel yöntemin avantajlarının kullanılması amaçlanmaktadır.

İşlemci tahsisi için meta-sezgisel yöntem olarak genetik algoritmanın kullanıldığı çalışmalarda genler işlemlere tahsis edilecek işlemci sayısını temsil ederken kromozomlar tam çözümü göstermektedir. Bu yüzden algoritma çözüm uzayında en iyi çözümü aramaktadır. Önerilen hiper-sezgisel yöntemde ise genler, işlemlere tahsis edilecek işlemci sayının belirlenmesinde kullanılacak açgözlü sezgiselleri temsil eder. Hiper-sezgisel yöntem sezgisellerden oluşan uzayda arama yapar. Başlangıç popülasyonu rastgele belirlenmiştir.

4.2.1. Amaç Fonksiyonu

Kromozom uygunluk değerini hesaplanması için çizelgeleme algoritması uygulanır. Her işlem için kromozomda tanımlı sezgisele göre tahsis edilen işlemci sayısı belirlenir. İşlemci sayısı belirlenmiş olan işlemler ile liste çizelgeleme prosedürü yürütülür. Liste çizelgeleme prosedüründe işlemler öncelik değerlerine göre uygun işlemcilere atanırlar. İşlem t_i 'nin öncelik değeri ($p(t_i)$), çizgede ilgili düğümden çıkış düğümüne giden en uzun yol boyunca bulunan tüm düğümlerin maliyetinin toplamıdır. Şekil 4.2 'de sözde kodu verilmiştir. Öncelik değerlerine göre işlemler sıralanır. Tüm işlemlerin atanacağı işlemciler belirlenene kadar maksimum öncelik değerine sahip olan işlem uygun işlemciye (tahsis edilen işlemci sayısı kadar) atanır. Algoritma sonlandığında çizelgenin makespan kromozomun uygunluk değerini verir.

#ListeÇizelgeleme
1. Öncelik değeri $p(t_i)$ 'ye göre işlemleri sırala
2. Tüm işlemlere işlemci atanana kadar:
3. Maksimum $p(t_i)$ değerine sahip t_i işlemini seç
4. t_i işlemine tahsis edilmiş işlemci sayısı kadar uygun ilk işlemcileri ata

Şekil 4.2: Liste Çizelgeleme Sözde Kodu.

4.2.2. Çaprazlama

Ata kromozomların seçilmesi için turnuva tekniği uygulanır. Kromozomların çaprazlanması için rastgele üretilmiş sayı kromozomda kaçınıcı genden sonra değişimin gerçekleşeceğini gösterir. Şekil 4.3'de ata kromozomda kaçınıcı genden itibaren kromozomların kesileceği rastgele bir sayı ile belirlenir. Kromozom belirlenen üçüncü genden itibaren kesilir. Birinci ve ikinci ata kromozomun kesilen genleri yer değiştirir. Değişim sonucu iki yeni çocuk kromozom oluşur.

	Rastgele Belirlenen Nokta					
	↓					
1. Ata	2	1	4	3	1	4
2. Ata	4	2	1	4	3	3
1.Çocuk	2	1	4	4	3	3
2.Çocuk	4	2	1	3	1	9

Şekil 4.3: Hiper-sezgisel GA Çaprazlama.

4.2.3. Mutasyon

Algoritmada belirli olasılıkla mutasyon işleminin gerçekleşmesine karar verilir. Oluşan çocuk kromozomlarda mutasyon işlemi yapılarak yeni kromozomlar oluşturulur. Mutasyon işleminde çocuk kromozomun her bir geni için değişsin ya da değişmesin kararı rastgele verilir. Değişmesine karar verilen her bir gen için diğer rastgele seçilen bir sezgisel ile değişimi sağlanır. Şekil 4.4 de çocuk kromozomun üçüncü ve altıncı genleri için değişsin kararı rastgele verilmiştir. Bu genlerin sezgisellerinin değişmesi ile yeni kromozom oluşmuştur.

2	1	4	3	1	4
2	1	1	3	1	3

Şekil 4.4: Hiper-sezgisel GA Mutasyon.

4.3. Statik Çizelgeleme Deneysel Sonuçları

Tez kapsamında geliştirilen statik çizelgeleme testleri 4 çekirdekli i7 2.70 Ghz işlemci, 8 GB ana bellek üzerinde Microsoft 10 işletim sisteminde koşturmaktadır. Geliştirilen algoritmalar C# programlama dili kullanılarak kodlanmıştır.

Algoritmamızda başlangıç popülasyonu sayısı 50, jenerasyon sayısı ise 70'dir. Algoritma, çaprazlama olasılığı 0.7 ve mutasyon olasılığı 0.05 ile yürütülmüştür. Tüm parametreler yapılan seri testler sonucu belirlenmiştir.

Performans metriği olarak makespan ve çizelge uzunluğu oranı (SLR) kullanıldı. SLR, çizgenin makespan değerinin kritik yol uzunluğuna olan oranıdır. Kritik yol uzunluğu hesaplanırken tüm işlemlere tek işlemci atandığı varsayılarak çalışma süreleri belirlenir. SLR, çizgenin boyutuna bağımlı olmadan algoritmanın performansını ölçmeyi hedefler.

Bu tez çalışmasında statik çizelgeleme problemleri için işlemci tahsisinde hiper-sezgisel yöntem ve çizelgelemede liste çizelgeleme prosedürü uygulanmıştır. Önerilen hiper sezgisel yöntem CPA, MCPA, MCPA2, IAES ağgözlü algoritmaları ile karşılaştırılmıştır. Problemden 8, 16, 24, 32 gibi değişen işlemci sayısına sahip işlemci kaynaklarının tüm işlemcilerinin aynı kapasiteye sahip olduğu (homojen işlemci kaynakları) varsayılır.

Hiper-sezgisel yöntem ile işlemci tahsisi, rastgele seçilmiş sezgiseller ile işlemci tahsisi çözümleri ve her işlem için tekrarsız sezgisel permütasyonu ile oluşturulan çözümler ile karşılaştırılmıştır. CPA, MCPA, MCPA2, IAES algoritmalarında uygulanan işlemci tahsisi sezgiselleri kullanılmıştır. Rastgele sezgiseller ile çözüm oluşturma yönteminde her bir işlemci tahsisi çözümünde işlemler için sezgiseller rastgele seçilmiştir. Çizelgelemede liste çizelgeleme prosedürü uygulanmıştır. Yirmi adet çözümün ortalaması alınarak hiper-sezgisel işlemci tahsisi ile karşılaştırılmıştır. Diğer bir çözüm oluşturma yöntemi ise her bir işlem için işlemci tahsisinde sezgisellerin tekrarsız permütasyon ile belirlenmesidir. Dört sezgisel ile yirmi dört tekrarsız sıralama yapabilir. Bu sıralama ile işlemlerin işlemci tahsisini belirleyen sezgiseller seçilir. Yirmi dört işlemden daha fazla işleme sahip problem için sıranın başından başlanarak sezgiseller işlemci tahsisi için belirlenir. Çizelgelemede liste çizelgeleme prosedürü uygulanır. Hiper-sezgisel yöntem ile elde edilen çözümlerle karşılaştırılmıştır.

Deneyisel çalışmalarda standart işlem çizge, gerçek dünya uygulamaları ve yeni nesil dizileme analiz iş akışı kullanılmıştır. Standart işlem çizge ve gerçek dünya uygulamalarının veri paralel işlemlerinin çalışma süresinin hesaplanmasında Amdahl yasası kullanılmıştır. Veri paralel işlemler için paralellik değeri 0.9 olarak belirlenmiştir. Yeni nesil dizileme iş akışındaki veri paralel işlem olan hizalama için ise Bölüm 3.1 ve Tablo 3.1’de verilen F_{est} değeri (0.960558) kullanılmıştır.

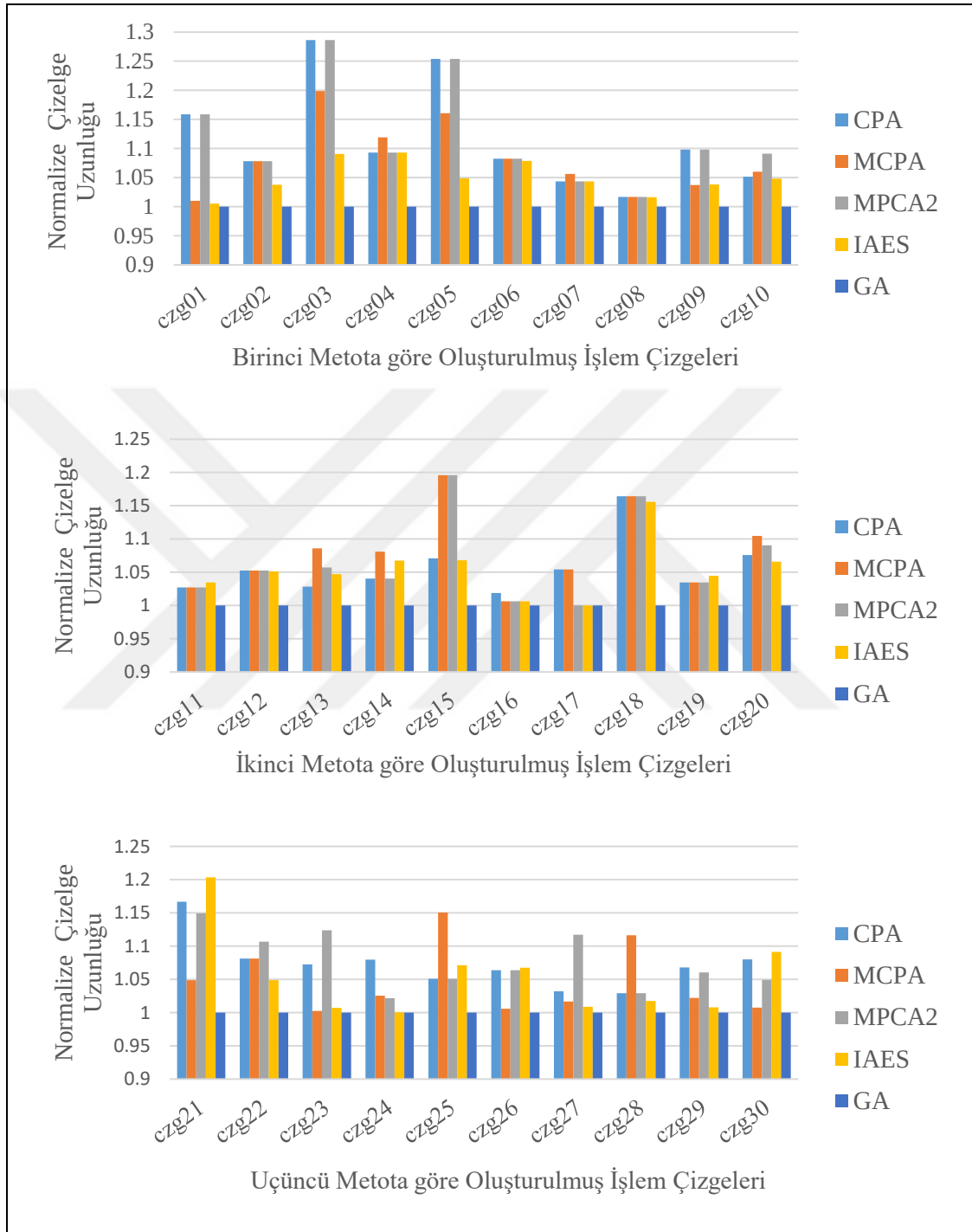
4.3.1. Standart İşlem Çizgeleri

Hiper-sezgisel işlemci tahsisi algoritmasının performansını ölçmek amacıyla standart işlem çizgeleri (STG) kullanılmıştır. 50 işlem içeren 30 çizge ve 100 işlem içeren 30 çizge kullanılmıştır [62], [63].

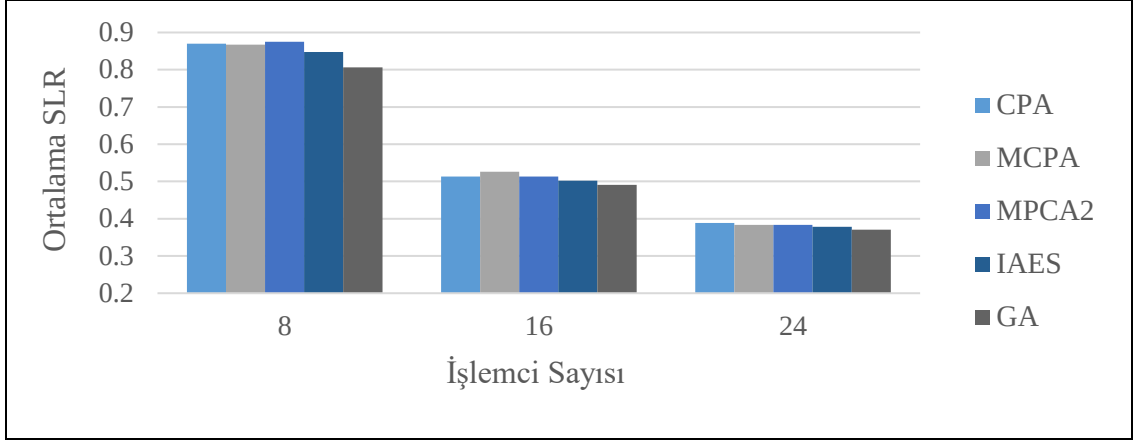
Standart işlem çizgeleri oluşturulurken üç farklı metot kullanılmıştır. Çizgede iki düğüm arasındaki kenar bağımlılığı oranı p ile gösterilir. Birinci metotta öncelikle bağımsız düğüm sayısına karar verilir [64]. Düğümler arasında kenarlar p değerine göre oluşturulur. Birinci metotta çizgelerde p değeri 0.05 ile 0.2 arasında değişir. İkinci metotta ise öncelikle çizgedeki katman sayısına karar verilir [65]. Sonrasında her bir katmandaki düğüm sayısına rastgele karar verilir. İkinci metotta çizgelerde p değeri 0.1 ile 0.2 arasında değişir. Bu değere göre düğümler arasındaki kenarlar oluşturulur. Diğer üçüncü metot ise birinci metot ile benzerdir. Birinci metottan farkı her düğüm için öncül düğüm sayısının belirlenmiş olmasıdır. Bu üç metot ile oluşturulmuş çizgeler deneysel çalışmada kullanılmıştır.

Şekil 4.5’de 8 işlemcili kaynakta her biri 50 işlem içeren 30 çizge için önerilen hiper-sezgisel algoritmanın diğer dört açgözlü algoritmalara göre maksimum tamamlanma süresinin normalize edilmiş sonuçları gösterilmektedir. Normalize edilmiş değer açgözlü algoritmanın maksimum tamamlanma süresinin hiper-sezgisel algoritma ile elde edilen maksimum tamamlanma süresine oranıdır. CPA, MCPA, MCPA2 algoritmalarının czg02, czg06, czg08, czg11, czg18, czg19 çizgelerinde verdiği sonuçlar aynıdır. IAES algoritması ise diğer üç açgözlü algoritmaya göre daha iyi sonuçlar vermiştir. Tüm çizgelerde önerilen algoritma açgözlü algoritmalara göre daha iyi maksimum tamamlanma süresi (makespan) vermiştir. Şekil 4.6’da ise çizgelerin 8, 16, 24 işlemcili hesaplama kaynakları üzerindeki SLR performansı gösterilmiştir. Önerilen algoritma ile farklı işlemci sayısına sahip kaynaklarda da daha

iyi sonuçlar elde edilmiştir. Şekil 4.6'da işlemci sayısının ağgözlü algoritmalar için performansı etkilediği görülmektedir.

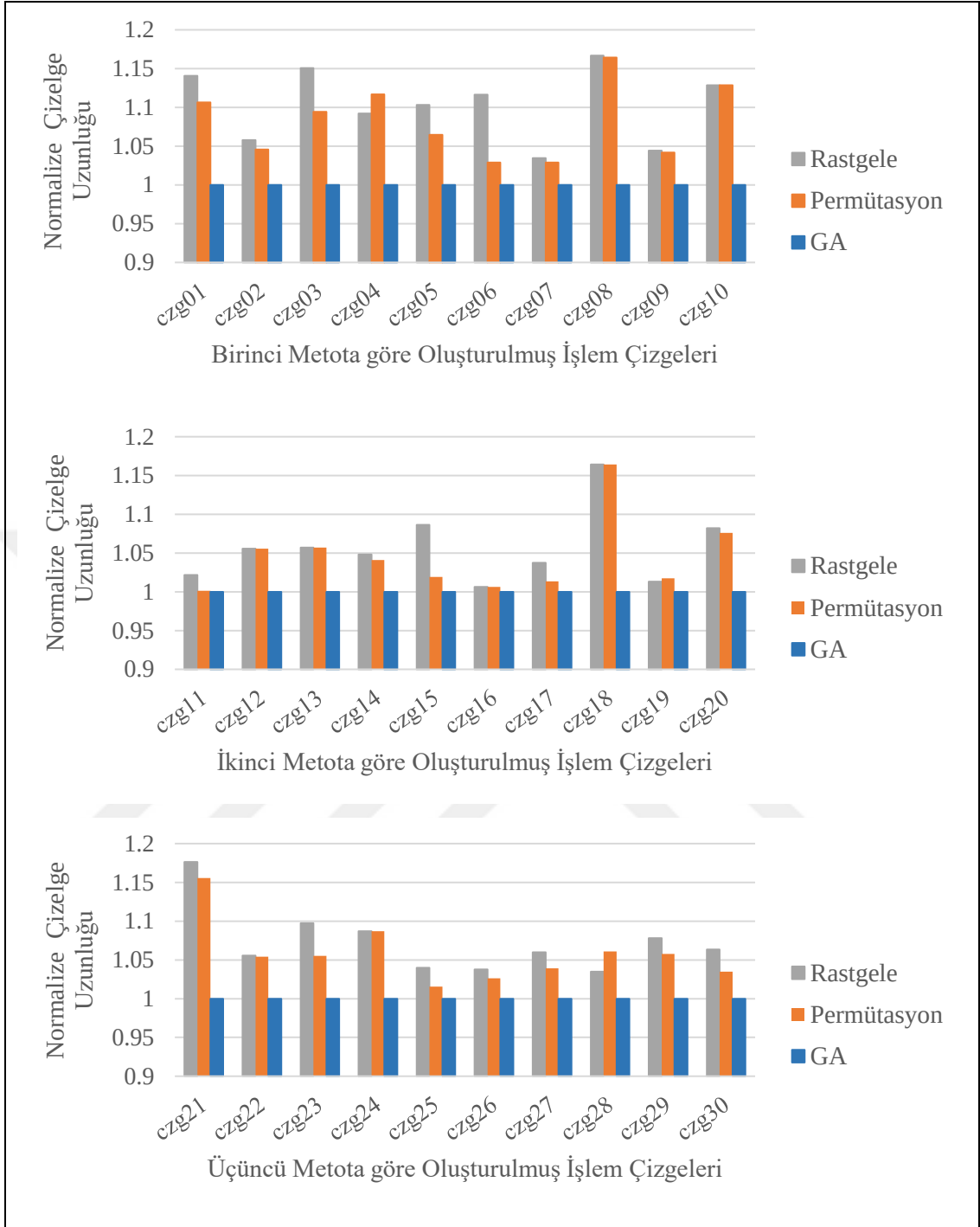


Şekil 4.5: 50 işlem içeren 30 çizge ile 8 işlemcili kaynakta ağgözlü algoritmalar ile makespan karşılaştırması.

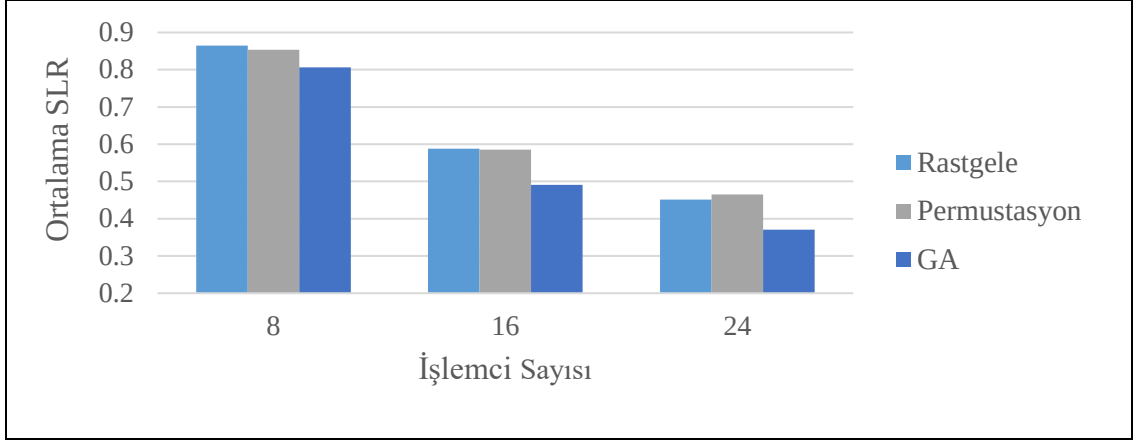


Şekil 4.6: 50 işlem içeren 30 çizge ile ağgözlü algoritmalar ile SLR karşılaştırması.

Şekil 4.7’de 8 işlemcili kaynakta 50 işlem içeren 30 çizge için önerilen hiper-sezgisel algoritmanın, rastgele seçilmiş sezgiseller ile işlemci tahsisi çözümünün ve her işlem için tekrarsız sezgisel permütasyonu ile oluşturulan çözümün maksimum tamamlanma sürelerinin normalize edilmiş sonuçları gösterilmektedir. Şekil 4.8’de ise çizgelerin 8, 16, 24 işlemcili hesaplama kaynakları üzerindeki SLR performansı gösterilmiştir. Önerilen algoritma rastgele sezgiseller ile oluşturulmuş çözüme göre %12, sezgisellerin permütasyonuna göre oluşturulmuş çözüme göre ise %10 daha iyi sonuçlar vermiştir.

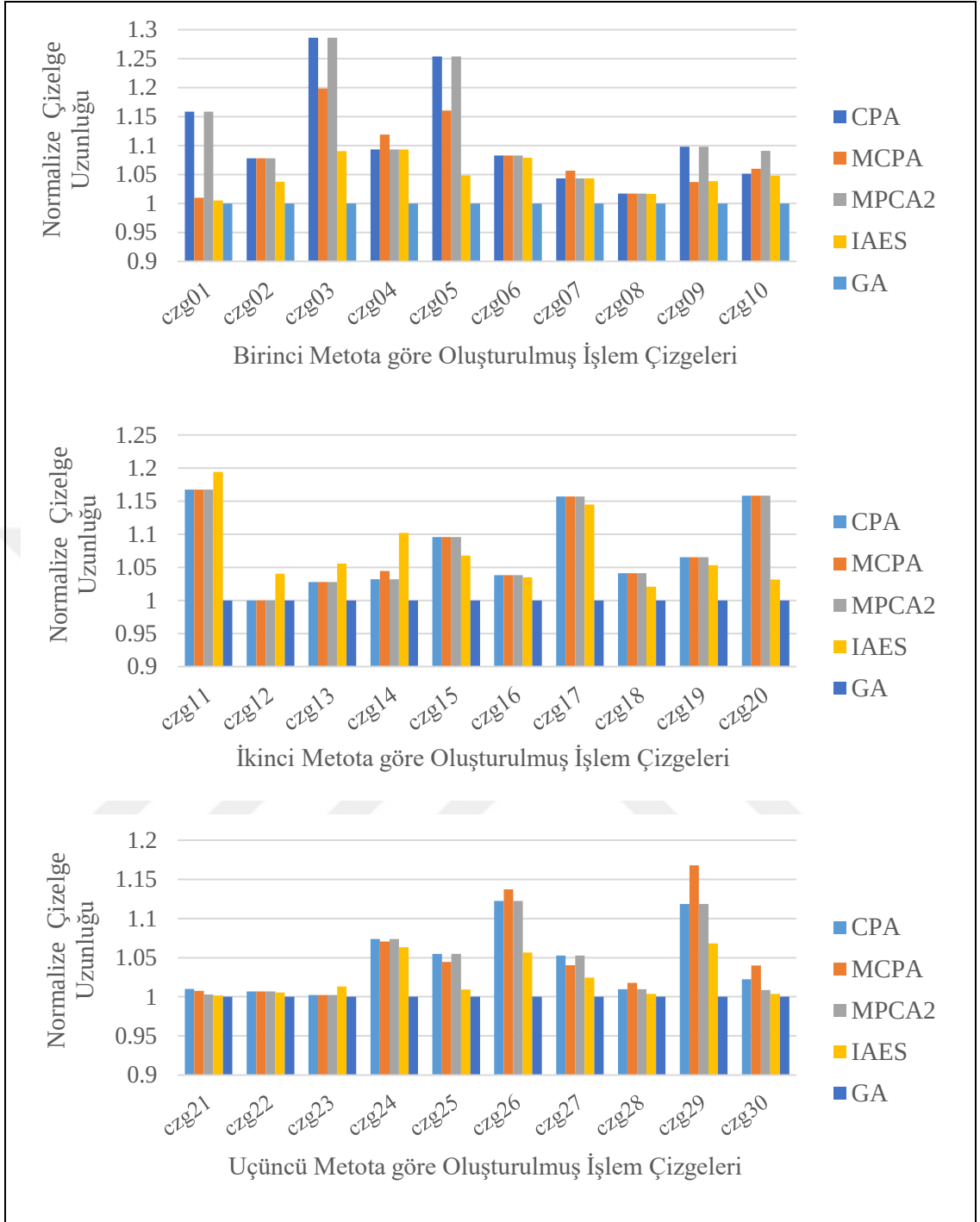


Şekil 4.7: 50 işlem içeren 30 çizge ile 8 işlemcili kaynakta rastgele ve permütasyon çözümleri ile makespan karşılaştırması.

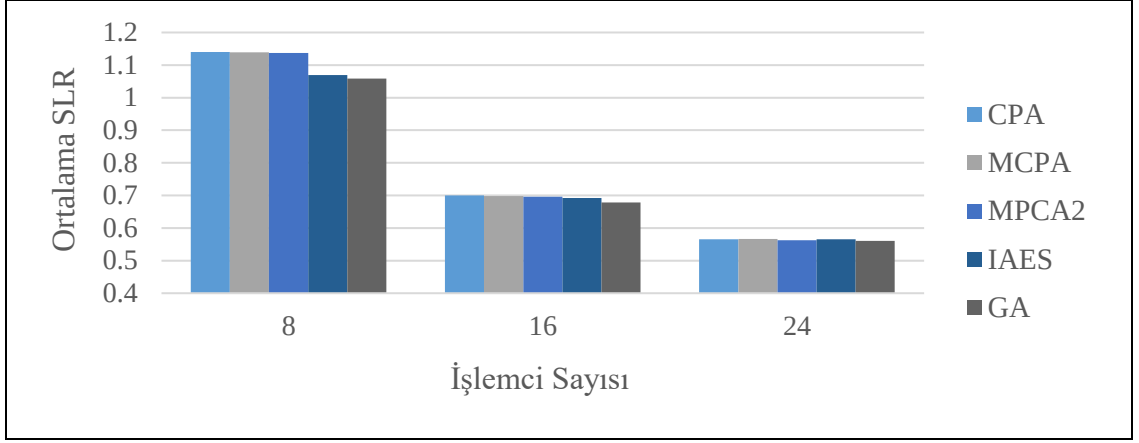


Şekil 4.8: 50 işlem içeren 30 çizge ile rastgele ve permütasyon çözümleri ile SLR karşılaştırması.

Şekil 4.9’da 8 işlemcili kaynakta her biri 100 işlem içeren 30 çizge için önerilen hiper-sezgisel algoritmanın ağgözlü algoritmalara göre maksimum tamamlanma süresinin normalize edilmiş sonuçları gösterilmektedir. Şekil 4.10’da ise çizgelerin 8, 16, 24 işlemcili hesaplama kaynakları üzerindeki SLR performansı gösterilmiştir. Önerilen algoritmanın diğer algoritmalarından daha iyi sonuçlar verdiği görülür. Ağgözlü algoritmalar işlemci sayısının artması ile performansı artar. Bu durumun veri paralel görevlere atanan işlemci sayısındaki artış ile paralel olduğu görülmektedir.

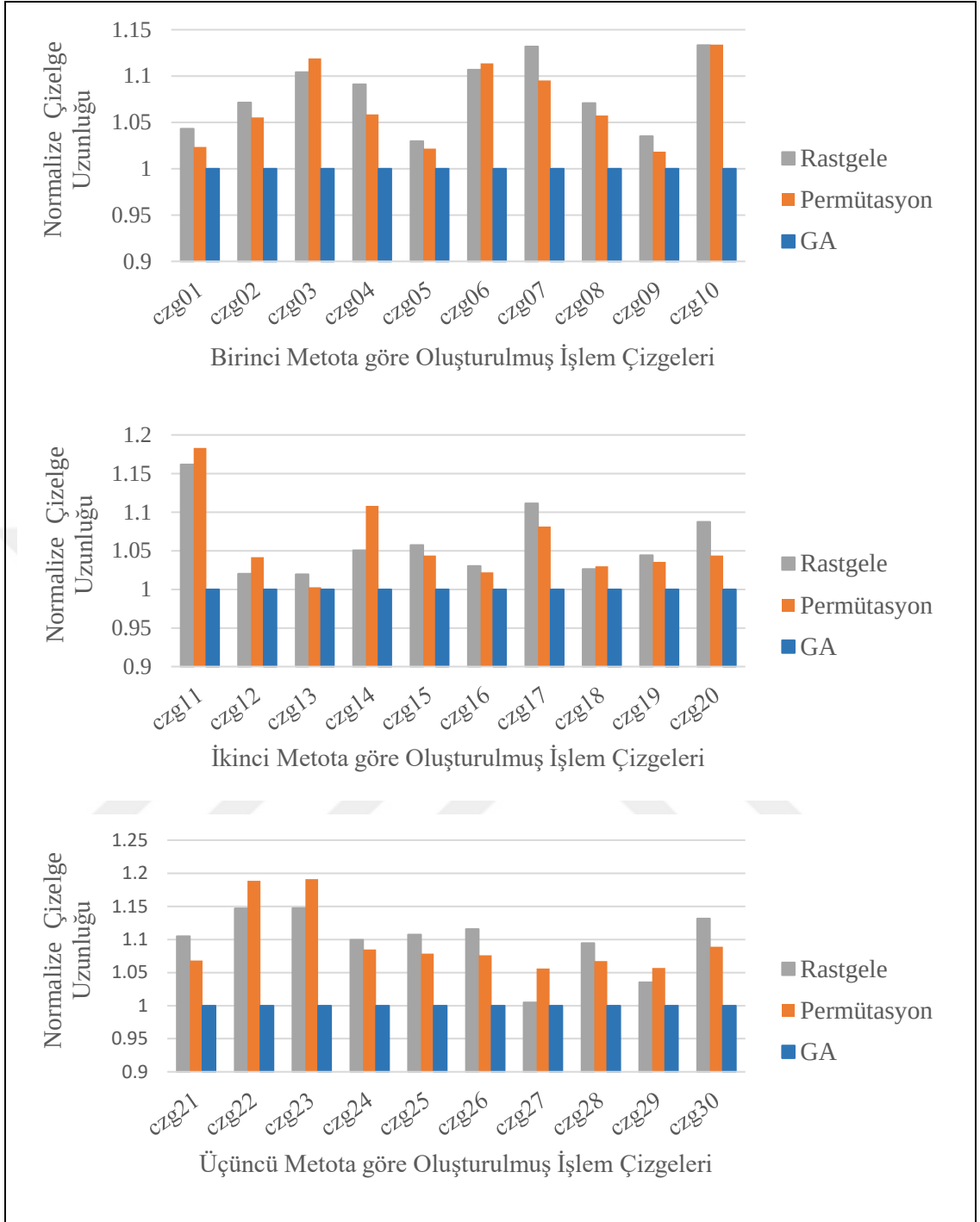


Şekil 4.9: 100 işlem içeren 30 çizge ile 8 işlemcili kaynakta ağgözlü algoritmalar ile makespan karşılaştırması.

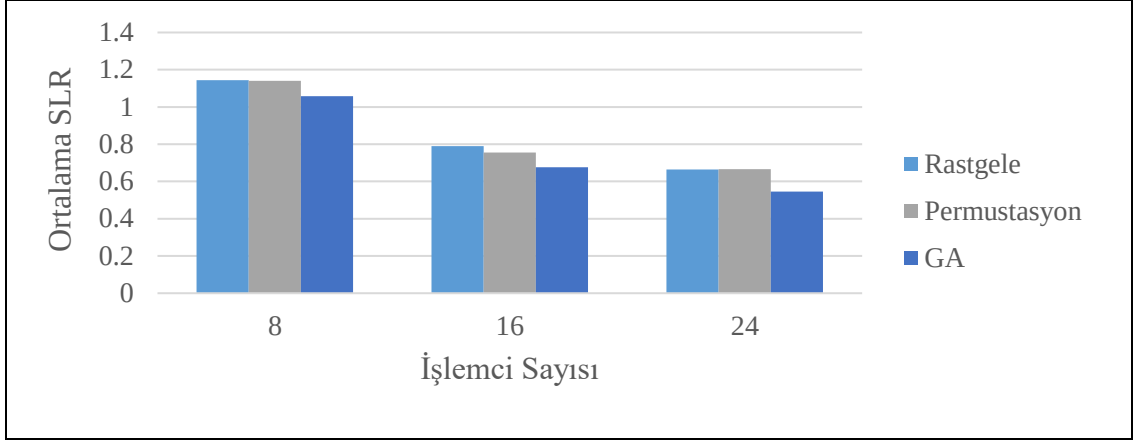


Şekil 4.10: 100 işlem içeren 30 çizge ile ağgözlü algoritmalar ile SLR karşılaştırması.

Şekil 4.11’de 8 işlemcili kaynakta 100 işlem içeren 30 çizge için önerilen hiper-sezgisel algoritmanın, rastgele seçilmiş sezgiseller ile işlemci tahsisi çözümünün ve her işlem için tekrarsız sezgisel permütasyonu ile oluşturulan çözümün maksimum tamamlanma sürelerinin normalize edilmiş sonuçları gösterilmektedir. Şekil 4.12’de ise çizgelerin 8, 16, 24 işlemcili hesaplama kaynakları üzerindeki SLR performansı gösterilmiştir. Önerilen algoritma, rastgele sezgiseller ile oluşturulmuş çözüme göre %10, sezgisellerin permütasyonuna göre oluşturulmuş çözüme göre ise %9 daha iyi sonuçlar vermiştir.



Şekil 4.11: 100 işlem içeren 30 çizge 8 işlemcili kaynakta rastgele ve permütasyon çözümleri ile makespan karşılaştırması.

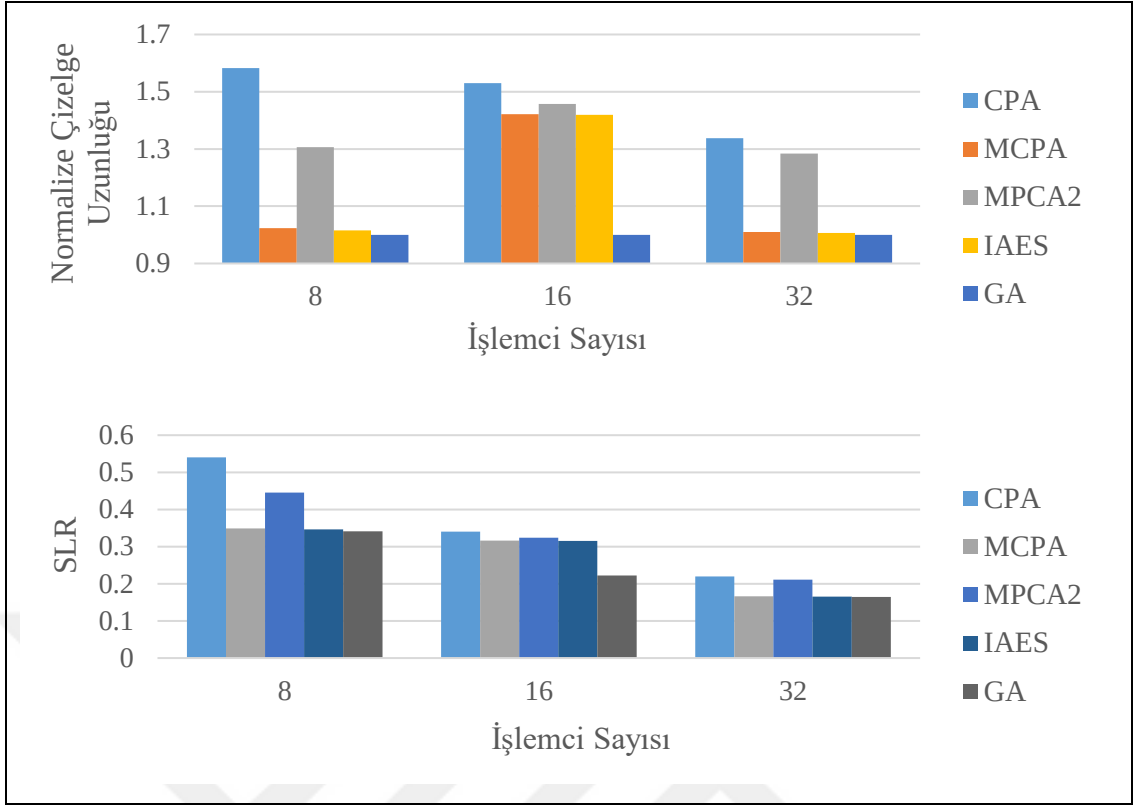


Şekil 4.12: 100 işlem içeren 30 çizge rastgele ve permütasyon çözümleri ile SLR karşılaştırması.

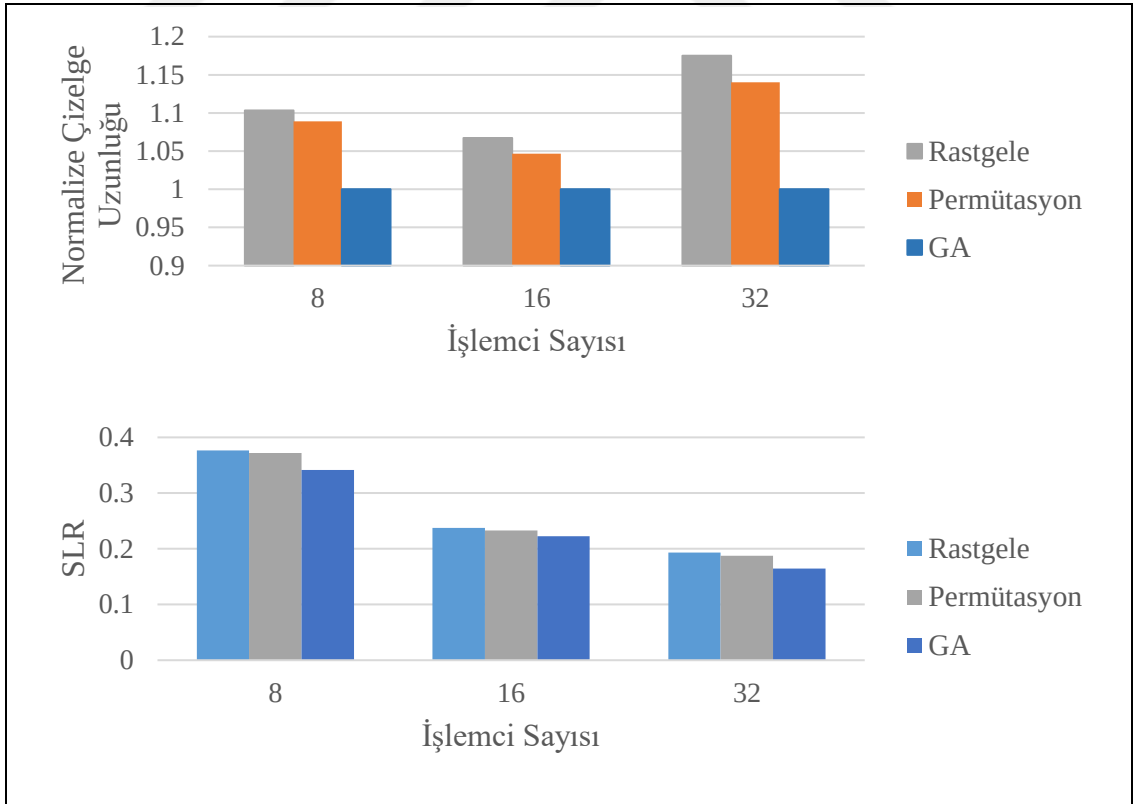
4.3.2. Uygulama Çizgeleri

Hiper-sezgisel işlemci tahsisi algoritmasının performansını ölçmek amacıyla gerçek dünya uygulamaları kullanılmıştır. Matrix Multiplacation (Matmul) uygulaması birçok veri ve işlem paralel çizgelerin çizelgelenmesi problemi üzerine geliştirilmiş birçok çalışmada performans değerlendirmesinde kullanılmıştır [4]- [7]. Matmul çizgesinde 7 işlem ve 10 kenar bulunmaktadır. Şekil 4.13’ da Matmul’un makespan ve SLR sonuçları 8, 16, 32 işlemcili kaynaklarda görülmektedir. CPA algoritması bu çizgede kaynak parçalanmasına yol açar. CPA algoritması en kötü sonucu verirken önerilen hiper-sezgisel algoritma en iyi sonucu vermektedir.

Şekil 4.14’ de önerilen hiper-sezgisel algoritmanın, rastgele seçilmiş sezgiseller ile işlemci tahsisi çözümünün ve her işlem için tekrarsız sezgisel permütasyonu ile oluşturulan çözümün makespan ve SLR performans metriğine göre karşılaştırması verilmiştir. Önerilen algoritma, rastgele sezgiseller ile oluşturulmuş çözüme göre %11, sezgisellerin permütasyonuna göre oluşturulmuş çözüme göre ise %9 daha iyi sonuçlar vermiştir.

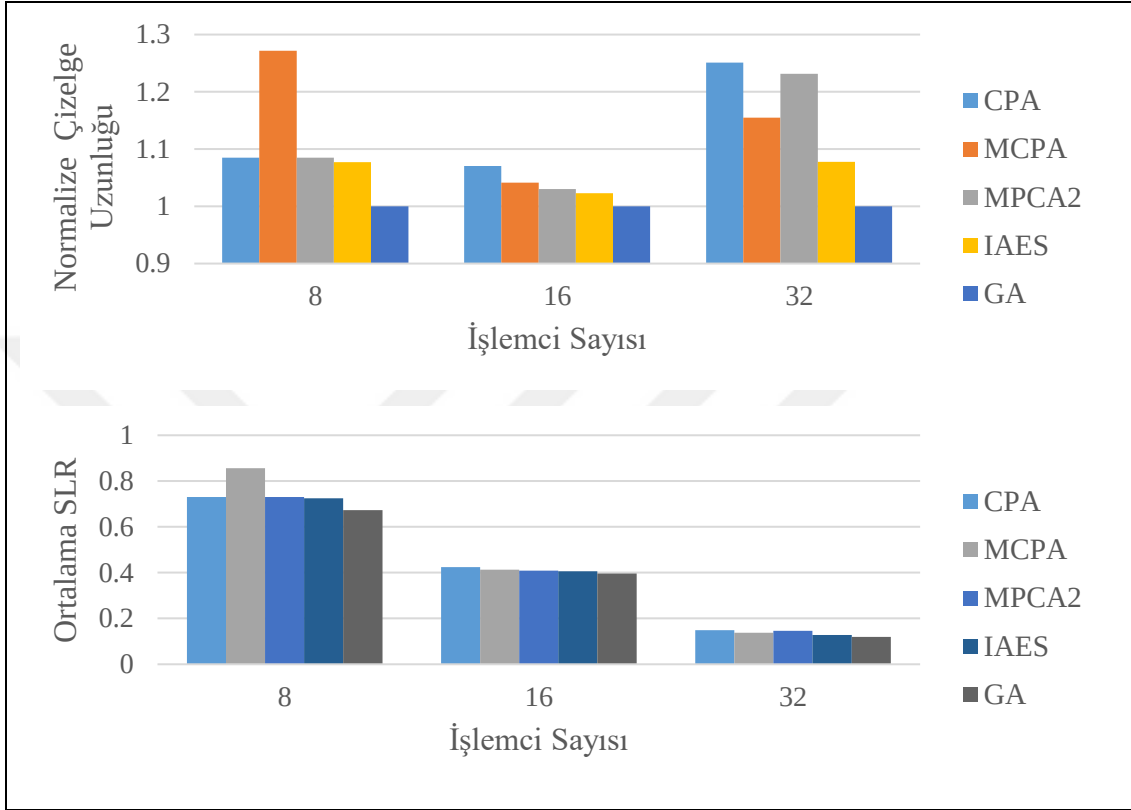


Şekil 4.13: Matmul için ağgözlü algoritmalar ile makespan ve SLR karşılaştırması.



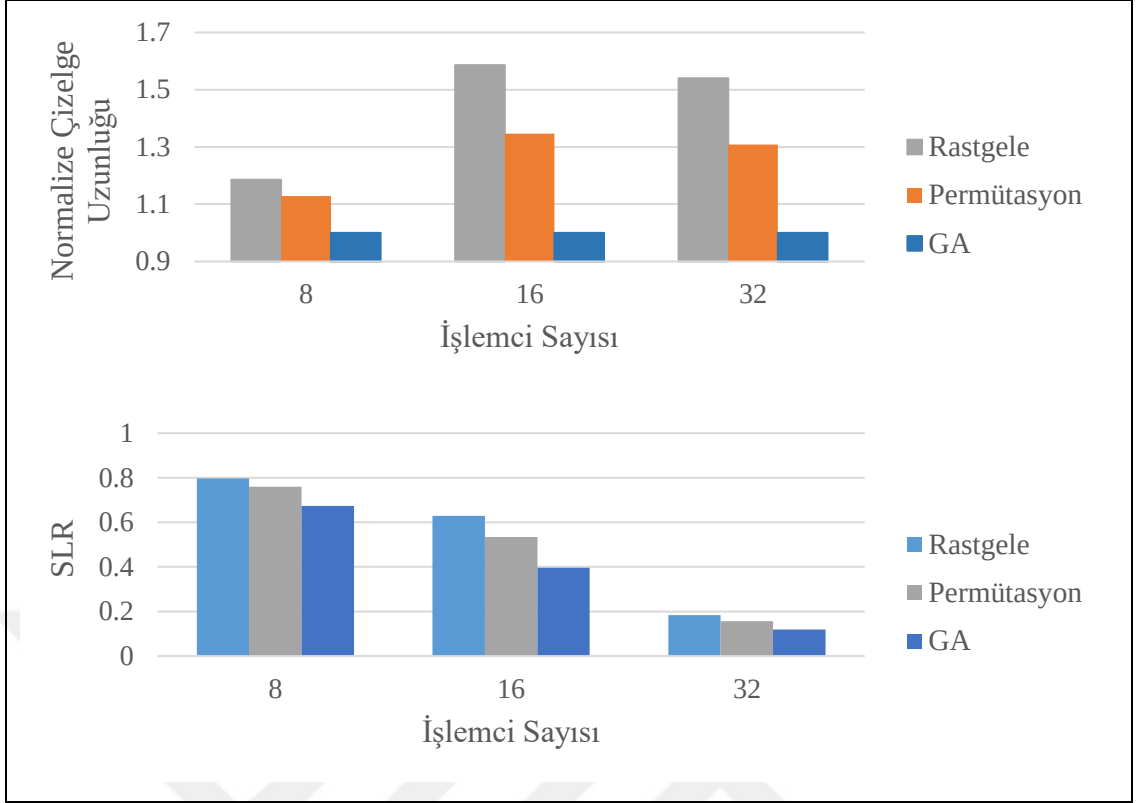
Şekil 4.14: Matmul için rastgele ve permütasyon çözümleri ile makespan ve SLR karşılaştırması.

Strassen Matrix Multiplication (Strassen) diğer popüler gerçek dünya uygulamasıdır [4]- [7]. Bu uygulama 26 işlem ve 46 kenar içerir. Şekil 4.15’ de Strassen’in makespan ve SLR sonuçları 8, 16, 32 işlemcili kaynaklarda görülmektedir. Tüm koşullarda önerilen algoritma en iyi sonucu vermektedir.



Şekil 4.15: Strassen için açgözlü algoritmalar ile makespan ve SLR karşılaştırması.

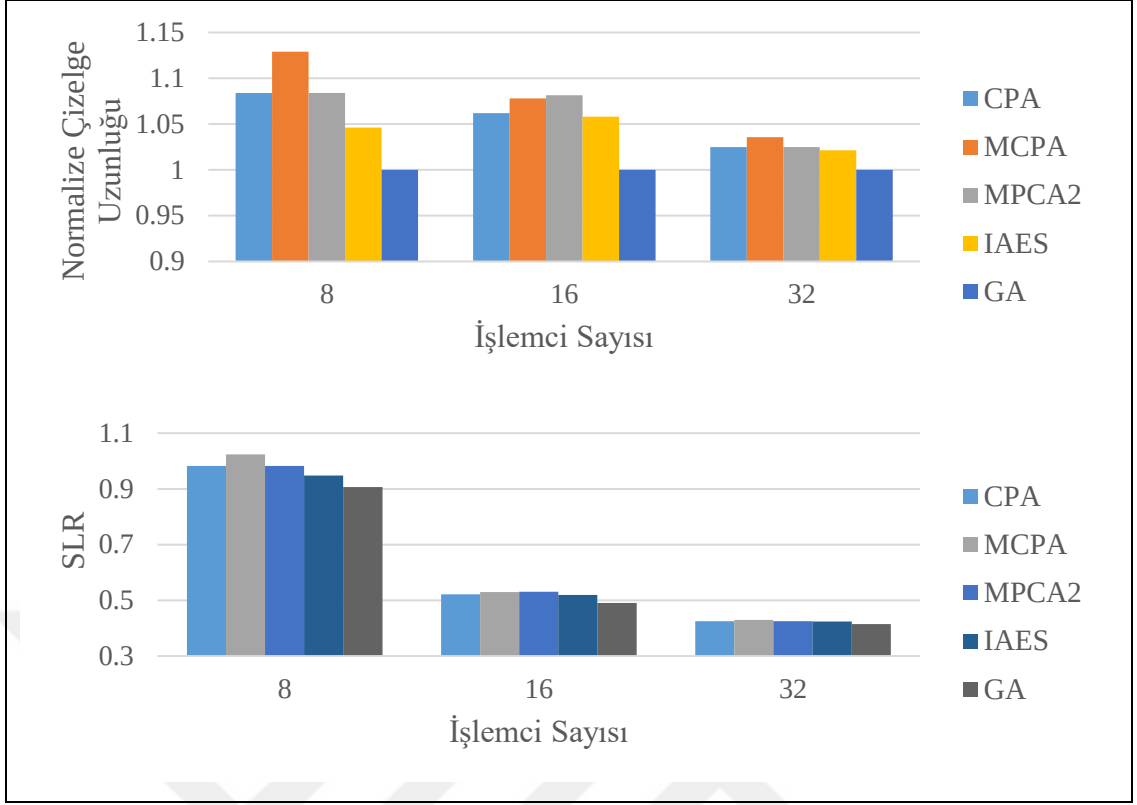
Şekil 4.16’ de Strassen uygulama çizgesi için önerilen hiper-sezgisel algoritmanın, rastgele seçilmiş sezgiseller ile işlemci tahsisi çözümünün ve her işlem için tekrarsız sezgisel permütasyonu ile oluşturulan çözümün makespan ve SLR performans metriğine göre karşılaştırması verilmiştir. Önerilen algoritma, rastgele sezgiseller ile oluşturulmuş çözüme göre ortalama %33, sezgisellerin permütasyonuna göre oluşturulmuş çözüme göre ise %20 daha iyi sonuçlar vermiştir.



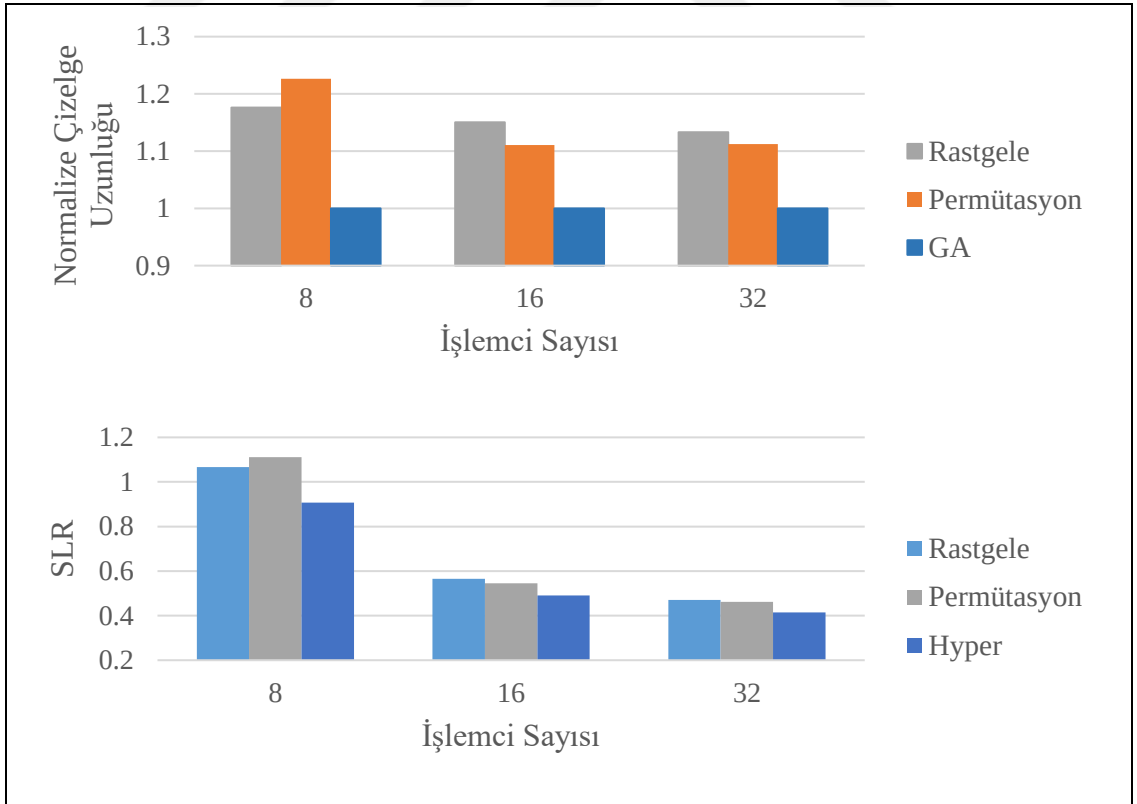
Şekil 4.16: Strassen için rastgele ve permütasyon çözümleri ile makespan ve SLR karşılaştırması.

Spec fppp, popüler gerçek dünya uygulamasıdır [68]. Bu uygulama 334 işlem ve 1145 kenar içerir. Şekil 4.17’ de çizgenin makespan ve SLR sonuçları 8, 16, 32 işlemcili kaynaklarda görülmektedir. Tüm koşullarda önerilen algoritma en iyi sonucu vermektedir.

Şekil 4.18’ de Spec fppp uygulama çizgesi için önerilen hiper-sezgisel algoritmanın, rastgele seçilmiş sezgiseller ile işlemci tahsisi çözümünün ve her işlem için tekrarsız sezgisel permütasyonu ile oluşturulan çözümün makespan ve SLR performans metriğine göre karşılaştırması verilmiştir. Önerilen algoritma, rastgele sezgiseller ile oluşturulmuş çözüme göre ortalama %15, sezgisellerin permütasyonuna göre oluşturulmuş çözüme göre ise %15 daha iyi sonuçlar vermiştir.

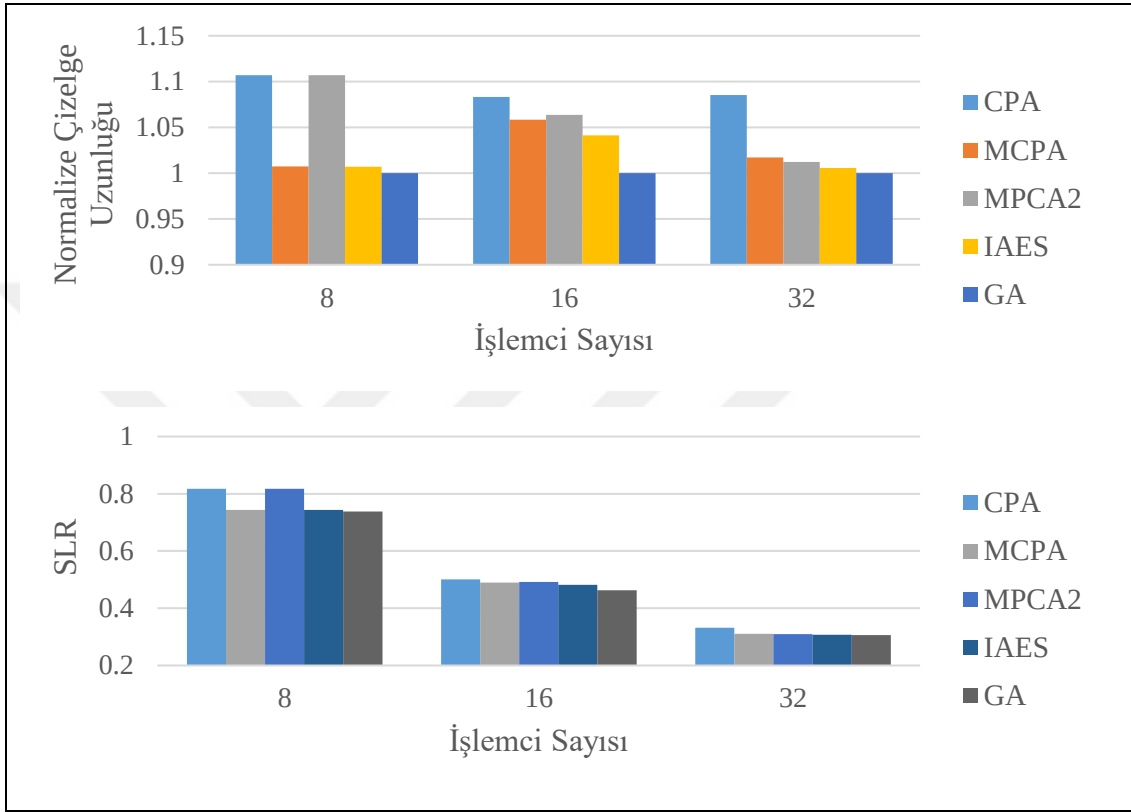


Şekil 4.17: Spec fppp için ağgözlü algoritmalar ile makespan ve SLR karşılaştırması.



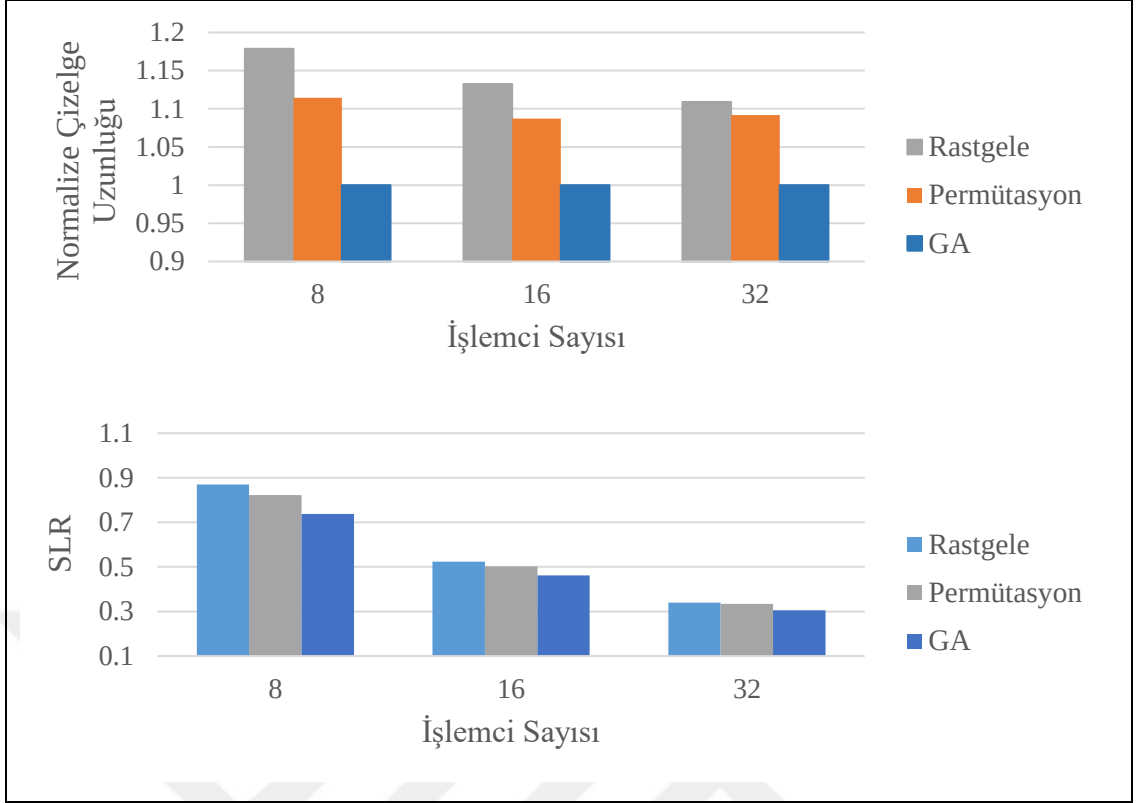
Şekil 4.18: Spec fppp için rastgele ve permütasyon çözümleri ile makespan ve SLR karşılaştırması.

Robot Kontrol, gerçek dünya uygulamasıdır [71]. Çizgesi 88 işlem ve 131 kenara sahiptir. Robot kontrol çizgesini 8, 16, 32 işlemci kaynakları üzerindeki sonuçları Şekil 4.19’da gösterilmiştir. Farklı işlemci sayısına sahip kaynaklarda önerilen algoritma en iyi sonucu vermiştir. İşlemci sayısı arttıkça açgözlü algoritmalar daha iyi sonuçlar vermektedir.



Şekil 4.19: Robot Kontrol için açgözlü algoritmalar ile makespan ve SLR karşılaştırması.

Şekil 4.20’ de Robot kontrol uygulama çizgesi için önerilen hiper-sezgisel algoritmanın, rastgele seçilmiş sezgiseller ile işlemci tahsisi çözümünün ve her işlem için tekrarsız sezgisel permütasyonu ile oluşturulan çözümün makespan ve SLR performans metriğine göre karşılaştırması verilmiştir. Önerilen algoritma, rastgele sezgiseller ile oluşturulmuş çözüme göre ortalama %14, sezgisellerin permütasyonuna göre oluşturulmuş çözüme göre ise %10 daha iyi sonuçlar vermiştir.

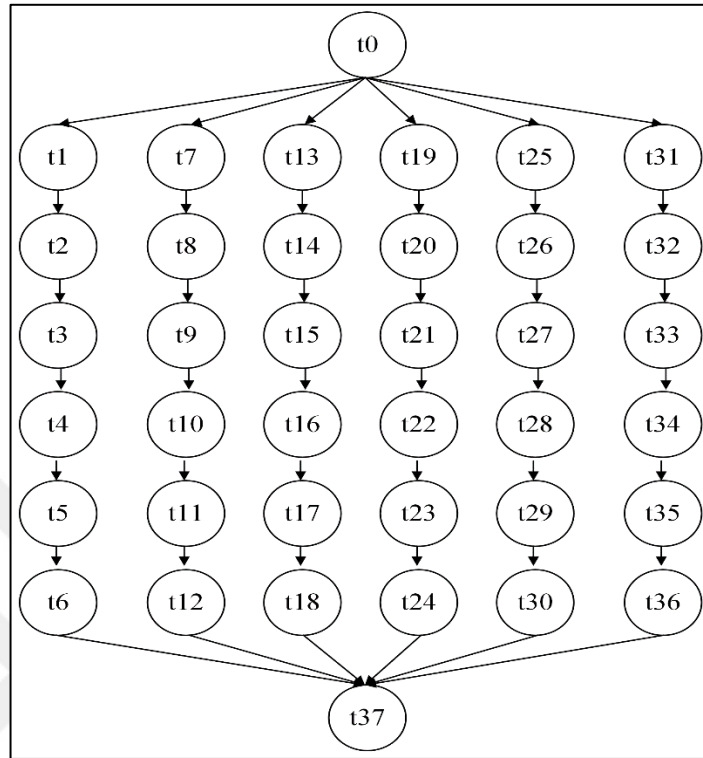


Şekil 4.20: Robot Kontrol için rastgele ve permütasyon çözümleri ile makespan ve SLR karşılaştırması.

4.3.3. Yeni Nesil Dizileme İş Akış Çizgesi

Yeni nesil dizileme analizi iş akışındaki işlem adımları üçüncü bölümde açıklanmıştır. Bölüm 3.1’de işlemlerin çalışma zamanı modellenmiştir. Deneysel çalışmalarda 8 ve 20 adet iş akışına sahip veri kümeleri çalışma zamanı modeline göre üretilmiştir. Başlangıçta her bir iş akışının girdi dosyası olan FASTQ dosyası için rastgele dosya boyutları belirlenmiştir. Belirlenen her dosya boyutu için bir iş akışındaki her işlem için çalışma zamanı tahminlemesi yapılmıştır. Veri paralel işlem olan hizalama için paralellik değeri Tablo 3.1’de verilen F_{est} değeri (0.960558) kullanılmıştır. Şekil 4.21’de statik çizelgeleme için YND iş akışlarının 0 anında geldiği varsayılarak oluşturulmuş yönlü döngüsüz çizge bulunmaktadır. Bu çizge altı iş akışından oluşmaktadır. İş akışlarının başlangıç işlemleri sırayla t1, t7, t13, t19, t25, t31’dir. Bir iş akışı Şekil 4.21’de görüldüğü üzere veri bağımlılığı (kenar bağlantısı) bulunan işlemler ile ifade edilir. Örneğin t1 işlemi ile başlayan iş akışı, sırayla t2, t3, t4, t5 ve son işlem olan t6 ile sonlanır. Başlangıçta birbirinden bağımsız olan bu altı iş

akışı çalışma zamanı maliyeti sıfır olan başlangıç (t0) ve bitiş düğümleriyle (t37) bağlanarak yönlü döngüsüz çizge oluşturur.

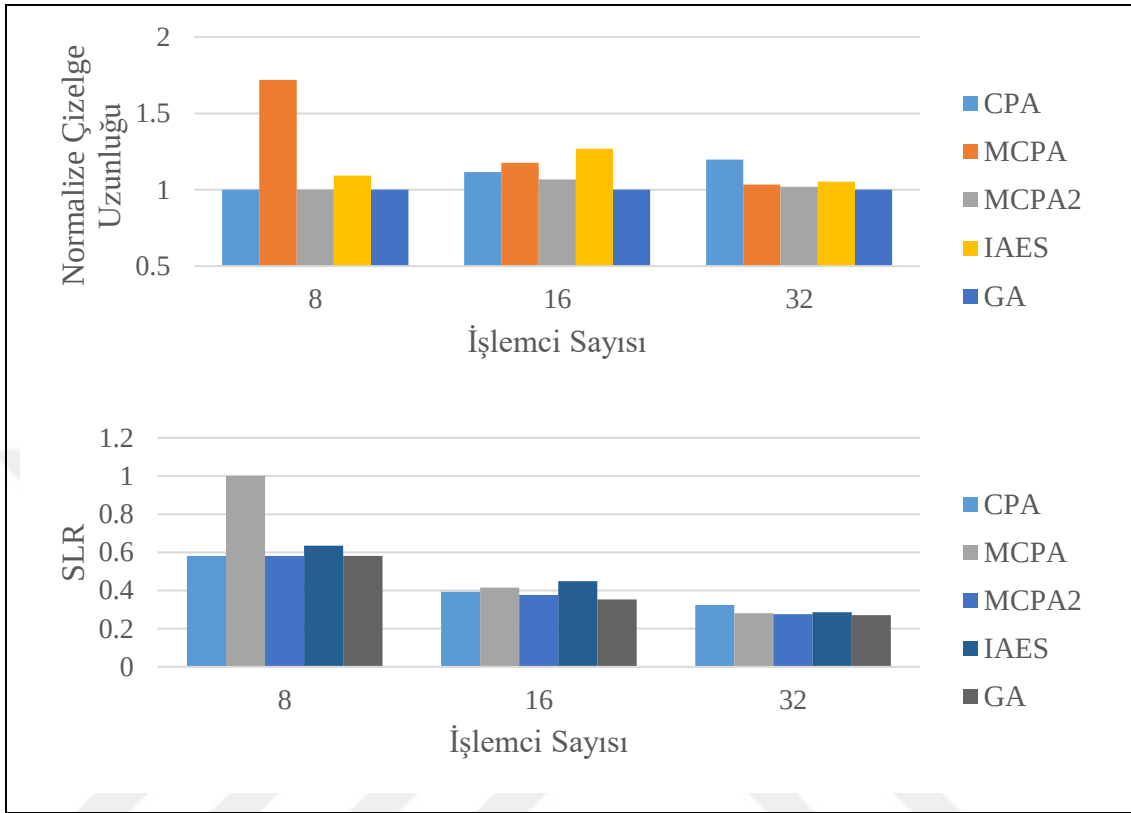


Şekil 4.21: YND analizi iş akışı yönlü döngüsüz çizgesi.

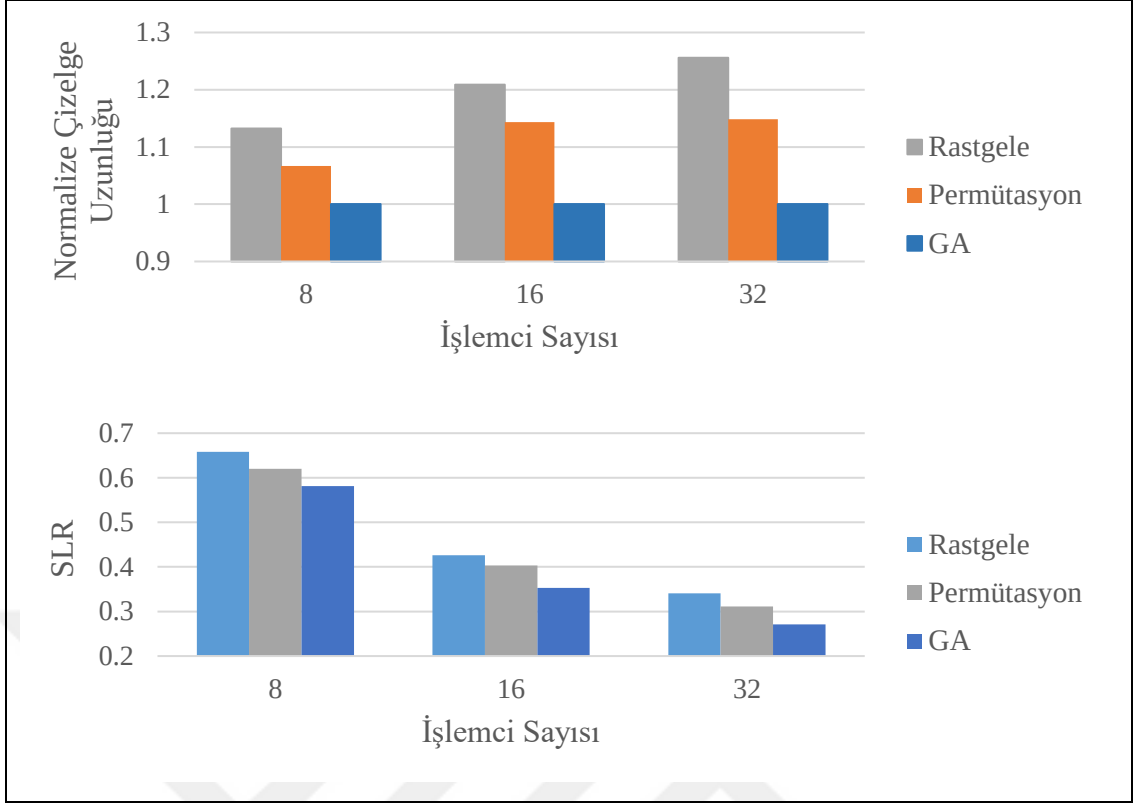
MCPA algoritmasında belirli bir katmanda bulunan veri paralel işlemler için tahsis edilen işlemci sayısı en fazla işlemci kaynağının işlemci kapasitesi kadar olmaktadır. Şekil 4.22’de 8 yeni nesil dizileme analizi iş akışına sahip 48 işlem içeren çizge için sonuçlar görülmektedir. Çizgede ikinci katmanda bulunan 8 veri paralel işlem için işlemci kaynağında 8 işlemci olması durumunda tüm işlemlere birer işlemci atanabilmektedir. Bu durum MCPA algoritmasının diğer algoritmalara göre daha kötü performans göstermesine neden olmuştur. İşlemci sayısının artmasıyla MCPA algoritmasının performansının iyileştiği görülmektedir. Önerilen hiper-sezgisel algoritma ise tüm işlemci sayılarında en iyi performansı göstermiştir.

Şekil 4.23’ de önerilen hiper-sezgisel algoritmanın, rastgele seçilmiş sezgiseller ile işlemci tahsisi çözümünün ve her işlem için tekrarsız sezgisel permütasyonu ile oluşturulan çözümün makespan ve SLR performans metriğine göre karşılaştırması verilmiştir. Önerilen algoritma, rastgele sezgiseller ile oluşturulmuş çözüme göre

ortalama %19, sezgisellerin permütasyonuna göre oluşturulmuş çözüme göre ise %12 daha iyi sonuçlar vermiştir.

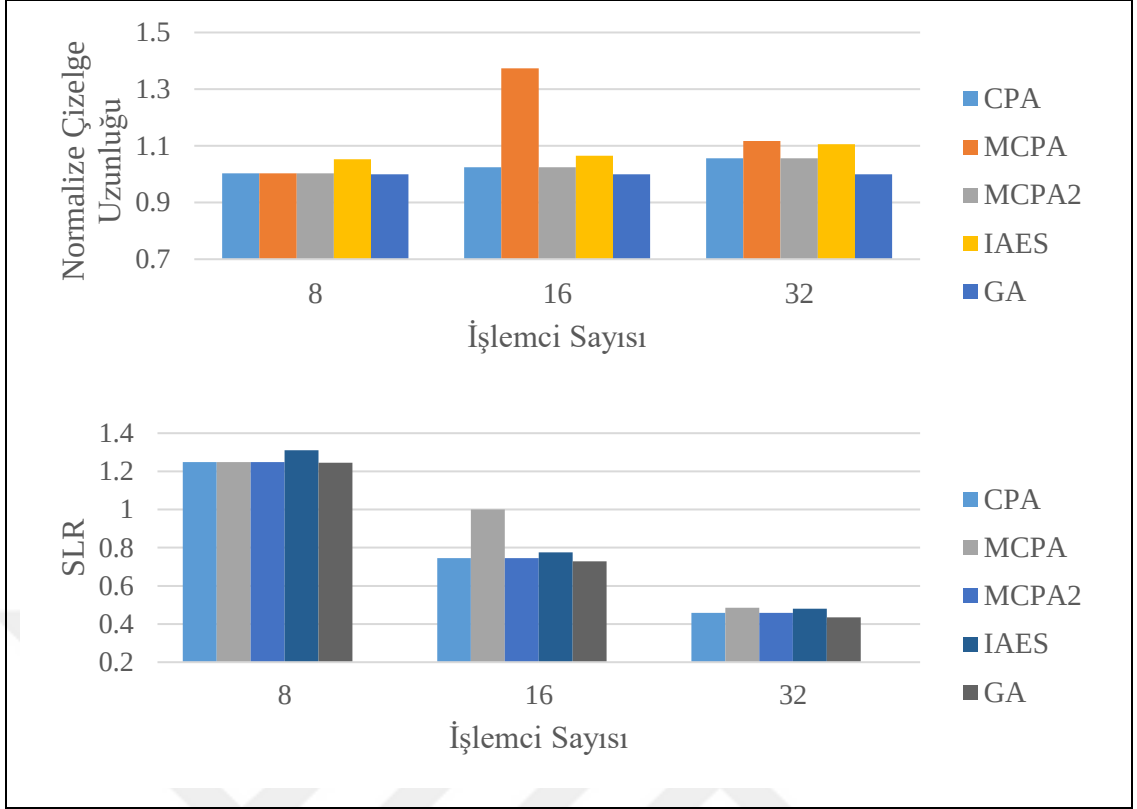


Şekil 4.22: YND analizi 8 iş akışı için açgözlü algoritmalar ile makespan ve SLR karşılaştırması.



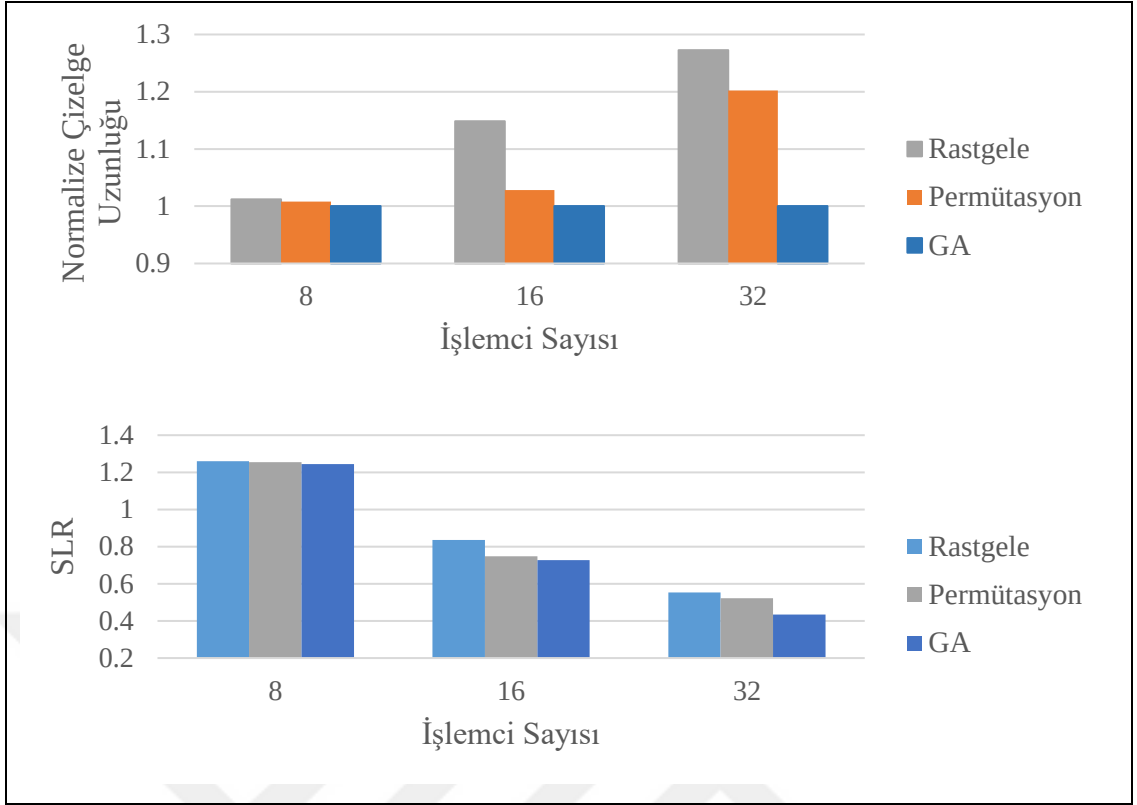
Şekil 4.23: YND analizi 8 iş akışı için rastgele ve permütasyon çözümleri ile makespan ve SLR karşılaştırması.

Şekil 4.24’ de 20 YND biyoinformatik iş akışına sahip 160 işlem içeren çizge için sonuçlar görülmektedir. İş akışı artıkça işlemci sayısı az olan işlemci kaynağında algoritmaların daha kötü performans gösterdiği SLR değerlerinden görülmektedir. İşlemci sayısı 16 olduğunda MCPA haricindeki algoritmaların performans değerleri iyileşir. MCPA algoritması belirli bir katmandaki işlemlere toplam işlemci sayısı kadar işlemci tahsis etmesinden dolayı performans iyileştirmesi diğer algoritmalara göre daha azdır. İşlemci kaynağındaki işlemci sayısı 32 olduğunda ise tüm algoritmalar daha iyi sonuçlar vermeye devam eder. Tüm koşullarda diğer algoritmalar ile karşılaştırıldığında en iyi sonuç veren hiper-sezgisel algoritmadır.



Şekil 4.24: YND analizi 20 iş akışı için ağgözlü algoritmalar ile makespan ve SLR karşılaştırması.

Şekil 4.25’ de 20 YND biyoinformatik iş akışına sahip 160 işlem içeren çizge için önerilen hiper-sezgisel algoritmanın, rastgele seçilmiş sezgiseller ile işlemci tahsisi çözümünün ve her işlem için tekrarsız sezgisel permütasyonu ile oluşturulan çözümün makespan ve SLR performans metriğine göre karşılaştırması verilmiştir. Önerilen algoritma, rastgele sezgiseller ile oluşturulmuş çözüme göre ortalama %14, sezgisellerin permütasyonuna göre oluşturulmuş çözüme göre ise %8 daha iyi sonuçlar vermiştir.

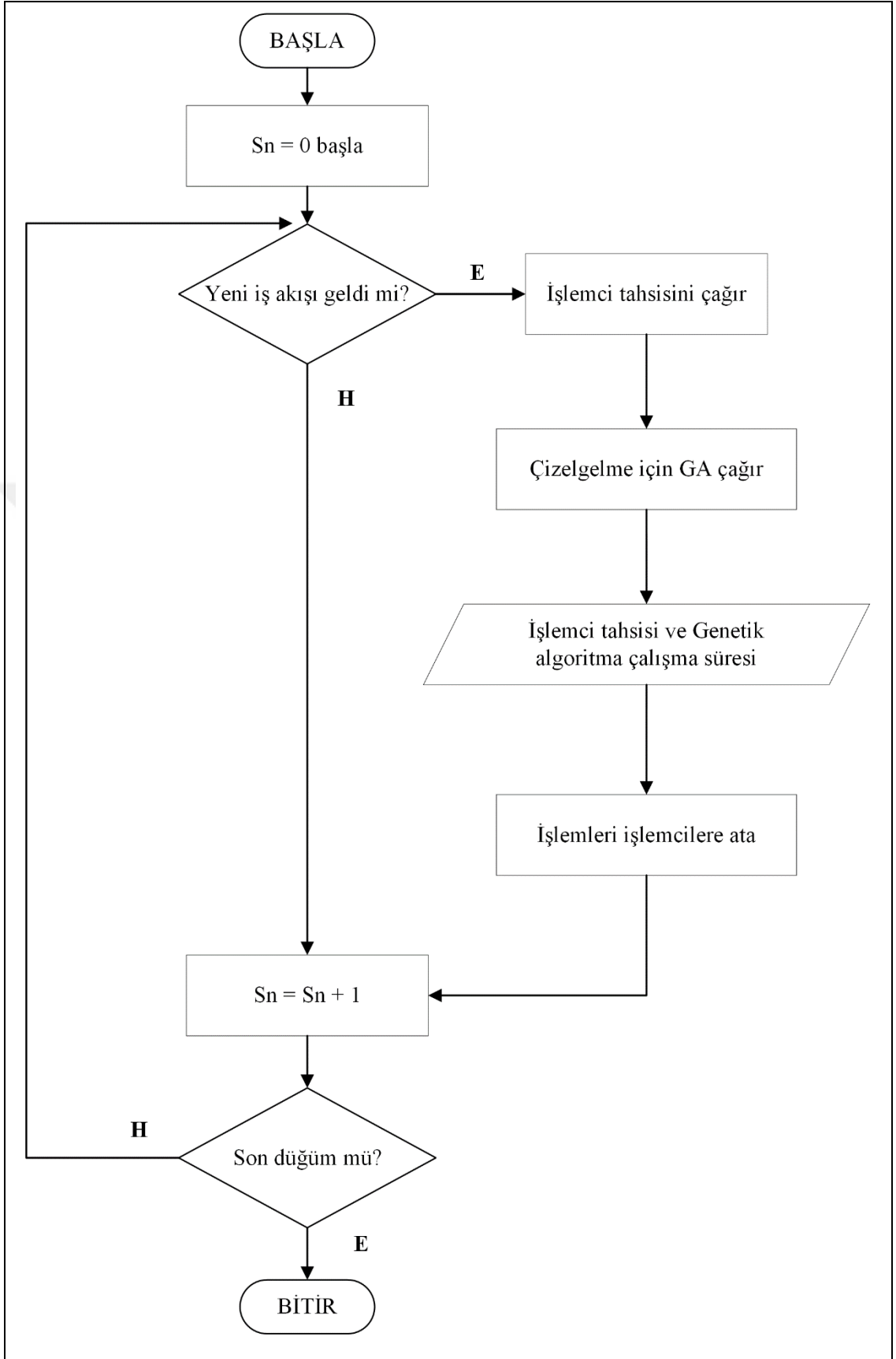


Şekil 4.25: YND analizi 20 iş akışı için rastgele ve permütasyon çözümleri ile makespan ve SLR karşılaştırması.

5. GELİŞTİRİLEN DİNAMİK ÇİZELGELEME MODELİ

Bu tez çalışmasında veri paralel işlemler içeren dinamik çizelgeleme problemi için işlemci tahsisinde Bölüm 4’de işlemci tahsisi için geliştirilen hiper-sezgisel yöntem kullanılmıştır. Çizelgeleme süreci için meta-sezgisel bir yöntem olarak genetik algoritma geliştirilmiştir. Bölüm 5.1 ‘de geliştirilen genetik algoritmanın detayları verilmiştir. Dinamik çizelgeleme problemlerinde veri akışlarının sisteme varış zamanı bilinmez. Geliştirilen algorithmada varış zamanına bağlı olarak işlemci tahsisi ve çizelgeleme algoritmalarının çözüm üretmeleri planlanmıştır. Başlangıçta sistemde var olan iş akışlarının işlemleri için işlemci tahsisi ve çizelgeleme prosedürleri yürütülür. Algoritmaların çalışma süresi maliyeti bulunmaktadır. Bu maliyet çizelgelemenin maksimum tamamlama süresine eklenir. Bu eklemenin yapılması için sistemde bir işlemci üzerinde işlemci tahsisi ve çizelgeleme prosedürlerinin yürütüldüğü varsayılır. Algoritma çizelgeleme sonucu verene kadar işlemci kaynağındaki diğer işlemciler boştur. Çizelgeleme sıralamasının belirlenmesi ile işlemlerin atanacağı işlemciler belirlenecek ve yürütülmeye başlayacaktır. Çizelgeleme sonucunda algoritmanın çalışma süresi maksimum tamamlanma süresine eklenmiş olur. Önceden bilinmeyen bir zaman (S_n)’da yeni bir iş akışı sisteme vardığında henüz yürütülmeye başlamamış olan işlemler ve yeni gelen iş akışının işlemleri için işlemci tahsisi ve çizelgeleme algoritmaları tekrar yürütülerek yeni çizelgeleme sıralaması belirlenir.

Yeni iş akışı sisteme vardığında henüz çalışması tamamlanmamış işlemler bulunabilir. Bu durum yeni çizelgeleme çözümü üretilirken çalışan işlemler tamamlanana kadar dolu işlemcilerin kullanılamayacağını gösterir. Genetik algoritma yeni çizelgeleme sıralaması üretirken amaç fonksiyonunda uygun olmayan işlemcilere (önceki çizelgelemede işlemcilere atanmış işlemler tamamlanmamış) işlem ataması yapmaz. Şekil 5.1 ‘de dinamik çizelgeleme için akış şeması verilmiştir.



Şekil 5.1: Dinamik Çizelgeleme için önerilen akış şeması.

5.1. Genetik Algoritma Kullanımı

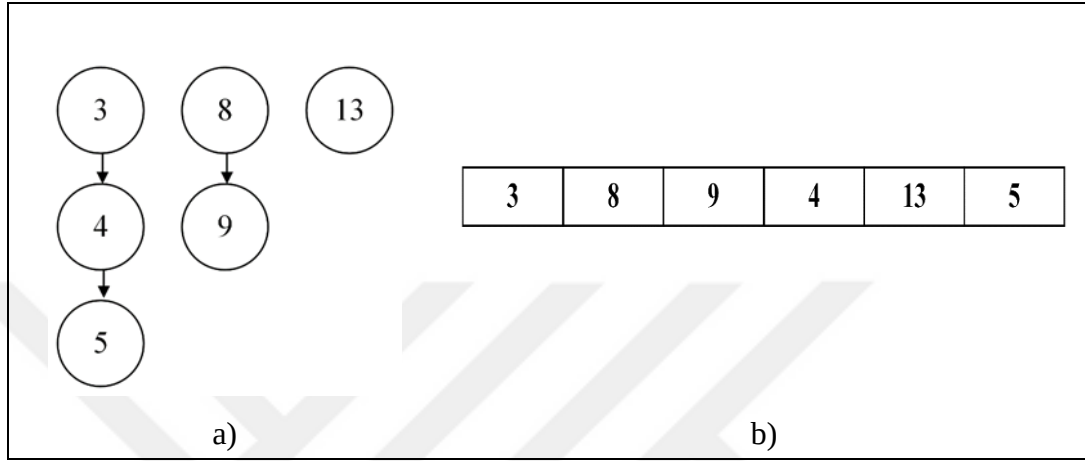
Dinamik çizelgeleme probleminin çizelgeleme aşaması için meta-sezgisel bir yöntem olarak genetik algoritma uygulanmıştır. Şekil 5.2’de genetik algoritmanın sözde kodu verilmiştir.

#GenetikAlgoritma
chroNum: Kromozom Sayısı mutPro: Mutasyon Olasılığı crosPro: Çaprazlama Olasılığı itNum: İterasyon Sayısı = 100 dugModelList: Çizge Düğüm Listesi proList: İşlemci Listesi eniYi 1. populasyon =baslangicpopulasyonUret(chroNum) % Rassal olarak populasyonu üret 2. hesapla amacfonksiyonu (populasyon) 3. enIyi = min(amacfonksiyonu) 4. while (itNum < 100) 5. ebeveynler = seçilim (amacfonksiyonu, populasyon) 6. yavru = caprazlama (ebeveynler) 7. dahaİyiysePopulasyondaDegistir (yavru) 8. yavru = mutasyon (ebeveynler) 9. dahaİyiysePopulasyondaDegistir (yavru) 10. if (min(amacfonksiyonu) < eniyi) 11. eniyi = populasyon(bul(amacfonksiyonu==eniyi)) 12. end 13. end 14. bitir eniyi

Şekil 5.2: Genetik Algoritma Sözde Kodu.

Genetik algoritmada her bir kromozom bir çizelgeleme çözümünü ifade eder. Kromozomdaki genler bir işlemi göstermektedir. Genlerin sıralaması işlemlerin

işlemcilere atanma sıralamasını gösterir. Şekil 5.3 a)'da işlemler ve bağımlılıkları gösterilirken Şekil 5.3 b)'de ise 6 işlemi içeren kromozom örneği görülmektedir. İşlemlerin işlemcilere atanmasında öncelikle işlem 3 tahsis edilen işlemci sayısı kadar işlemci boş olduğunda işlemcilere atanacak ve yürütülecektir. Sonrasında sırayla 8, 9, 4, 13, 5 işlemleri uygun işlemlere atanacak ve yürütülecektir.



Şekil 5.3: a) İşlemler ve bağımlılıkları, b) Kromozom.

Başlangıç popülasyonu belirlenirken kromozomlar işlem bağımlılıklarına göre oluşturulur. Öncül işlemi olan bir işlem, kromozomda öncül işlemini gösteren genden önce yer alamaz. Bir kromozom oluşturulurken öncelikle öncül işlemi olmayan işlem veya işlemler arasında rastgele seçilen işlem ilk gende gösterilir. Sonrasında ise öncül işlemleri kromozomda bir gen ile ifade edilmiş ya da öncül işlemi olmayan genler arasından rastgele belirlenen her işlem sırayla kromozomu oluşturur. Örneğin Şekil 5.2.a)'da işlemler için kromozom oluşturulmak istendiğinde öncelikle herhangi bir işleme bağımlılığı olmayan 3, 8, 13 işlemleri arasından rastgele belirlenen işlem birinci gen ile ifade edilir. Birinci gen için rastgele işlem 3'ün seçildiğini varsayarsak ikinci işlem için artık 4, 8, 13 işlemleri arasından rastgele belirlenen işlem ikinci gen ile ifade edilecektir. Başlangıç popülasyonuna bir önceki çizelgelemede üretilen çözüm varsa eklenir. Bu çözümdeki yürütülmüş ya da yürütülmeye başlamış olan işlemler çözüm içerisinde çıkarılır. Yeni gelen işlemler ise çözümün kromozomunun sonundan başlayarak veri bağımlılıkları dikkate alınarak rastgele sıralamada eklenir.

Kromozomun amaç fonksiyonu çizelge uzunluğunun hesaplamasıdır. Ebeveyn kromozomların seçilimi için rulet tekerleği tekniği kullanılır.

5.1.1. Çaprazlama

Çaprazlama işleminde çocuk kromozomlarda da işlem bağımlılıklarına dikkat edilmelidir. Seçilen iki ata kromozomun çaprazlanması için rastgele üretilmiş sayı kromozomda kaçınıcı genden sonra değişimin gerçekleşeceğini gösterir. Şekil 5.4’de ata kromozomda kaçınıcı genden itibaren kesileceği rastgele bir sayı ile belirlenmiştir. Kromozom belirlenen üçüncü genden itibaren kesilir. Birinci ve ikinci ata kromozomun başlangıçtaki 3 geni çocuk kromozomlara devredilir. Birinci çocuk kromozom için kalan 3 genin belirlenmesi için ikinci ata kromozomdaki genler sırayla kontrol edilir. Örneğin ikinci ata kromozomdaki işlem 13 çocuk kromozomun birinci ata kromozomdan devredilen ilk 3 geninde yoktur. İşlem 13 birinci çocuk kromozoma eklenir. İşlem 8 vardır eklenmez. Kalan genlerde sırayla kontrol edilerek çocuk kromozoma eklenir veya eklenmez. İkinci çocuk kromozom oluşturulması işlemide birinci ata kromozomun genlerinin sırayla kontrol edilmesi ile yapılır. Yapılan bu kontrollü eklemeler işlem bağımlılığı kısıtının da göz önünde bulundurularak yeni çocuk kromozomların oluşturulmasını sağlar.

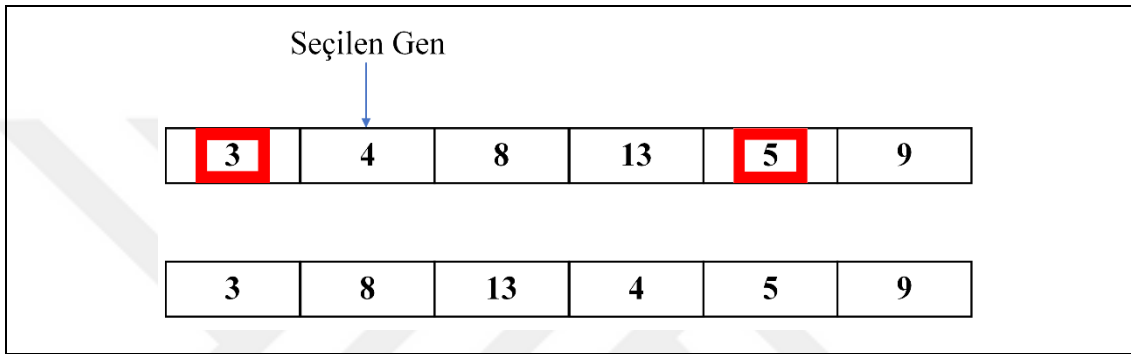
Rastgele Belirlenen Nokta						
1. Ata	3	8	9	4	13	5
2. Ata	13	8	3	9	4	5
1.Çocuk	3	8	9	4	5	13
2.Çocuk	13	8	3	4	5	9

Şekil 5.4: Çaprazlama.

5.1.2. Mutasyon

Mutasyon işleminde çocuk kromozomlardan yeni kromozomlar oluşturulur. Kromozomda rastgele bir gen belirlenir. Belirlenen genin temsil ettiği işlemin öncül ve ardıl işlemleri varsa bu işlemleri temsil eden genlerin yerleri belirlenir. Bu genlerin

arasında bulunan bir gen rastgele belirlenir. Seçilen gen rastgele belirlenen noktaya getirilerek diğer genler kaydırılır. Şekil 5.5’de mutasyon ile oluşturulmuş yeni kromozom örneği verilmiştir. Çocuk kromozomda rastgele seçilen gen ikinci gendir. İkinci gende işlem 4 bulunmaktadır. İşlem 4’ün öncül işlemi 3 ardıl işlemi ise 5’dir. İşlem 3 ve 5’in bulunduğu genlerin arasındaki genlerden işlem 4’ün bulunduğu gen hariç (işlem 8 ve 13’ün bulunduğu üçüncü ve dördüncü gen) rastgele bir nokta seçilir. Burada dördüncü gen seçilmiştir. İşlem 4 dördüncü gene yerleşirken 8 ve 13 işlemleri birer gen öne kaydırılmıştır.



Şekil 5.5: Mutasyon.

5.2. Dinamik Çizelgeleme Deneysel Sonuçları

Tez kapsamında geliştirilen dinamik çizelgeleme testleri 4 çekirdekli i7 2.70 Ghz işlemci, 8 GB ana bellek üzerinde Microsoft 10 işletim sisteminde koşturmaktadır. Geliştirilen algoritmalar C# programlama dili kullanılarak kodlanmıştır.

Algoritmamızda başlangıç popülasyonu sayısı 100, jenerasyon sayısı ise 30’dur. Algoritma, çaprazlama olasılığı 0.7 ve mutasyon olasılığı 0.05 ile yürütülmüştür. Tüm parametreler yapılan seri testler sonucu belirlenmiştir.

Bu tez çalışmasında dinamik çizelgeleme problemleri için işlemci tahsisinde CPA sezgiseli, IAES sezgiseli ve hiper-sezgisel yöntem uygulanmış çizelgelemede ise geliştirilen dinamik genetik algoritma kullanılarak üç farklı algoritma ile test edilmiştir. Dinamik çizelgeleme problemi için geliştirilmiş ağgözlü algoritma MOWS ile işlemci tahsisinde CPA, IAES ve hiper-sezgisel yöntemi uygulayan dinamik genetik algoritma sonuçları karşılaştırılmıştır. Önerilen hiper-sezgisel yöntemde CPA, MCPA, MCPA2, IAES işlemci tahsisi sezgiselleri kullanılmıştır. Problemden 8, 16, 32

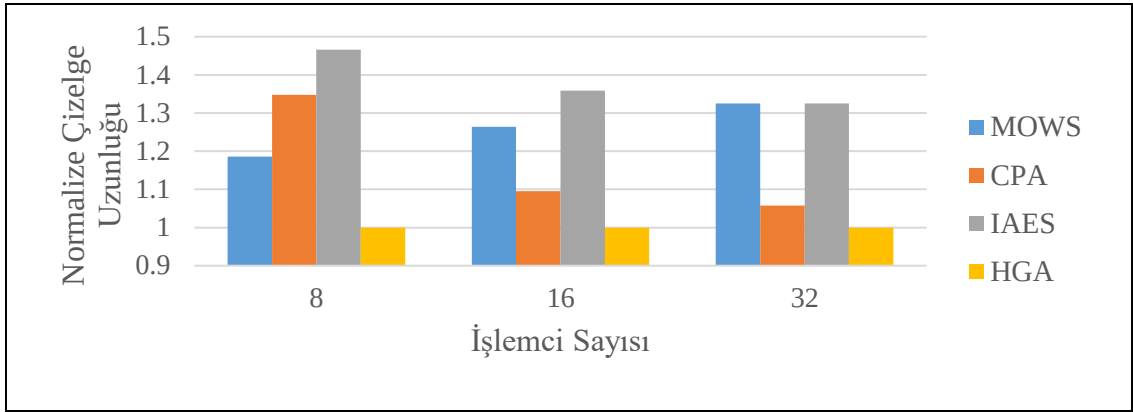
gibi deęişen işlemci sayısına sahip işlemci kaynaklarının tüm işlemcilerinin aynı kapasiteye sahip olduęu (homojen işlemci kaynakları) varsayılır.

Dinamik çizelgeleme algoritmasında çizelgeleme sürecinde algoritmanın çalışma süresi de maksimum tamamlanma süresine dahil edilmiştir. Deneysel çalışmalarda çizelgeleme algoritmasının yürütme süresi dahil performans karşılaştırılması yapılmıştır. Karşılaştırılması yapılan dinamik açgözlü algoritmanın (MOWS) tamamlanması ile çalışma zamanı maksimum tamamlanma süresine eklendi.

Deneysel çalışmalarda standart işlem çizgeleri, gerçek dünya uygulamaları ve yeni nesil dizileme analiz iş akışı kullanılmıştır. Standart işlem çizge ve gerçek dünya uygulamalarının veri paralel işlemlerinin çalışma süresinin hesaplanmasında Amdahl yasası kullanılmıştır. Veri paralel işlemler için paralellik değeri 0.9 olarak belirlenmiştir. Yeni nesil dizileme iş akışındaki veri paralel işlem olan hizalama için ise Bölüm 3.1 ve Tablo 3.1’de verilen F_{est} değeri (0.960558) kullanılmıştır.

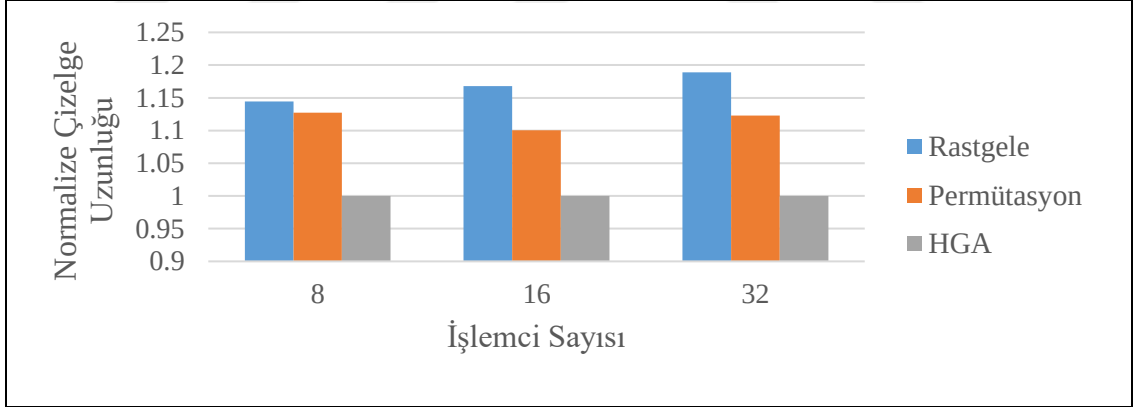
5.2.1. Yeni Nesil Dizileme İş Akış Çizgesi

Deneysel çalışmalar 8 ve 20 adet YND iş akışına sahip veri kümeleri çalışma zamanı modeline göre üretilmiştir. İş akışları için rastgele varış zamanları belirlenmiştir. Şekil 5.6’de 8 iş akışının normalize edilmiş çizelgeleme sonuçlarının maksimum tamamlanma süresi (makespan) karşılaştırması verilmiştir. Normalize edilmiş değeri her bir algoritmanın maksimum tamamlanma süresinin hiper-sezgisel algoritma ile elde edilen maksimum tamamlanma süresine oranıdır. İş akışlarında dört tanesinin 0 anından farklı zamanlarda sisteme geldiği varsayılır. Önerilen algoritmada işlemci tahsisinde CPA sezgiseli, IAES sezgiseli ve hiper-sezgisel genetik algoritma kullanılmıştır. Şekil 5.6’de HGA, hiper-sezgisel işlemci tahsisi ile dinamik çizelgeleme sonucunu gösterir. HGA algoritması diğer dinamik çizelgeleme için üretilmiş MOWS açgözlü algoritması ve CPA, IAES işlemci tahsisi ile çizelgelemede önerilen dinamik algoritmayı kullanan algoritmalarla göre en iyi sonucu vermiştir. CPA sezgiseli veya IAES sezgiseli ile işlemci tahsisi yapıldığında 8 işlemcili kaynaktaki MOWS açgözlü algoritmasına göre daha iyi performans gösterememiştir. Bu durum işlemci tahsisinin maksimum tamamlanma süresi üzerindeki etkisini göstermektedir. İşlemci sayısı artıkça CPA ve IAES sezgiseli ile geliştirilen çözümün HGA çözümüne göre performansı iyileşirken MOWS algoritmasının performansı azalmıştır.



Şekil 5.6: YND analizi 8 iş akışı için ağgözlü algoritmalar ile makespan karşılaştırması.

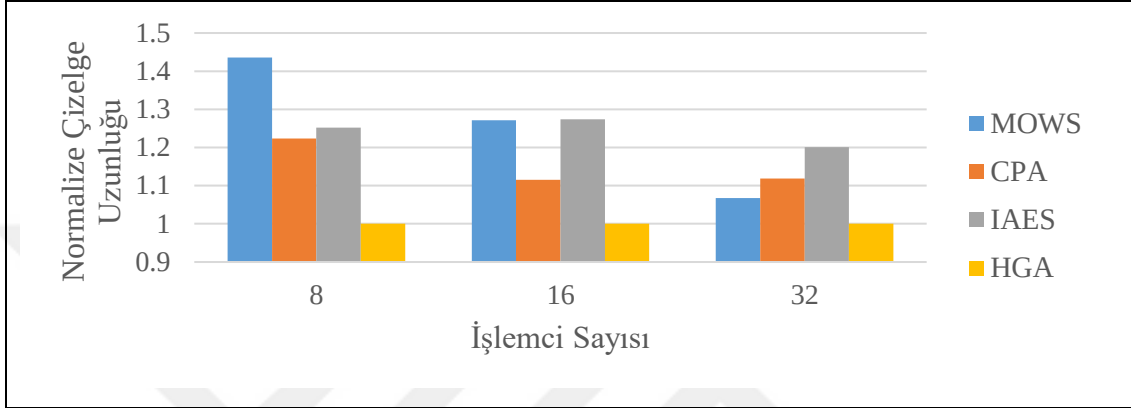
Şekil 5.7’de HGA algoritmasının, rastgele seçilmiş sezgiseller ile işlemci tahsisi çözümünün ve her işlem için tekrarsız sezgisel permütasyonu ile oluşturulan çözümün makespan karşılaştırması verilmiştir. HGA, rastgele sezgiseller ile oluşturulmuş çözüme göre ortalama %16, sezgisellerin permütasyonuna göre oluşturulmuş çözüme göre ise %11 daha iyi sonuçlar vermiştir.



Şekil 5.7: YND analizi 8 iş akışı için rastgele ve permütasyon çözümleri ile makespan karşılaştırması.

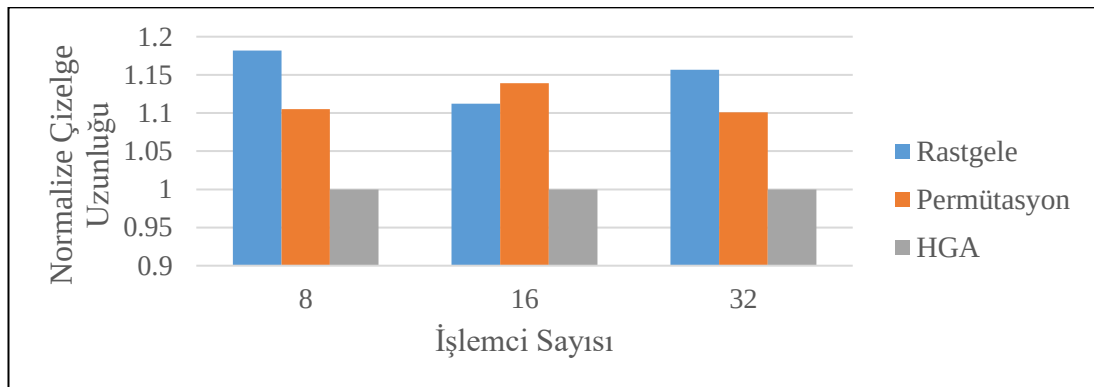
Şekil 5.8’de 20 iş akışının çizelgeleme sonuçlarının maksimum tamamlanma süresi (makespan) karşılaştırması verilmiştir. İş akışlarından on tanesi 0 anından farklı varış zamanına sahiptir. Bu iş akışlarından 0 anından farklı zamanda sisteme gelen 3 iş akışı aynı anda gelmektedir. Herhangi bir iş akışı sisteme vardığında yeni iş akışının işlemleri ve henüz yürütülmemiş işlemler için önerilen algoritmada işlemci tahsisi ve çizelgeleme adımları gerçekleşir. CPA işlemci tahsisi sezgiseli gibi bazı sezgiseller işlemci kaynağında paralel işlemlere dağıtılacak yeterli işlemci olmaması durumunda

işlemlere tahsis edilecek işlemci sayısını kısıtlar. Bu durum işlemci kaynağındaki işlemci sayısına göre çizelgelemede daha kötü performans göstermelerine neden olabilir. Şekil 5.8’de HGA algoritmasının tüm koşullarda en iyi sonuç verdiği görülmektedir. Hiper-sezgisel işlemci tahsisinin kullandığı sezgisellerin güçlü yanlarını kullanabiliyor olması tüm koşullarda önerilen dinamik algoritmanın da iyi sonuçlar vermesini sağlamıştır.



Şekil 5.8: YND analizi 20 iş akışı için açgözlü algoritmalar ile makespan karşılaştırması.

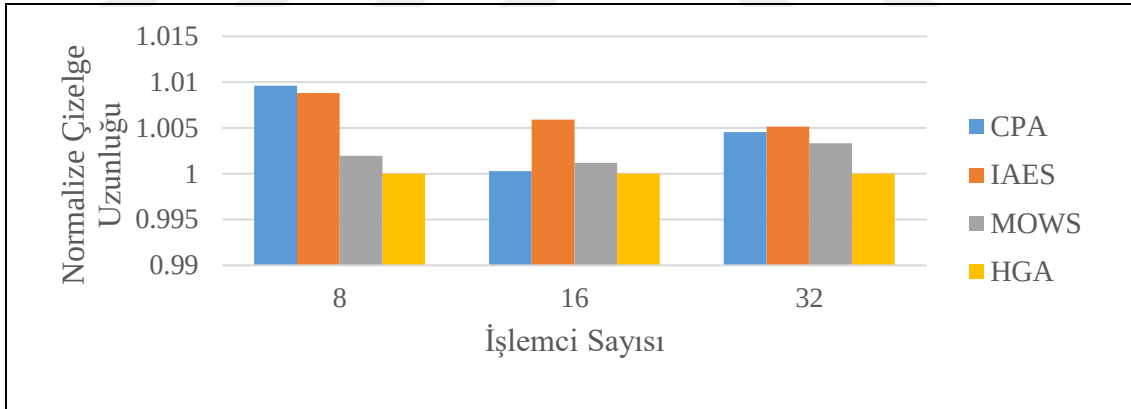
Şekil 5.9’ de 20 iş akışı için HGA algoritmasının, rastgele seçilmiş sezgiseller ile işlemci tahsisi çözümünün ve her işlem için tekrarsız sezgisel permütasyonu ile oluşturulan çözümün makespan karşılaştırması verilmiştir. HGA, rastgele sezgiseller ile oluşturulmuş çözüme göre ortalama %15, sezgisellerin permütasyonuna göre oluşturulmuş çözüme göre ise %11 daha iyi sonuçlar vermiştir.



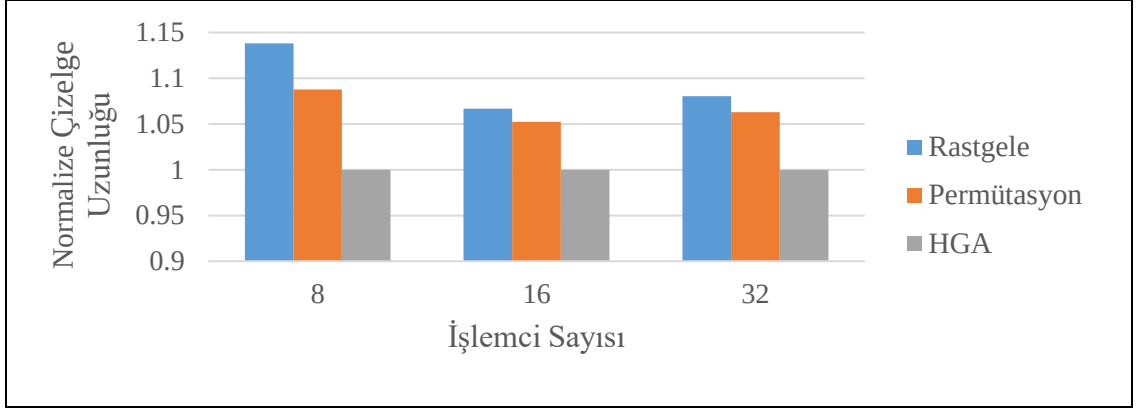
Şekil 5.9: YND analizi 20 iş akışı için rastgele ve permütasyon çözümleri ile makespan karşılaştırması.

5.2.2. Standart İşlem Çizgeleri

Dinamik çizelgeleme problemleri için önerilen dinamik algoritmanın performansını ölçmek amacıyla standart işlem çizgeleri (STG) kullanılmıştır. 50 işlem içeren 5 çizgenin sisteme farklı zamanlarda geldiği veri kümesi oluşturulmuştur [67], [68]. İş akışlarında tüm işlemler veri paralel işlemlerdir. İş akışlarından dördünün sisteme varış zamanı farklıdır. Seçilen 5 iş akışından (STG) 2 tanesi birinci metotla üretilirken 3 tanesi ise ikinci metotla üretilmiştir. Şekil 5.10'de algoritmaların performans karşılaştırmaları verilmiştir. HGA algoritması diğer dinamik çizelgeleme için üretilmiş MOWS açgözlü algoritması ve CPA, IAES işlemci tahsisi yapılmış geliştirilen dinamik algoritmalarla göre farklı işlemci sayısına sahip kaynaklarında en iyi sonucu vermiştir. Şekil 5.11' de HGA algoritmasının, rastgele seçilmiş sezgiseller ile işlemci tahsisi çözümünün ve her işlem için tekrarsız sezgisel permütasyonu ile oluşturulan çözümün makespan karşılaştırması verilmiştir. HGA, rastgele sezgiseller ile oluşturulmuş çözüme göre ortalama %9, sezgisellerin permütasyonuna göre oluşturulmuş çözüme göre ise %7 daha iyi sonuçlar vermiştir.



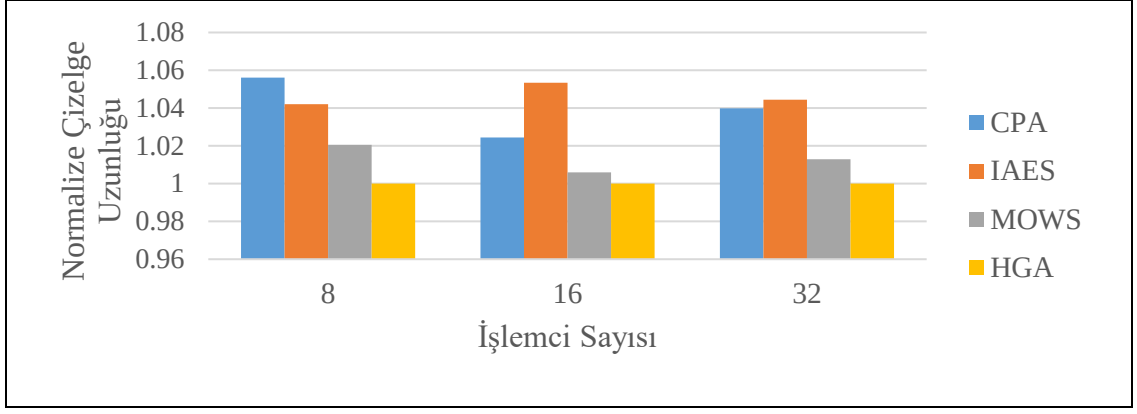
Şekil 5.10: 50 işlem içeren 5 iş akışı için açgözlü algoritmalar ile makespan karşılaştırması.



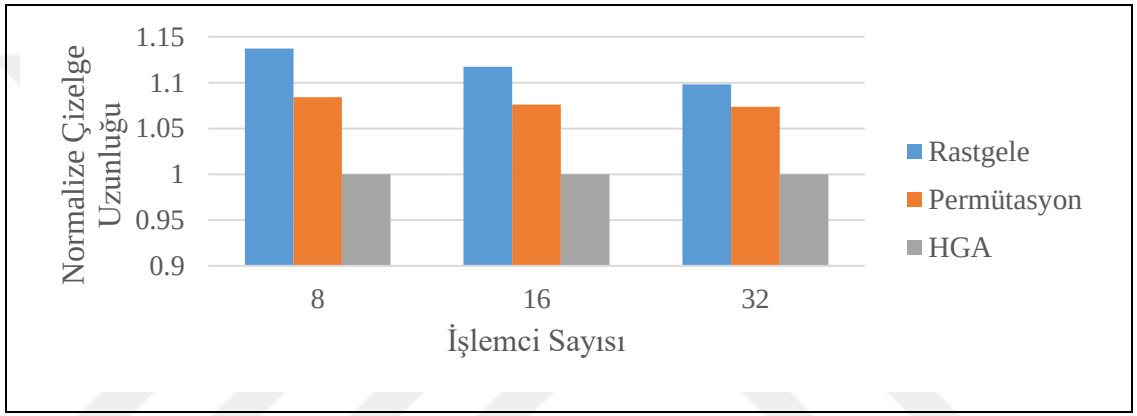
Şekil 5.11: 50 işlem içeren 5 iş akışı için rastgele ve permütasyon çözümleri ile makespan karşılaştırması.

5.2.3. Uygulama Çizgeleri

Geliştirilen dinamik çizelgeleme algoritmasının performansını ölçmek amacıyla statik çizelgeleme deneysel çalışmalarında performans karşılaştırılmasında kullanılan gerçek dünya uygulamaları kullanılmıştır. Kullanılan gerçek dünya uygulamaları Matmul, Strassen, Spec fppp ve Robot Kontrol'dür. Bu dört uygulamadan ikisinin sisteme farklı zamanlarda geldiği varsayılarak dinamik çizelgeleme algoritması yürütülür. Şekil 5.12'de HGA algoritmasının diğer dinamik çizelgeleme için üretilmiş MOWS açgözlü algoritması ve CPA, IAES ile işlemci tahsisi yapılmış önerilen dinamik algoritmalarla göre farklı işlemci sayısına sahip kaynaklarda en iyi sonucu verdiği görülmektedir. Şekil 5.13' de HGA algoritmasının, rastgele seçilmiş sezgiseller ile işlemci tahsisi çözümünün ve her işlem için tekrarsız sezgisel permütasyonu ile oluşturulan çözümün makespan karşılaştırması verilmiştir. HGA, rastgele sezgiseller ile oluşturulmuş çözüme göre ortalama %9, sezgisellerin permütasyonuna göre oluşturulmuş çözüme göre ise %7 daha iyi sonuçlar vermiştir.



Şekil 5.12: 4 uygulama için ağgözlü algoritmalar ile makespan karşılaştırması.



Şekil 5.13: 4 uygulama için rastgele ve permütasyon çözümleri ile makespan karşılaştırması.

6. SONUÇLAR

Bu tez çalışmasında veri paralel işlem içeren paralel işlemlere sahip uygulamalar için işlemci tahsisinde hiper-sezgisel yöntem uygulanmıştır. Literatürdeki çalışmalarda önerilen algoritmalar çizge yapısına bağlı olarak performans farklılıkları göstermektedir. Hiper-sezgisel yöntem, bu algoritmaların işlemci tahsisi sezgiselini kullanarak algoritmaların güçlü yönlerini kullanır. Diğer algoritmalar çizge yapısına göre performans farklılıkları gösterirken hiper-sezgisel yöntem tüm çizgelerde en iyi sonuçları verir.

Veri paralel işlemler içeren iş akışlarının dinamik çizelgeleme problemi üzerine literatürde yapılan çalışmalar kısıtlıdır. Bu çalışmaların büyük bir kısmı çizelgeleme aşamasına gelmeden işlemlere tahsis edilmiş işlemci sayının belirlendiğini kabul eder. Veri paralel işlemleri katı işlemlerdir. Bu tez çalışmasında ise veri paralel işlemler kalıplanabilir işlemlerdir. Kalıplanabilir işlemlerin işlemci tahsisi çizelgeleme sürecinde belirlenir. Deneysel çalışmalarda önerilen dinamik algoritmada işlemci tahsisinde CPA sezgiseli, IAES sezgiseli ve hiper-sezgisel işlemci tahsisi kullanılarak karşılaştırma yapılmıştır. Dinamik çizelgeleme problemine çözüm olarak geliştirilen açgözlü algoritma MOWS ile karşılaştırılmıştır. Önerilen dinamik algoritmada hiper-sezgisel işlemci tahsisi ile elde edilen sonuçlar diğer çözümlere göre yüksek performans göstermiştir.

Deneysel çalışmalarda önerilen algoritmalar açgözlü algoritmalar haricinde rastgele seçilmiş sezgiseller ile işlemci tahsisi çözümleri ve her işlem için tekrarsız sezgisel permütasyonu ile oluşturulan çözümler ile de karşılaştırılmıştır. Önerilen algoritmalar diğer çözümlere göre yüksek performans göstermiştir.

Tez çalışmasının deneysel çalışmaları kapsamında, yeni nesil dizileme analizi iş akışı işlemleri için çalışma zamanı tahminleme modeli oluşturulmuştur. Veri paralel işlem içeren iş akışında paralellik değerinin tahminlemesi için Amdahl Yasası ve Karp-Flatt metriği uygulanmıştır. Deneysel çalışmalarda aynı problem türüne çözüm geliştiren algoritmalar ile geliştirilen algoritmaların performans karşılaştırması yapılabilmesi için literatürdeki çalışmaların deneysel çalışmalarında kullanılan Standart işlem çizge (STG) ve gerçek dünya uygulamaları kullanılmıştır. Geliştirilen özgün algoritmaların tüm çizgelerde yüksek performans gösterdikleri yapılan deneysel çalışmalar ile saptanmıştır.

Dinamik çizelgeleme algoritmasında çizelgeleme sürecine algoritmanın çalışma süresi de eklenmiştir. Algoritmanın işlemci kaynaklarında yürütüldüğü varsayılmış ve maksimum tamamlanma süresine çalışma süresi de dahil edilmiştir. Bu durum çizelgeleme algoritmasının yürütme süresi dahil performansının dinamik açgözlü algoritmaya göre daha iyi olduğunu göstermektedir. Ancak maksimum tamamlanma süresi haricinde değerlendirildiğinde dinamik açgözlü algoritma önerilen algoritmaya göre çok daha kısa süresinde çözüm üretmektedir. Önerilen algoritmanın çalışma süresi performansının iyileştirilmesi geliştirmeye açıktır. Bu problem gelecek çalışmalardan biridir.



KAYNAKLAR

- [1] Lenstra J. K., Kan A. H. G. R., (1978), "Complexity of Scheduling under Precedence Constraints", *Operations Research*, 26 (1), 22-35.
- [2] Radulescu A., Gemund A. J. C., (2001), "A low cost approach towards mixed task and data parallel scheduling", *International Conference on Parallel Processing*, 69-76, Valencia, Spain, 03-07 September.
- [3] Bansal S., Kumar P., Singh K., (2006), "An improved two-step algorithm for task and data parallel scheduling in distributed memory machines", *Parallel Computing*, 32 (10), 759-774.
- [4] Hunold S., (2010), "Low-Cost Tuning of Two-Step Algorithms for Scheduling Mixed-Parallel Applications onto Homogeneous Clusters", *10th IEEE/ACM International Conference on Cluster*, 253-262, Melbourne, Australia, 17-20 May.
- [5] Huang K. C., Wu W. Y., Wang F. J., (2016), "An iterative expanding and shrinking process for processor allocation in mixed-parallel workflow scheduling", *SpringerPlus*, 5 (1), 1138.
- [6] Radulescu A., Nicolescu C., Gemund A. J. C., Jonker P. P., (2001), "CPR: mixed task and data parallel scheduling for distributed systems", *15th international parallel and distributed processing symposium*, 9, San Francisco, USA, 23-27 April.
- [7] Hunold S., Lepping J., (2011), "Evolutionary Scheduling of Parallel Tasks Graphs onto Homogeneous Clusters", *2011 IEEE International Conference on Cluster Computing*, 344-352, Austin, TX, USA, 26-30 September.
- [8] Liu Y., Meng L., Taniguchi I., Tomiyama H., (2019), "A Genetic Algorithm for Scheduling of Dataparallel Tasks on Multicore Architectures", *IPSI Transactions on System LSI Design Methodology*, 12, 74-77.
- [9] Wang Y. R., Huang K. C. , Wang F. J., (2016), "Scheduling online mixed-parallel workflows of rigid tasks in heterogeneous multi-cluster environments", *Future Generation Computer Systems*, 60, 35-47.
- [10] Nath R., Nagaraju A., (2021), "Genetic algorithm based on-arrival task scheduling on distributed computing platform, *International Journal of Computers and Applications*", *International Journal of Computers and Applications*, 44 (107318), 887-896.
- [11] Amalarethinam G., Josphin A. M., (2020), "Dynamic Ant Colony Optimization (DyACO) in Heterogeneous Environments", *International Journal of Pure and Applied Mathematics*, 118 (6), 425-434.

- [12] Mahi M., Kodaz H., (2014), "Genetic Algorithm with Descendants Idea for Scheduling Tasks Graph in the MultiProcessor Architecture", *Journal of Advances in Computer Networks*, 2 (1), 10-13.
- [13] Zhang J., Luo J., Dong F., (2013), "Scheduling Parallel Task Graphs on non-dedicated heterogeneous multicluster platform with Moldable Task Duplication", *Proceedings of the 2013 IEEE 17th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, 313-318, Whistler, Canada, 27-29 June.
- [14] Liu Y., Taniguchi I., Tomiyama H., Meng L., (2013), "List Scheduling Strategies for Task Graphs with Data Parallelism", *2013 First International Symposium on Computing and Networking*, 168-172, Matsuyama, Japan, 04-06 December.
- [15] Ramaswamy S., Sapatnekar S., Banerjee P., (1997), "A framework for exploiting task and data parallelism on distributed memory multicomputers", *IEEE Transactions on Parallel and Distributed Systems*, 8 (11), 1098-1116.
- [16] Vydyanathan N., Krishnamoorthy S., Sabin G. M., Catalyurek U. V., Kurc T., Sadayappan P., Saltz J. H., (2009), "An Integrated Approach to Locality Conscious Processor Allocation and Scheduling of MixedParallel Applications", *IEEE Transactions on Parallel and Distributed Systems*, 20 (8), 1158-1172.
- [17] Hunold S., (2014), "One step toward bridging the gap between theory and practice in moldable task scheduling with precedence constraints", *Concurrency and Computation: Practice and Experience*, 27 (4), 1010 – 1026.
- [18] Boveiri H. R., (2016), "A Novel ACO-Based Static Task Scheduling Approach for Multiprocessor Environments", *International Journal of Computational Intelligence Systems*, 9 (5), 800-811.
- [19] Xiao J. P., Hu X. M., Chen W., (2020), "Dynamic Cloud Workflow Scheduling with a HeuristicBased Encoding Genetic Algorithm.", *International Conference on Neural Information Processing*, 38-49, Bangkok, Thailand, 23–27 November.
- [20] Hikida T., Nishikawa H., Tomiyama H., (2020), "Heuristic Algorithms for Dynamic Scheduling of Moldable Tasks", *2020 International SoC Design Conference (ISOCC)*, 55-56, Yeosu Korea (South), 21-24 October.
- [21] Hunold S., Rauber T., Runger G., (2007), "Dynamic scheduling of multiprocessor tasks on clusters of clusters", *2007 IEEE International Conference on Cluster Computing*, 507-514, Austin, TX, USA, 17-20 September.
- [22] Hunold S., Rauber T., Suter F., (2008), "Scheduling Dynamic Workflows onto Clusters of Clusters using Postponing", *2008 Eighth IEEE International Symposium on Cluster Computing and the Grid (CCGRID)*, 669-674, Lyon, France, 19-22 May.
- [22] Hunold S., Rauber T., Suter F., (2008), "Scheduling Dynamic Workflows onto Clusters of Clusters using Postponing", *2008 Eighth IEEE International*

Symposium on Cluster Computing and the Grid (CCGRID), 669-674, Lyon, France, 19-22 May.

- [23] Boregowda U., Chakravarthy V. R., (2014), "A Framework for Multiple Parallel Task Graphs (PTG) Scheduler", 2014 International Conference on Information Technology, 6-11, Bhubaneswar, India, 07-09 April.
- [24] Barbosa J. G., Moreira B., (2011), "Dynamic scheduling of a batch of parallel task jobs on heterogeneous clusters", *Parallel Computing*, 37 (8), 428-438.
- [25] Topcuoglu H., Hariri S., Wu M. W., (2002), "Performance-effective and low-complexity task scheduling for heterogeneous computing", *IEEE Transactions on Parallel and Distributed Systems*, 13 (3), 260-274.
- [26] Hsu C. C., Huang K. C., Wang F. J., (2011), "Online scheduling of workflow applications in grid environments", *Advances in Grid and Pervasive Computing 5th International Conference GPC 2010*, 300-310, Hualien, Taiwan, 10-13 May.
- [27] Hung P. P., (2019), "A dynamic scheduling method for collaborated cloud with thick clients", *Int. Arab J. Inf. Technol.*, 16 (4), 633-643.
- [28] Muhuri P. K., Rauniyar A., Nath R., (2019), "On arrival scheduling of real-time precedence constrained tasks on multi-processor systems using genetic algorithm", *Future Generation Computer Systems*, 93 (2019), 702-726.
- [29] Torabi S., Safi-Esfahani F., (2018), "A dynamic task scheduling framework based on chicken swarm and improved raven roosting optimization methods in cloud computing", *J Supercomput* 74 (1), 2581–2626.
- [30] Doğan M., Eröz R., Yüce H., Özmerdivenli R., (2018), "Yeni Nesil Dizileme (YND) Hakkında Bilinenler (Literatür Taraması)", *Duzce Medical Journal*, 19 (1), 27-30.
- [31] Dönmez D., Şimşek Ö., Kaçar Y. A., (2015), "Yeni Nesil DNA Dizileme Teknolojileri ve Bitkilerde Kullanımı", *Türk Bilimsel Derlemeler Dergisi*, 8 (1), 30-37.
- [32] Taşar O., Çınar E., Onay H., (2018), "Hastalık Tanısı İçin Yeni Nesil Dizileme Verisi Analizi: Gereksinimler ve Bir Çözüm Önerisi", *CERU-WS.org*, 2201, 12.
- [33] Pereira R., Oliveira J., Sousa M., (2020), "Bioinformatics and Computational Tools for Next-Generation Sequencing Analysis in Clinical Genetics", *J Clin Med.*, 9 (1), 132.
- [34] Li H., Durbin R., (2009), "Fast and accurate short read alignment with Burrows-Wheeler Transform", *Bioinformatics*, 25 (14), 1754-1814.
- [35] Langmead B., Salzberg S. L., (2012), "Fast gapped-read alignment with Bowtie 2", *Nat Methods*, 9 (4), 357-359.

- [36] Li R., Yu C., Li Y., Lam T. W., Yiu S. M., Kristiansen K., Wang J., (2009), "SOAP2: an improved ultrafast tool for short read alignment", *Bioinformatics*, 25 (15), 1966-1967.
- [37] Li H., Handsaker B., Wysoker A., (2009), "The Sequence Alignment/Map format and SAMtools", *Bioinformatics*, 25 (16), 2078–2079.
- [38] Gilissen C., Hoischen A., Brunner H. G., Veltman J.A., (2012), "Disease gene identification strategies for exome sequencing", *Eur J Hum Genet*, 20 (5), 490-497.
- [39] Stitzel N. O., Kiezun A., Sunyaev S., (2011), "Computational and statistical approaches to analyzing variants identified by exome sequencing", *Genome Biol*, 12 (9), 227.
- [40] McKenna A., Hanna M., Banks E., Sivachenko A., Cibulskis K., Kernysky A., Garimella K., Altshuler D., Gabriel S., Daly M., (2010), "The Genome Analysis Toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data", *Genome Res.*, 20 (9), 1297–1303.
- [41] McKenna A., Hanna M., Banks E., (2010), "Genom analizi araç takımı: yeni nesil DNA dizileme verilerini analiz etmek için bir MapReduce çerçevesi", *Genome Research*, 20 (9), 1297–1303.
- [42] Yang H., Wang K., (2015), "Genomic variant annotation and prioritization with ANNOVAR and WANNOVAR", *Nat. Protoc.*, 10 (10), 1556–1566.
- [43] McLaren W., Pritchard B., Rios D., Chen Y., Flicek P., Cunningham F., (2010), "Deriving the consequences of genomic variants with the Ensembl API and SNP Effect Predictor.", *Bioinformatics*, 26 (16), 2069–2070.
- [44] Cingolani P., Platts A., Wang L. L., Coon M., Nguyen T., Wang L., Land S. J., Lu X., Ruden D. M., (2012), "A program for annotating and predicting the effects of single nucleotide polymorphisms", *SnEff. Fly*, 6 (2), 80–92.
- [45] Ng S. B., Turner E. H., Robertson P. D., Flygare S. D., Bigham A. W., Lee C., Shaffer T., Wong M., Bhattacharjee A., Eichler E. E., (2009), "Targeted capture and massively parallel sequencing of 12 human exomes.", *Nature*, 461 (7261), 272.
- [46] Web 1, (2023), <http://www.ensembl.org>, (Erişim Tarihi: 02/06/2023).
- [47] Web 2, (2023), <http://www.ncbi.nlm.nih.gov/RefSeq/>, (Erişim Tarihi: 02/06/2023).
- [48] Web 3, (2022), <http://genome.ucsc.edu>, (Erişim Tarihi: 02/06/2023).
- [49] Lek M., Karczewski K. J., Minikel E. V., Samocha K. E., Banks E., Fennell T., O'Donnell-Luria A. H., Ware J. S., Hill A. J., Cummings B. B., (2016),

"Analysis of protein-coding genetic variation in 60,706 humans", *Nature*, 536 (7616), 285– 291.

- [50] Web 4, (2023), <http://gnomad.broadinstitute.org/>, (Eriřim Tarihi: 02/06/2023).
- [51] Web 5, (2023), <https://www.ncbi.nlm.nih.gov/projects/faspftp/1000genomes/>, (Eriřim Tarihi: 02/08/2023).
- [52] Zahedi S. M., Llull Q., Lee B. C., (2018), "Amdahl's Law in the Datacenter Era: A Market for Fair Processor Allocation", 2018 IEEE International Symposium on High Performance Computer Architecture (HPCA), 1-14, Vienna, Austria, 24-28 February.
- [53] Karp A., Flatt H., (1990), "Measuring parallel processor performance", *Communications of the ACM*, 33 (5), 539-543.
- [54] Web 6, (2023), <https://www.nih.gov/>, (Eriřim Tarihi: 02/08/2023).
- [55] Denzinger J., Fuchs M., Fuchs M., (1996), "High performance atp systems by combining several ai methods", *IJCAI'97: Proceedings of the 15th international joint conference on Artificial intelligence*, 102-107, Nagoya, Japan, 23-29 August.
- [56] Fisher M., Thompson G., (1961), "Probabilistic learning combinations of local job-shop scheduling rules", *Factory Scheduling Conference*, 225-251, Prentice Hall, Englewood Cliffs, NJ, 1 November.
- [57] Cowling P., Kendall G., Soubeiga E., (2001) "A hyperheuristic approach to scheduling a sales summit", *3rd International Conference on Practice and Theory of Automated PATAT 2000*, 176–190, Konstanz, German, 16-18 August.
- [58] Burke E. K., Gendreau M., Hyde M., Kendall G., Ochoa G., Ozcan E., Qu R., (2012), "Hyper-heuristics: A survey of the state of the art", *the Journal of the Operational Research Society*, 64 (12), 1695–1724.
- [59] Burke E. K., Hyde M., Kendall G., Ochoa G., Ozcan E., Wood J., (2010), "A Classification of Hyper-heuristic Approaches", In: M. Gendreau, J.Y. Potvin, Editors, "Handbook of Metaheuristics", Springer Publishing Company.
- [60] Drake J., Hyde M., Ibrahim K., Özcan E., (2014), "A Genetic Programming Hyper-Heuristic for the Multidimensional Knapsack Problem", *Kybernetes*, 43 (9), 1500-1511.
- [61] Karaboęa, D., (2018), "Yapay Zeka Optimizasyon Algoritmaları", 7. Baskı, Nobel Akademik Yayıncılık.
- [62] Tobita T., Kasahara H., (2002), "A standard task graph set for fair evaluation of multiprocessor scheduling algorithms", *J Sched*, 5 (5), 379-394.

- [63] Web 7, (2022), <https://www.kasahara.cs.waseda.ac.jp/>, (Eriřim Tarihi: 02/08/2023).
- [64] Almeida V. A. F., Vasconcelos I. M. M., Arabe J. N. C., Menasce D. A., (1992), “Using random task graphs to investigate the potential benefits of heterogeneity in parallel systems”, Supercomputing '92:Proceedings of the 1992 ACM/IEEE Conference on Supercomputing, 683-691, Minneapolis, MN, USA, 16-20 November.
- [65] Yang T., Gerasoulis A., (1994), “DSC: scheduling parallel tasks on an unbounded number of processors”, IEEE Trans Parallel Distrib Syst, 5 (9), 951-967.
- [66] Kasahara H., Narita S., (1985), “Robot-Arm Control Computation on a Multiprocessor System”, IEEE Trans Rob Autom, 1 (2), 104-113.



ÖZGEÇMİŞ

Gülçin YILDIZ, 2008 yılında başladığı Namık Kemal Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümünü 2012 yılında tamamladı. 2021 yılında yüksek lisans eğitimine Gebze Teknik Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim dalında başladı. Şu anda Aksigorta şirketinde kıdemli yazılım uzmanı olarak görevine devam etmektedir.



EKLER

Ek A: Tez Çalışması Kapsamında Yapılan Yayınlar

Bolat G., Sevilgen F. E., (2022), “Paralel İşlerin Çizelgelemesinde İşlemci Tahsisi için Hiper Sezgisel Yaklaşım”, 7th HPC Conference (BAŞARIM 2022), 1-7, İstanbul, Türkiye, 11-13 Mayıs.

Yildiz G., Sevilgen F. E., (2023), “Hyper-heuristic method for processor allocation in parallel tasks scheduling”, Concurrency Computat Pract Exper., e7757.

Ek B: Terimler Sözlüğü

Tablo B1.1: Tez çalışmasında Türkçe olarak kullanılan İngilizce terimler.

Türkçe Terim	İngilizce Terim
Çizelge Uzunluğu Oranı	Schedule Length Ratio
Çoğaltma	Duplication
Yönlü döngüsüz çizge	Directed Acyclic Graph
Geri doldurma	Backfilling
Göç	Migration
İşlemci tahsisi	Processor Allocation
Maksimum tamamlanma süresi	Makespan
Paketleme	Packing allocations
Rulet Seçimi	Roulette Selection
Son Teslim Tarihi	Deadline
Standart İşlem Çizge	Standart Task Graph
Yönlü döngüsüz çizge	Directed Acyclic Graph