YUSUFÇUK OPTİMİZASYON ALGORİTMASININ DAĞITIK VE PAYLAŞIMLI BELLEK MİMARİLERİ ÜZERİNDE PARALELİZASYONU

YÜKSEK LİSANS TEZİ

RAMAZAN POLAT

MERSİN ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ
ANABİLİM DALI

MERSİN
KASIM - 2019

YUSUFÇUK OPTİMİZASYON ALGORİTMASININ DAĞITIK VE PAYLAŞIMLI BELLEK MİMARİLERİ ÜZERİNDE PARALELİZASYONU

YÜKSEK LİSANS TEZİ

RAMAZAN POLAT

MERSİN ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ
ANABİLİM DALI

Danışman
Prof. Dr. Ali AKDAĞLI

MERSİN
KASIM - 2019

ONAY

Ramazan Polat tarafından Prof. Dr. Ali AKDAĞLI danışmanlığında hazırlanan "Yusufçuk optimizasyon algoritmasının dağıtık ve paylaşımlı bellek mimarileri üzerine paralelizasyonu" başlıklı çalışma aşağıda imzaları bulunan jüri üyeleri tarafından 22 Kasım 2019 tarihinde yapılan Tez Savunma Sınavı sonucunda oy birliği ile Yüksek Lisans tezi olarak kabul edilmiştir.

Görevi	Ünvanı, Adı ve Soyadı	İmza
Başkan	Prof. Dr. Ali AKDAĞLI	
Üye	Dr. Öğr. Üyesi Çiğdem ACI	
Üye	Dr. Öğr. Üyesi Deniz ÜSTÜN	

Yukarıdaki Jüri kararı Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 24.01.2020 tarih ve 2020/155 sayılı kararıyla onaylanmıştır.

Prof. Dr. Cahit BİLİM
Fen Bilimleri Enstitü Müdürü

Bu tezde kullanılan özgün bilgiler, şekil, tablo ve fotoğraflardan kaynak göstermeden alıntı yapmak 5846 sayılı Fikir ve Sanat Eserleri Kanunu hükümlerine tabidir.

ETİK BEYAN

Mersin Üniversitesi Lisansüstü Eğitim-Öğretim Yönetmeliğinde belirtilen kurallara uygun olarak hazırladığım bu tez çalışmasında,

- Tez içindeki bütün bilgi ve belgeleri akademik kurallar çerçevesinde elde ettiğimi,
- Görsel, işitsel ve yazılı tüm bilgi ve sonuçları bilimsel ahlâk kurallarına uygun olarak sunduğumu,
- Başkalarının eserlerinden yararlanılması durumunda ilgili eserlere bilimsel normlara uygun olarak atıfta bulunduğumu,
- Atıfta bulunduğum eserlerin tümünü kaynak olarak kullandığımı,
- Kullanılan verilerde herhangi bir tahrifat yapmadığımı,
- Bu tezin herhangi bir bölümünü Mersin Üniversitesi veya başka bir üniversitede başka bir tez çalışması olarak sunmadığımı,
- Tezin tüm telif haklarını Mersin Üniversitesi'ne devrettiğimi

beyan ederim.

ETHICAL DECLERATION

This thesis is prepared in accordance with the rules specified in Mersin University Graduate Education Regulation and I declare to comply with the following conditions:

- I have obtained all the information and the documents of the thesis in accordance with the academic rules.
- I presented all the visual, auditory and written informations and results in accordance with scientific ethics.
- I refer in accordance with the norms of scientific works about the case of exploitation of others' works.
- I used all of the referred works as the references
- I did not do any tampering in the used data.
- I did not present any part of this thesis as an another thesis at Mersin University or another university.
- I transfer all copyrights of this thesis to the Mersin University.

22 Kasım 2019 / 22 Nov 2019

Ramazan POLAT

ÖZET

YUSUFÇUK OPTİMİZASYON ALGORİTMASININ DAĞITIK VE PAYLAŞIMLI BELLEK MİMARİLERİ ÜZERİNDE PARALELİZASYONU

Tez çalışmasında son yıllarda çok kullanılan ve sürü davranışlarını örnek alan optimizasyon algoritmalarından biri olan Yusufçuk Algoritmasının (YA) Paylaşımlı ve Dağıtık Bellek (PDB) ile tasarlanması ele alınmıştır. Günümüzde doğa esinli meta sezgisel optimizasyon algoritmalarının paralel hesaplama ve programlama teknikleriyle çözümlenmesi araştırmacılar tarafından sıkça incelenmiştir. Bu çalışmalarda, paylaşımlı ve dağıtık bellek mimarileri kullanılarak, algoritmanın çalışma süresinde kayda değer bir iyileşme gözlemlenmiştir. Doğadaki canlıların içgüdüsel davranışlarını ele alan bu algoritma en uygun sonuca gitmek için her bir yusufçuğun belirli bir fonksiyon üzerinden optimum değerinin hesaplandığı davranış biçimi incelenmiştir. Bu tez çalışmasının amacı, optimizasyon algoritmalarını, paralel programlama teknikleri kullanarak zaman ve bellek açısından daha yüksek başarımlı hale getirebilmektir. Aynı zamanda düşük maliyet ve yüksek performans sunan yeni bir yaklaşım olan, paralel programlama teknikleri ile verimliliği arttırmaktır. Grafik İşlem Birimi (GPU) programlama tekniklerini kullanarak daha hızlı çözüm üreten YA algoritmasının paralelleştirilmesi incelenmiştir. Ayrıca diğer optimizasyon algoritmaları ile karşılaştırılarak YA 'nın önemi belirtilmiştir. Yusufçuk algoritması formulasyonunun performansı, farklı boyutlardaki tek veya çok modlu karşılaştırma fonksiyonları kullanılarak incelenmiş ve daha sonra aynı algoritma NVIDIA firması tarafından geliştirilen Birleşik Hesap Cihazı Mimarisi (CUDA) mimarisinde paralel olarak geliştirilmiştir. YA 'nın paralelleştirilmiş halinin performansı test fonksiyonları üzerinde analiz edilmiş ve seri halinin performansı ile kıyaslanmıştır. Paralel olarak ele alınan YA 'nın, CPU üzerinde karşılaştırdığımızda sistem 9 kat, GPU üzerinde ise 20 kat hızlanmıştır. Paralel olarak ele alınan Çok amaçlı Yusufcuk Algoritması (CAYA) 'nın CPU üzerinde karşılaştırdığımızda sistem 4 kat, GPU üzerinde ise 10 kat hızlanmıştır.

Anahtar Kelimeler: Yusufçuk Algoritması, Optimizasyon, Çok Katmanlı Dağıtık ve Paylaşımlı Bellek, Paralelizasyon, Cuda Programlama

Danışman: Prof. Dr., Ali AKDAĞLI, Mersin Üniversitesi, Bilgisayar Bilimleri Anabilim Dalı, Mersin

ABSTRACT

PARALLELIZATION OF DRAGONFLY OPTIMIZATION ALGORITHM ON DISTRIBUTED AND SHARED MEMORY ARCHITECTS

In this thesis, the design of Dragonfly Algorithm (DA), which is one of the optimization algorithms used in recent years and which takes the swarm behavior as an example, is designed with Distributed and Shared Memory (DSM). Nowadays, the analysis of nature-inspired meta-heuristic optimization algorithms with parallel computing and programming techniques has been frequently studied by researchers. In these studies, a significant improvement was observed in the runtime of the algorithm using shared and distributed memory architectures. Using this algorithm, which deals with the instinctive behavior of living beings in nature, the behavior of which the optimum value is calculated is examined using a certain function of dragonfly. The aim of this thesis is to make optimization algorithms more efficient in terms of time and memory by using parallel programming techniques. At the same time, it is to increase productivity with parallel programming techniques, which is a new approach that offers low cost and high performance. The parallelization of the DA algorithm, which produces faster solutions using Graphic Processing Unit (GPU) programming techniques, is examined. In addition, the importance of DA is indicated by comparing with other optimization algorithms. The performance of the dragonfly algorithm formulation was examined using single or multi-mode comparison functions of different sizes and then the same algorithm was developed in parallel with the Compute Unified Device Architecture (CUDA) architecture developed by NVIDIA. The performance of the parallelized state of DA was analyzed on the test functions and compared with the performance of the series state. In parallel, DA is accelerated 9 times on the CPU and 20 times on the GPU. Multi-Objective Dragonfly Algorithm (MODA), which is handled in parallel, accelerated 4 times on the CPU and 10 times on the GPU.

Keywords: Dragonfly Algorithm, Optimization, Multi-tier Distributed and Shared Memory, Parallelization, Cuda Programming

Advisor: Prof. Dr., Ali AKDAĞLI, Department of Computer Science, University of Mersin, Mersin.

TEŞEKKÜR

Bu çalışmayı hazırlamam konusunda destek olan danışman hocam Prof. Dr. Ali AKDAĞLI, Dr. Öğr. Üyesi Çiğdem ACI hocama, değerli kardeşim Oğuzhan POLAT , saygıdeğer anneciğim Zennup AKDOĞAN ve öğrenim hayatımız boyunca her türlü desteği aldığımız değerli arkadaşlarıma teşekkürlerimi sunarım.

İÇİNDEKİLER

TABLOLAR DİZİNİ

ŞEKİLLER DİZİNİ

KISALTMALAR ve SİMGELER

Kısaltma/Simge	Tanım
ABA	Ateş Böceği Algoritması
CAYA	Çok Amaçlı Yusufçuk Algoritması
CPU	Merkezi İşlem Birimi (Central Processing Unit)
GPU	Grafik İşlem Birimi (Graphics Processing Unit)
IYA	Tek amaçlı İkili Yusufcuk Algoritması
KKO	Karınca Koloni Optimizasyonu
OPENGL	Grafik programlama kütüphanesi
PDB	Paylaşımlı ve Dağıtık Bellek
PSO	Parçacık Sürü Optimizasyonu
UCI	Makine Ögrenimi Deposu (Machine Learning Repository)
YA	Tek Amaçlı Yusufçuk Algoritması
YAKA	Yapay Arı Kolonisi Algoritması

1. GİRİŞ

Son zamanlarda meta sezgisel algoritmalar, veri miktarının artması ile mevcut sistemlerde büyük-ölçekli verileri işleyen algoritmalar tarafından çalışan uygulamaların gereksinimlerini karşılayamaz hale gelmiştir. Bu sebeple teknolojinin de hızla gelişmesiyle algoritmaların zaman ve maliyet bakımından önemi artmıştır. Araştırmacılar buna bağlı olarak düşük maliyet ve yüksek performans sunan yeni bir yaklaşım olan paralel programlamada genel amaçlı Grafik İşlem Birimi (GPU) programlama tekniklerine yönelmişlerdir.

Paralel programlama büyük miktarda işlem ve zaman gerektiren işlerin birden fazla işlemciye dağıtılarak çözülmesini amaçlayan bir programlama şeklidir. Yani çok fazla miktarda hesaplama yapılması gerekiyorsa bu hesaplamalar belirli algoritma ve teknikler kullanılarak, birden fazla işlemciye paylaştırılıp hesaplatılır. Geleneksel yöntemleri kullanan bir bilgisayarda komutlar sırayla işleme alınıp, birim zamanda sadece bir komut işlenebilirken; paralel programlama mantığı ile tasarlanmış bir yazılım aynı anda birden çok komutu farklı işlemciler üzerinde işleyerek hem zaman hem de kaynak kullanımında büyük fayda sağlar.

Zaman ve maliyet açısından önemli olan sistemlerde, tasarım sürecindeki model, bilgisayar ortamında simulasyon ile test edilerek iyileştirmeler yapılmaktadır. Örnek olarak günümüzde araba firmaları üreteceği arabaların modellerini bilgisayar ortamında simulasyon yoluyla çarpışma ve hasar sonuçlarını değerlendirip, üreteceği modeli kısa zamanda iyileştirerek revizyonlarını yaparak, üretime geçmektedir [1].

Doğa esinli optimizasyon algoritmaları, yapıları gereği uzun süren hesaplamalarla birlikte yüksek iterasyon ve popülasyon sayılarına sahip olabilirler. Bu problemleri çözebilmek için literatürde bu algoritmaların paralelleştirilmesiyle ilgili pek çok çalışma yapılmıştır. Bu çalışmalarda, paylaşımlı ve/veya dağıtık bellek mimarileri kullanılarak, algoritmanın çalışma süresinde kayda değer gelişmeler gözlemlenmiştir. 2015 yılında Mirjalili tarafından önerilmiş olan Yusufcuk Algoritması (YA), doğada bulunan yusufçukların sürü halindeki statik ve dinamik davranışlarını inceleyip, matematiksel modellemesi oluşturulan doğa esinli bir optimizasyon algoritmasıdır [2]. Bu algoritma yusufçuk böceklerinin besin arama ve düşmandan kaçınma olarak belirlenen iki temel içgüdüsel davranışının incelenmesiyle ortaya çıkmıştır. Yusufçuklar sürü halinde oldukları zaman kendilerine has işaretleşme hareketleri sergileyip zekice bir iletişim kurmaktadırlar. Matematiksel modellemesinin açık şekilde ortaya konulduğu Mirjalili'nin makalesinde bu davranışlar detaylıca incelenip paylaşılmıştır. Bununla birlikte, YA diğer doğa esinli algoritmalarda olduğu gibi yüksek iterasyon ve popülasyon sayılarında oldukça fazla işlemci zamanı tüketmektedir.

Bu tez çalışmasının amacı, optimizasyon algoritmalarını, paralel programlama teknikleri kullanarak zaman ve bellek açısından daha yüksek başarımlı hale getirebilmektir.

GPU programlama tekniklerini kullanarak düşük maliyet ve yüksek performans ile daha hızlı çözüm üreten YA algoritmasının paralelleştirilmesi incelenmiştir. YA 'nın önemini belirtmek adına en önemli optimizasyon algoritmalarından olan Parçacık Sürü Optimizasyonu (PSO) ve Yapay Arı Kolonisi (YAKA) optimizasyon algoritmaları ile karşılaştırılmıştır.

YA ile ilgili olarak Ç. Acı ve H. Gülcan tarafından yapılan çalışmada Brownian hareketi ile iyileştirilmiş YA nın kaynak kiriş tasarımı problemine uygulaması ele alınmış ve %20 oranında daha düşük hesaplama ile başarı sağlamış [3]. M. M. Mafarja tarafından yapılan çalışmada İkili Yusufcuk Algoritması (IYA) önerilmiştir. Önerilen yaklaşımın performansını değerlendirmek için 18 UCI veri seti kullanılmış ve yöntemin sonuçları, sınıflandırma doğruluğu ve seçilen niteliklerin sayısı bakımından PSO, Genetik Algoritma sonuçları ile karşılaştırılmıştır. Sonuçlar, IYA' nın başarılı yeteneğini göstermiştir [4]. Arun Vikram ve arkadaşları tarafından yapılan çalışmada Frezeleme operasyonunda işlem parametrelerinin optimal performans analizi için örnek bir YA uygulaması yapmıştır. Frezeleme operasyonunda A ekseninde teğet ve dik açılı işlemleri gerçekleştirirken yüzey pürüzlülüğü, yüzey sertliği gibi işlem parametrelerinin üretilmesinde optimum çözüme odaklanılmıştır [5]. F. Katircioglu çalışmasında, Renkli görüntülerde eşik yöntemi ile kenar belirleme ve bölütleme tezini YA kullanarak gerçekleştirmiştir. Elde edilen matematiksel uygulama sonuçları, literatürdeki Yerçekimi Arama ve Harmoni Arama Algoritması ile karşılaştırılmıştır. YA algoritması ile gerçekleştirilen çalışma sonucu önerilen algoritmanın sayısal görüntü bölütleme ve kenar belirleme konusunda da güçlü olduğu görülmüştür [6].

CUDA ile ilgili olarak M.Kıran ve A. Çınar tarafından yapılan çalışmada Ağaç tohumu algoritmasının CUDA alt yapısı ile GPU üzerinde paralel uygulaması ele alınmıştır. Deneysel çalışmalar sonucunda algoritmanın paralel versiyonunun seri versiyonuna göre bazı problemler için 184,65 kata kadar performans artışı sağladığı görülmüştür [7]. 2008 yılında Garland ve arkadaşları tarafından CUDA ile medikal alanda görüntü islemeye yönelik yaptıkları çalışmada CPU ve GPU üzerindeki performans değerlendirmesi de yapmışlardır [8].

2. KAYNAK ARAŞTIRMALARI

Bu bölümde optimizasyon ve paralel programlama ile ilgili kaynak araştırmasına yer verilmiştir.

2.1. Sürü davranışı

Bir araya kümelenmiş benzer hayvanların aynı bölgelerde gezmeleri, sürü halinde hareket etmeleri veya aynı yöne doğru göç etmeleri ile sergilenen toplu bir davranıştır. Craig Reynolds' ın 1986 da geliştirdiği boids yazılımı; "komsularınla aynı yöne ilerle" , "komşularına yakın dur" ve "komşularınla çarpışmaktan kaçın" kurallarını temel alan sürü davranışının yazılımıdır. Araştırmacılar tarafından yapılan ve günümüzde incelenen algoritmaların matematiksel modeli de bu 3 temel kurala dayanmaktadır [9]. Sürü davranışı birçok araştırmacı tarafından çeşitli alanlarda kullanılmıştır [10]. Literatürde en fazla kullanılan sürü algoritmaları olarak Karınca Koloni Optimizasyonu (KKO) [11-13], YAKA algoritması [14], Parçacık-Sürü Optimizasyonu (PSO) [15-17], Ateş Böceği Algoritması (ABA) [18], Yarasa Optimizasyon Algoritması [19], Aslan Optimizasyon Algoritması [20] sayılabilir.

YAKA algoritması, sayısal problemleri optimize etmek için Karaboga tarafından 2005'te sunulan bir meta-sezgisel algoritmadır [21]. KKO algoritması, zor optimizasyon problemlerine yaklaşık çözümler bulmak için kullanılabilecek popülasyona dayalı bir meta-sezgisel algoritmadır. ABA algoritması , Xin-She Yang [18] tarafından parlaklığa duyarlı ateşböceklerinin sosyal davranışlarını ele alarak geliştirilmiştir.

2.2. Optimizasyon

Optimizasyon temelde, eldeki kısıtlı kaynakların optimum kullanımını gerçekleştirmektedir. Optimizasyonda bir amaç fonksiyonunu verilen tanım aralığında optimum yapan değerini bulma anlayışı vardır. Yöneylem araştırması ve işletme yönünden, bir sistemi en düşük maliyetle en yüksek verimi elde etmek üzere düzenleme yapılması olarak tanımlanabilir [22].

Optimizasyonun temelinde her zaman istenilene ulaşma arzusu söz konusudur. Geçmişten günümüze kadar, karşılaşılan mühendislik problemlerinin çözülmesi amacıyla bir çok optimizasyon teknikleri geliştirilmiş ve farklı alanlara uygulanmıştır [23]. Optimizasyon problemlerinin farklı yapıları ve matematiksel özellikleri, devamlılık , doğrusal olmayan, dış bükeylik vb . özellikleri olan özel sistemlerin çözümünde kullanılacak yazılımlar geliştirilmesini gerektirmiştir [24].

Tez çalışmasında inceleyeceğimiz Yusufcuk Algoritması 'nın temeli olan Sürü davranışı ve optimizasyon ile ilgili çalışmada davranışı anlayabilmek için optimizasyonun iki önemli bileşeninden faydalanılmıştır

Optimizasyonun iki önemli bileşeni ise matematiksel modelleme ve modellemeden sonraki çözümleme işlemleridir. Optimizasyonun gelişmesinde öncelikle modelleme konusuyla ilgilenmişlerdir. Bu konudaki ilk araştırma Leontief tarafından yapılan Dış ticaret analizinde kayıtsızlık eğrilerinin kullanımı hakkında olmuştur [25].

2.2.1. Optimizasyon Çeşitleri

Amaçlarına göre Tek ve Çok amaçlı optimizasyon olmak üzere 2 şekilde incelenir.

2.2.2. Tek Amaçlı Problemler

Tek amaçlı optimizasyonda, tek bir hedef için en uygun olan bir tasarım modeli elde edilmeye çalışılır ki bu da genellikle minimizasyon veya maksimizasyon problemine dayalı olarak global minimum veya global maksimumun bulunmasıdır [22].

Tablo 2.1'de Yelpaze , Pervane , Fan , Klima gibi soğutucuların A1, A2, A3, A4 ve A5 kodu ile ürünlerin fiyatı TL cinsinden ve konforu ise yüzde cinsinden verilmiştir.

Tablo 2.1. Ürünlerin Fiyatı ve Konfor Özellikleri

Ürün	Fiyatı (TL)	Konfor (Yüzde)
A1	10	0
A2	30	30
A3	50	70
A4	100	70
A5	500	100

Tablo 2.1'de verilen A1, A2, A3, A4, A5 ürünlerini fiyat ve konforuna göre tek amaçlı olarak karşılaştırdığımızda

- Ürünlerden en uygun fiyatlı olanı A1 dir.
- Ürünlerden en konforlu olanı A5 tir.
- Ürünlerden A4 ve A5 arasında fiyatı en iyi olanı A4 tür.
- Ürünlerden A1 ve A5 arasında en konforlu olanı A5 tir.

Düşük kütle ve yüksek dayanıklılıkla iyi bir tasarım yaparak bir köprü inşa edilebilir. Yakıt verimliliği, yük ve ağırlık gibi parametrelerin eş zamanlı optimizasyonu ile uçak tasarımı

yapılabilir. Bir arabanın tasarımı için sürücünün duyacağı gürültüyü minimize etmek, havalandırmanın ise maksimize etmek amaçlanır [26].

2.2.3. Çok Amaçlı Problemler

Çok amaçlı optimizasyon problemlerinde, birden fazla ideal çözüm elde edildiğinden dolayı tüm amaçlara yönelik istekleri karşılaması çok zordur. Genellikle bir tek çözüm yerine çözüm kümesinden bahsedilmektedir. Bu çözümlere Pareto (veya baskın) çözüm denilmektedir. Çok amaçlı optimizasyonun esas yapısı, baskın olmayan çözümleri çözüm kümesinden eleyerek baskın olan iyi çözümlere ulaşmayı amaçlar [22].

Çok amaçlı optimizasyon problemlerinin çözülmesi için Pareto-optimal kümelerinin elde edilmesi gerekmektedir. Karar verici, problemin en iyi çözümlerini içeren bu küme içinden uygun çözümü seçmektedir. Çok amaçlı problemlerin, gerçek hayatta çok farklı alanlarda ortaya çıkması, çok çeşitli çözüm yöntemlerinin geliştirilmesine olanak sağlamıştır. Bu çözüm yöntemlerinin belirli özelliklerine göre sınıflandırılması, problemlerin yapısına uygun çözüm yöntemi nin seçilmesini kolaylaştırmaktadır.

Literatürde çeşitli sınıflandırma örnekleri vardır [27].

- Tercih bilgisinin kullanılmadığı yöntemler
- Tercih bilgisinin sonradan açıklandığı yöntemler
- Tercih bilgisinin önceden belirlendiği yöntemler
- Tercih bilgisinin ilerleyen bir ifadesinin yer aldığı yöntemler (interaktif yöntemler)

Tablo 2.1'de verilen A1, A2, A3, A4, A5 ürünlerini fiyat ve konforuna göre çok amaçlı olarak karşılaştırdığımızda

- Fiyatı 500 TL'den küçük, konfor oranı %30'dan büyük olanı A3 ve A4 ürünleridir. A3 ve A4 çözümleri birbirlerine göre baskın değillerdir.

Gerçek sorunların çoğunun birden fazla hedefi olduğu için birden fazla çözüm üretilmektedir. Çözümleri karşılaştırmamız için Pareto optimal kavramı ele alınmaktadır. Bu çözümlerin en az birinde daha iyi bir değer sağlanırsa diğer çözümlerden daha iyidir diyebiliriz.

2.2.4. Optimizasyon İçin Kıyaslama Fonksiyonları

Optimizasyon işlemlerinin kalitesi (halihazırda bilinenler ve yeni önerilenler) sık sık Tablo 2.3 ve Tablo 2.4'te verilen literatür karşılaştırma ölçütleri kullanılarak değerlendirilir [28].

2.2.4.1. Tek Amaçlı Problemler İçin Kıyaslama Fonksiyonları

Tablo 2.2. Tek Amaçlı Problemler için Kıyas Fonksiyonları

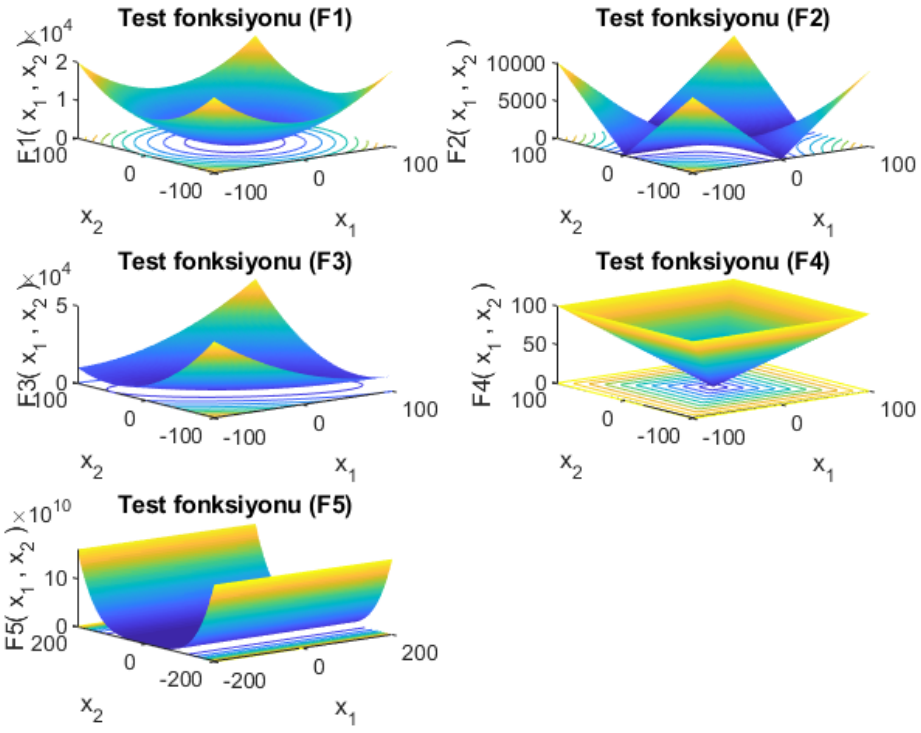
Fonksiyon	Boyut	Aralık
F1(x) = $\sum_{i=1}^{n} x_i^2$	10	[-100:100]
F2(x) = $\sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	10	[-10:10]
F3(x) = $\sum_{i=1}^{n} \left(\sum_{J-1}^{i} X_J\right)^2$	10	[-100:100]
F4(x) = $\max_i\{\lvert x_i \rvert, 1 \le i \le n\}$	10	[-100:100]
F5(x) = $\sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	10	[-30:30]
F6(x) = $\sum_{i=1}^{n} ([x_i + 0.5])^2$	10	[-100:100]
F7(x) = $\sum_{i=1}^{n} i x_i^4 + random[0,1)$	10	[-1.28:1.28]
F8(x) = $\sum_{i=1}^{n} -x_i sin(\sqrt{\lvert x_i \rvert}$	10	[-500:500]
F9(x) = $\sum_{i=0}^{n} [x_i^2 - 10cos(2\pi x_i) + 10]$	10	[-5.12:5.12]
F10(x) = -20exp (-0.2$\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}$)-exp ($\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)$)+20+e	10	[-32:32]
F11(x) = $\frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}} + 1\right)$	10	[-600:600]
F12(x) = $\frac{\pi}{n}\{10 \sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10sin^2(\pi y_{i+1})] + (y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = \frac{x_i + 1}{4}$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	10	[-50:50]
F13(x) = 0.1$\{sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2[1 + sin^2(3\pi x_i + 1)] + (x_n - 1)^2[1 + sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	10	[-50:50]

2.2.4.2. Çok Amaçlı Problemler İçin Kıyaslama Fonksiyonları

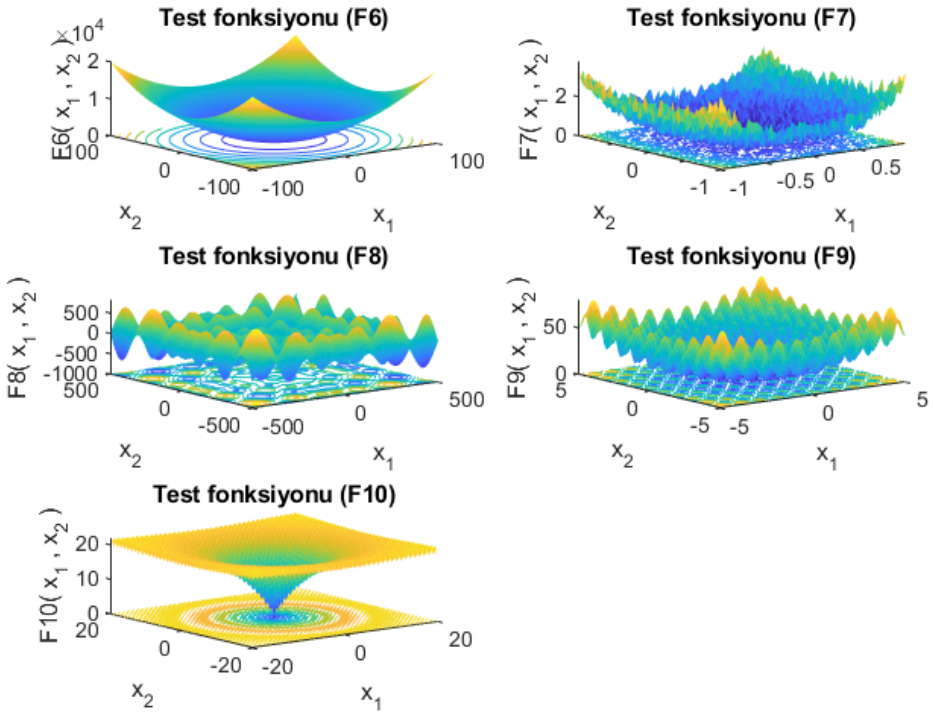
Tablo 2.3. Çok Amaçlı Problemler için Kıyas Fonksiyonları

Problem	Tanım
ZDT1	Minimize: $f_1(x) = x_1$ Minimize: $f_2(x) = g(x)\ x\ h(f_1(x), g(x))$ Şöyle ki: $G(x) = 1 + \frac{9}{N-1}\sum_{i=2}^{N} x_i$ $h(f_1(x), g(x)) = 1 - \sqrt{\frac{f_1(x)}{g(x)}}\ 0 \leq x_i \leq 1, 1 \leq i \leq 30$
ZDT2	Minimize: $f_1(x) = x_1$ Minimize: $f_2(x) = g(x)\ x\ h(f_1(x), g(x))$ Şöyle ki: $G(x) = 1 + \frac{9}{N-1}\sum_{i=2}^{N} x_i$ $h(f_1(x), g(x)) = 1 - \left(\frac{f_1(x)}{g(x)}\right)^2\ 0 \leq x_i \leq 1, 1 \leq i \leq 30$
ZDT 3	Minimize: $f_1(x) = x_1$ Minimize: $f_2(x) = g(x)\ x\ h(f_1(x), g(x))$ Şöyle ki: $G(x) = 1 + \frac{9}{29}\sum_{i=2}^{N} x_i$ $h(f_1(x), g(x)) = 1 - \sqrt{\frac{f_1(x)}{g(x)}} - \left(\frac{f_1(x)}{g(x)}\right)\sin(10\pi f_1(x))$ $0 \leq x_i \leq 1, 1 \leq i \leq 30$
ZDT 4	Minimize: $f_1(x) = x_1$ Minimize: $f_2(x) = g(x)\ x\ h(f_1(x), g(x))$ Şöyle ki: $G(x) = 1 + 10(N-1) + \sum_{i=2}^{N}\left(x_i^2 - 10\sin(4\pi x_i)\right)$ $h(f_1(x), g(x)) = 1 - \sqrt{\frac{f_1(x)}{g(x)}}\ 0 \leq x_i \leq 1, 1 \leq i \leq 30$
ZDT 6	Minimize: $f_1(x) = 1 - \exp(-4 * x_1) * \sin(6\pi x_1)^6$ Minimize: $f_2(x) = g(x)\ x\ h(f_1(x), g(x))$ Şöyle ki: $G(x) = 1 + 9\frac{\left(\sum_{i=2}^{N} x_i\right)}{(N-1)^{0.25}}$ $h(f_1(x), g(x)) = 1 - \left(\frac{f_1(x)}{g(x)}\right)^2\ 0 \leq x_i \leq 1, 1 \leq i \leq 30$

Şekil 2.1 ve Şekil 2.2'de YA algoritmasının testinde kullanılan ve araştırmacıların sıklıkla kullandığı kıyas fonksiyonlarının grafikleri gösterilmektedir.



Şekil 2.1 F1- F5 arası Test fonksiyonları



Şekil 2.2 F6- F10 arası Test fonksiyonları

2.3. Paralel Programlama

Paralel kavramı ile ilgili, ilk olarak 1958'de IBM araştırmalarında nümerik hesaplamaların yapılabileceği fikrini ortaya atan J. Cocke ve D. Slotnick ile ortaya çıkmıştır. Paralel programlama, bir problemin birden fazla bilgisayar veya işlemci kullanılmasıyla kısa sürede çözüme ulaşılması için kullanılmaktadır. Tez çalışmasında da 2 düzeyde paralellik incelenmiştir.

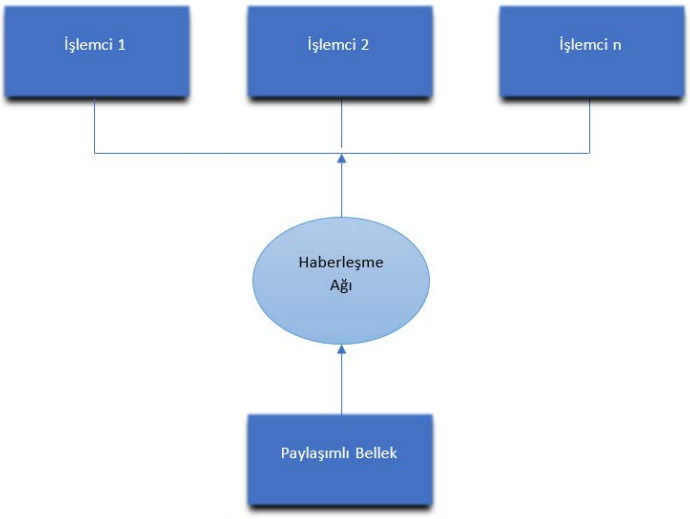
Veri düzeyinde paralellik : Veri paralelliği paralel olarak işlenecek farklı işlem düğümleri arasında veri dağıtımına odaklanan bir yöntemdir. Veri düzeyinde paralellik programlamada döngülerin yapısında karşımıza çıkar. Tüm işlemci dağıtılmış verilerin farklı parçaları üzerinde aynı görevi yaptığında veri paralelliği sağlanmış olur [29].

Görev paylaşımlı paralellik: Görev paylaşımlı paralel sistemlerde her fonksiyon farklı işlemciler üzerinde çalışır. Bu yüzden görev paylaşımlı paralel sistemler ortak bellekli (shared memory) ve dağıtık bellekli (distributed memory) olmak üzere iki ana sınıfa ayrılmaktadır. Görev paylaşımlı paralel sistemler CPU ve GPU üzerinde denenebilme yetisine sahiptir [29].

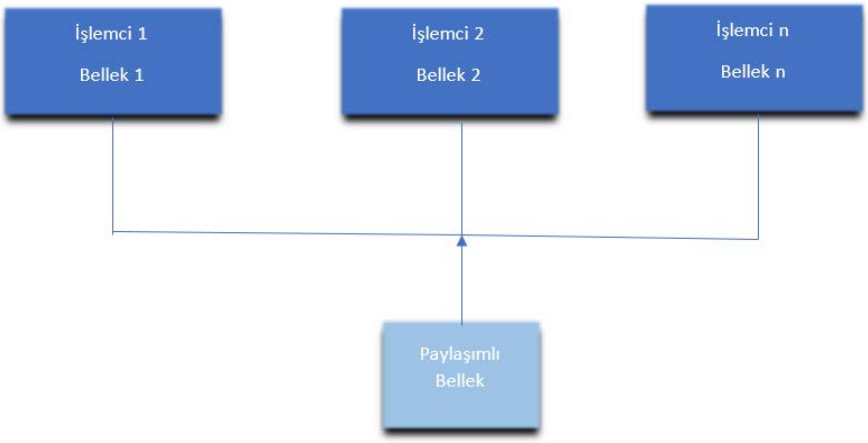
Paylaşımlı bellekli sistemlerde işlemciler ortak bir belleğe erişirler. Aynı bellek bölgesine eş zamanlı erişimleri engellemek açısından senkronizasyon büyük önem taşır. Dağıtık bellekli sistemlerde ise her bir işlemci kendi belleğine sahiptir ve bir işlemci başka bir işlemcinin belleğindeki veriyi mesaj geçme (message passing) ile elde eder [30].

Her iki sistemin de birbirlerine göre kullanım alanları olarak avantajları ve dezavantajları mevcuttur.

Şekil 2.3 ve Şekil 2.4'de sırasıyla paylaşımlı ve dağıtık bellekli paralel bilgisayar sistemlerinin mimarisi verilmiştir [31].

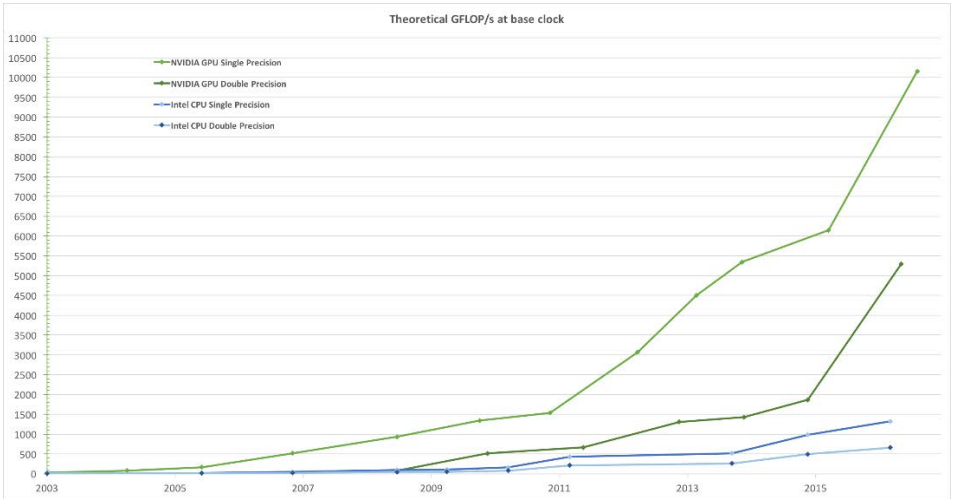


Şekil 2.3. Paylaşımlı Bellekli Paralel Bilgisayar Sistemi

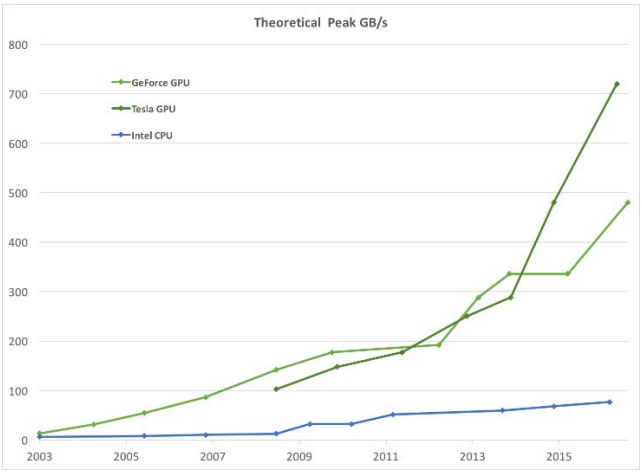


Şekil 2.4. Dağıtık Bellekli Paralel Bilgisayar Sistemi

1980'li yıllarda IBM ve Intel firmaları GPU geliştirmeye başlamış. 1990 da ise NVIDIA , S3 ve ATI gibi firmalar sadece grafik kartı geliştirmeye yönelmişlerdir. 2 boyutlu donanımsal hızlandırma işlemi yaygınlaştı ve OpenGL kullanılmaya başlandı. 2000 yıllarında ise GPU'nun matematiksel hesaplama gücü oldukça kuvvetlendi. 3 boyutlu gerçek zamanlı ve yüksek çözünürlüklü grafikler için oldukça fazla talep nedeniyle GPU'lar istenilen hale gelmiştir. Günümüze gelindiğinde GPU'lar yüksek hesaplama gücüne, çok çekirdekli işlemciye ve çok yüksek bant genişliğine sahip, programlanabilir, paralel ve çok işlemli hale gelmişlerdir. Şekil 2.5 ve Şekil 2.6'da GPU'ların yıllara göre gelişen durumunu CPU (merkezi işlemci birimi) ile karşılaştırarak göstermektedir [32].



Şekil 2.5. CPU ve GPU için saniyedeki virgüllü sayı işlemlerinin karşılaştırılması



Şekil 2.6. CPU ve GPU belleğin bant genişlikleri karşılaştırması

Şekil 2.6'da görüldüğü üzere ekran kartlarındaki gelişim CPU'ya göre artan bir şekilde hızlanmaktadır.

3. MATERYAL ve YÖNTEM

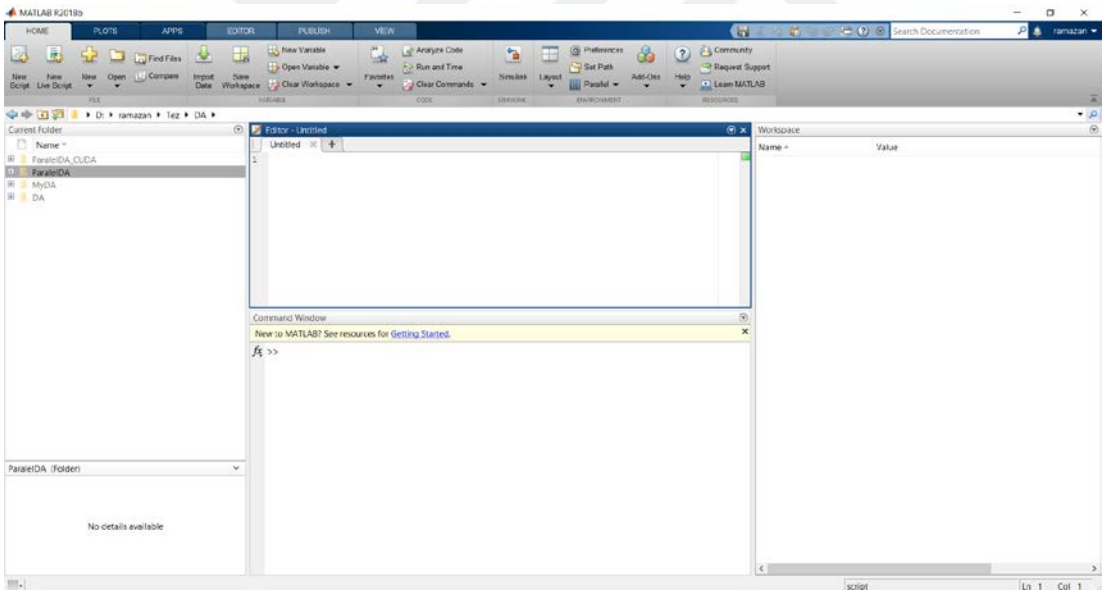
3.1. Materyaller

Bu bölümde, algoritmanın yazılımında kullanılan MATLAB , CUDA ve Visual C++ materyallerine yer verilmiştir..

3.1.1 MATLAB

MATLAB programı Mathworks firması tarafından özellikle mühendisler ve bilim adamları için geliştirdiği bir yazılımıdır. Bilim adamları, araştırmacılar , şirketler vb. çalışmalarını hızlandırmak, analiz ve metotların geliştirme zamanlarını en aza indirmek ve piyasaya daha gelişmiş teknoloji sunmak için MATLAB programını kullanmaktadırlar [33].

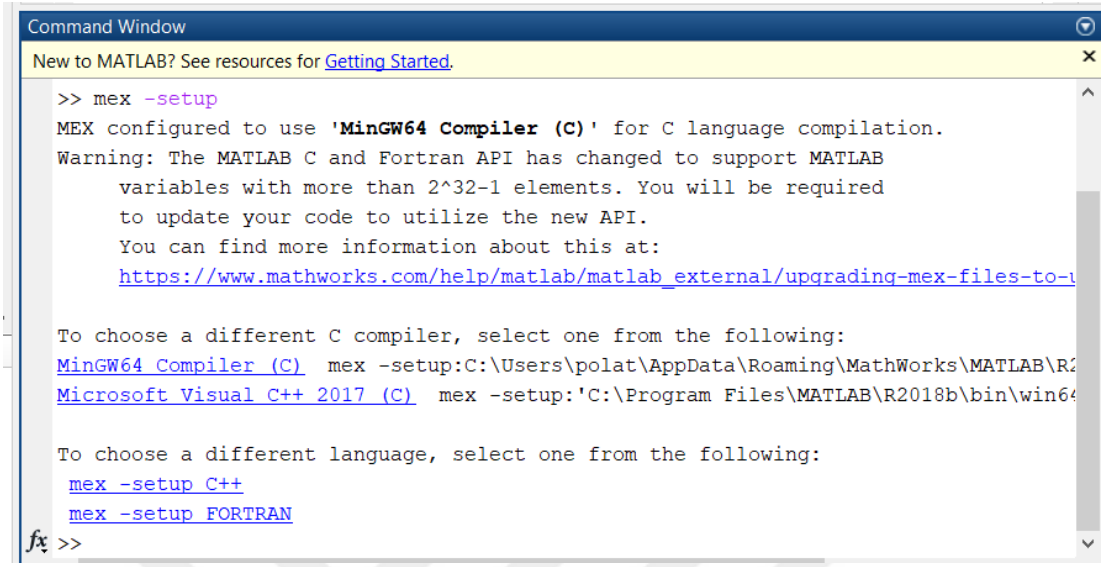
Yapay Zeka, Sinyal İşleme ve Görüntü İşleme derslerinde sıkça kullanılmaktadır. Şekil 3.1'de MATLAB programı akademik kampus versiyonu gösterilmektedir.



Şekil 3.1: MATLAB programı akademik kampus versiyonu

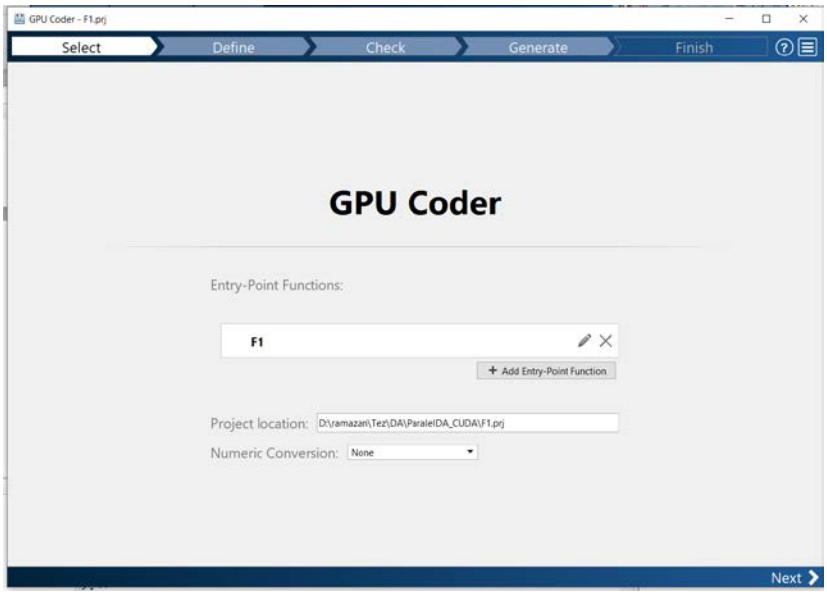
MATLAB ile C , C++, Fortran gibi programlama dili üzerinden fonksiyonlar çağırılabilir ve altprogramları çağırılabilir. TDM-GCC MinGW tarafında yazılan MinGW-w64 yazılımı ile C , C++ ve Fortran üzerinde yazılan fonksiyonlar Matlab üzerinde kullanılabilir Mex dosyalarına dönüştürülmektedir. Bu şekilde derlenen MEX uzantılı dosyalar fonksiyonlar tarafından oluşturulan dinamiksel yüklenebilir nesne dosyaları olarak adlandırılır [34].

Mex-setup komutu ile hangi dilden çevrileceği sisteme tanıtılmaktadır. Şekil 3.2'de mex ayarı için yazılan komut satırı ve seçilebilir yazılım dilleri gösterilmektedir.



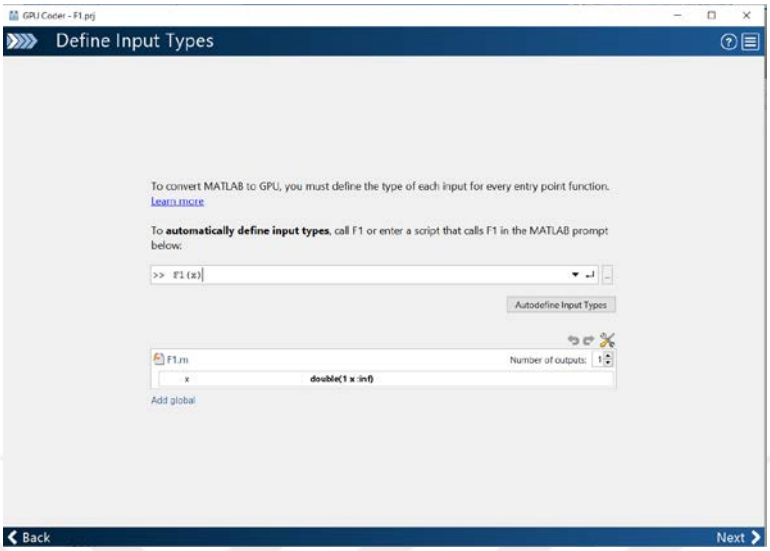
Şekil 3.2 Mex ayarı için yazılan komut satırı ve seçilebilir yazılım dilleri

MATLAB programının özel eklentilerinden olan GPU Coder eklentisi MATLAB tarafından yazılan kodları direk olarak GPU üzerinde kullanılabilirliğini sağlamaktadır. Şekil 3.3'te MATLAB üzerinde GPU Coder eklentisi ile YA testi için kullanılan F1 fonksiyonunun CUDA programlama diline dönüştürme işlemi ana ekranı gösterilmektedir.



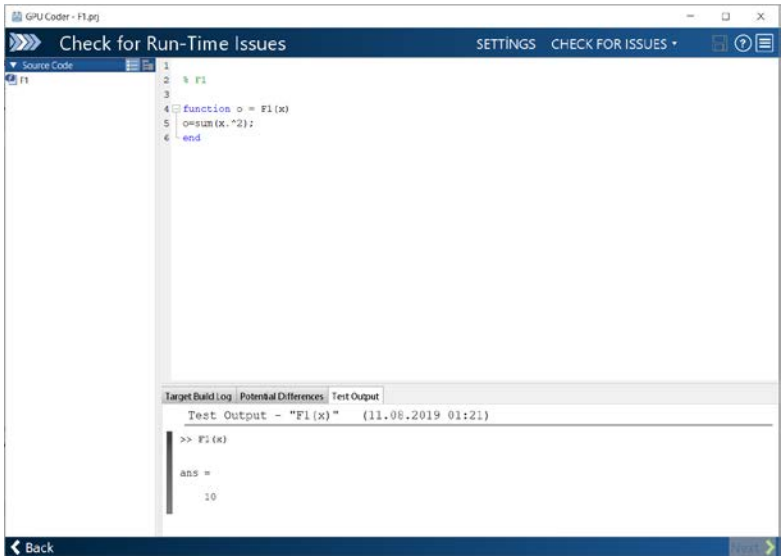
Şekil 3.3 : MATLAB üzerinde GPU Coder eklentisi ile YA testi için kullanılan F1 fonksiyonunun CUDA programlama diline dönüştürme işlemi ana ekranı

Şekil 3.4'te dönüştürme işleminde fonksiyonun parametre ayarı için ekran gösterilmektedir.



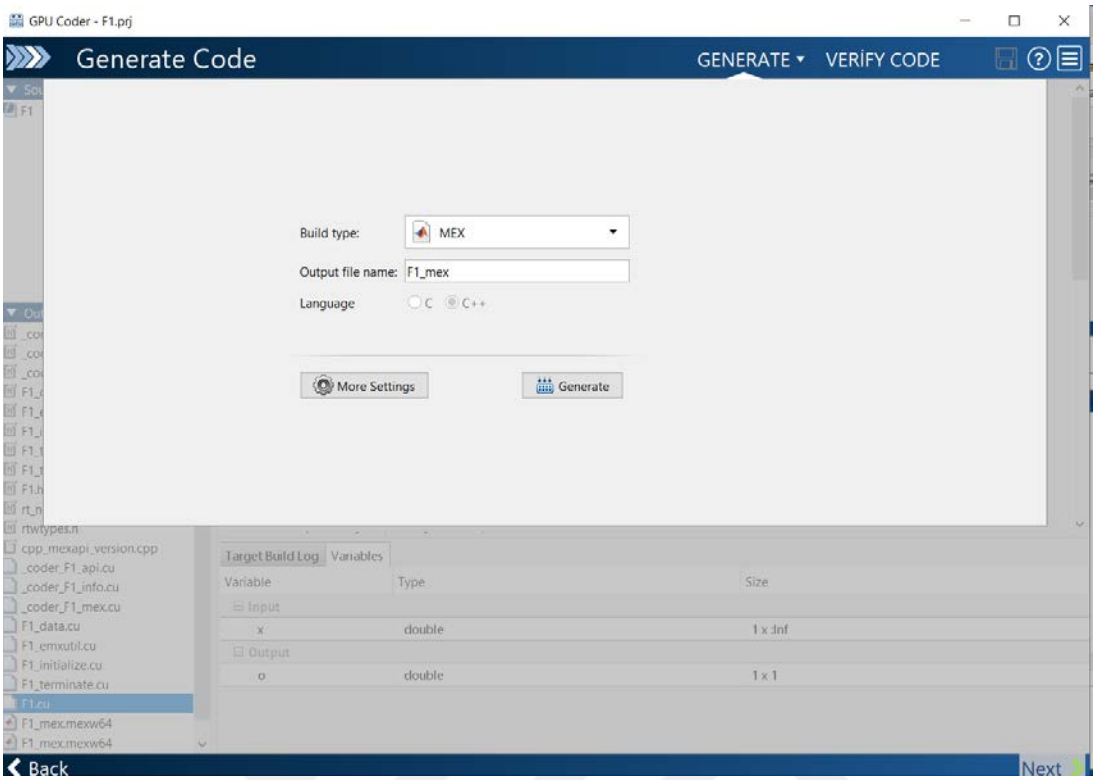
Şekil 3.4 : MATLAB üzerinde GPU Coder eklentisi ile YA testi için kullanılan F1 fonksiyonunun CUDA programlama diline dönüştürme işleminde parametre ayarı

Şekil 3.5'te dönüştürme işleminde fonksiyonun parametre ayarı için fonksiyon testi ekranı gösterilmektedir.



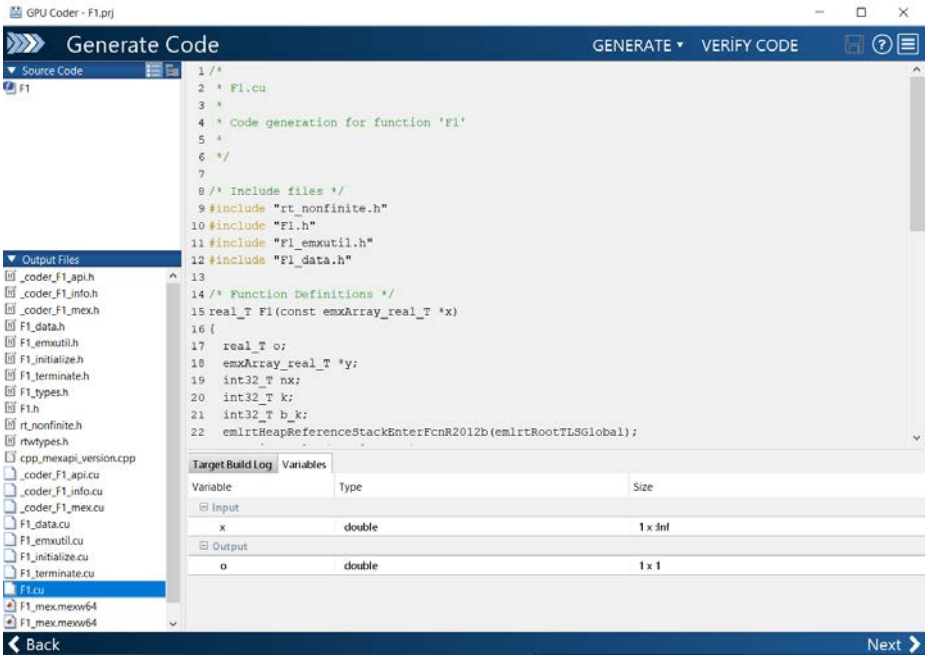
Şekil 3.5 : MATLAB üzerinde GPU Coder eklentisi ile YA algoritmasının testi için kullanılan F1 fonksiyonunun CUDA programlama diline dönüştürme işleminde fonksiyon testi

Şekil 3.6'da dönüştürme işleminde dönüştürme tipleri olan kaynak kod , mex kodu veya direk çalıştırılabilir kod ekranı gösterilmektedir.



Şekil 3.6: MATLAB üzerinde GPU Coder eklentisi ile YA algoritmasının testi için kullanılan F1 fonksiyonunun CUDA programlama diline dönüştürme tipi seçimi

Şekil 3.7'de dönüştürme işleminde Mex halinin kaynak kodları gösterilmektedir.



Şekil 3.7 : MATLAB üzerinde GPU Coder eklentisi ile YA testi için kullanılan F1 fonksiyonunun CUDA programlama diline dönüştürme işleminde mex halinin kaynak kod hali

Şekil 3.8'de dönüştürme işleminde eklenti kullanmadan yazılan kod gösterilmektedir. Parametre olarak kullanılan x' in MATLAB workspace kısmında tanımlı olması gerekmektedir. Aynı zamanda Şekil 3.2'de gösterilen mex -setup işleminin yapılmış olması gerekmektedir.

```
 9 -    cfg = coder.gpuConfig('mex');
10 -    cfg.GpuConfig.CompilerFlags = '--fmad=false';
11 -    codegen -config cfg -args {x} F1
12
```

Şekil 3.8: MATLAB üzerinde GPU Coder eklentisi ile YA testi için kullanılan F1 fonksiyonunun CUDA programlama diline dönüştürme işleminde direk çevrilebilir kod hali

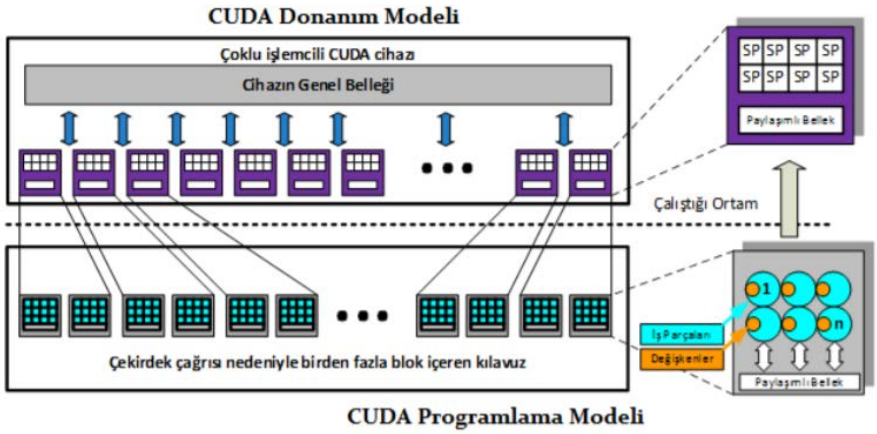
3.1.2 CUDA

CUDA, NVIDIA tarafından grafik işlem birimlerinde (GPU'lar) genel hesaplama için geliştirilen paralel bir bilgisayar platformu ve programlama modelidir . C, C++, Python , C#, Fortran, Java, gibi yazılım dilleri ile yazılan algoritmaların GPU üzerinde çalışmasını sağlayan bir sistem olarak da tanımlanabilir

CPU üzerinde yüksek performans sağlamak hem fiziksel hem de ekonomik olarak çok zordur. CUDA mimarisi sayesinde Genel amaçlı problemlerin paralel olarak çözülebilmesine olanak sağlayan ekran kartlarının üzerinde bulunan çok sayıdaki grafik işlemcisi bulunmaktadır. Genel amaçlı ekran kartları üzerindeki basit işlemcileri kontrol edebilmek amacıyla CUDA kütüphaneleri geliştirilmiş ve bu kütüphanelerin C, Fortran ve OpenCL gibi programlama dillerini desteklemesi sağlanmıştır [35].

CUDA mimarisinde geliştirilen yazılımlar yalnızca GPU üzerinde çalışmazlar. CPU tarafından kontrol edilen ana bellek üzerinden grafik kartı üzerindeki belleğe alınması gereklidir. GPU belleğindeki veri CUDA iş parçacıkları tarafından yürütülerek paralel olarak hesaplanır ve ardından ana belleğe tekrar gönderilerek işlem tamamlanır [35].

Şekil 3.9'de CUDA donanım modeli görülmektedir.



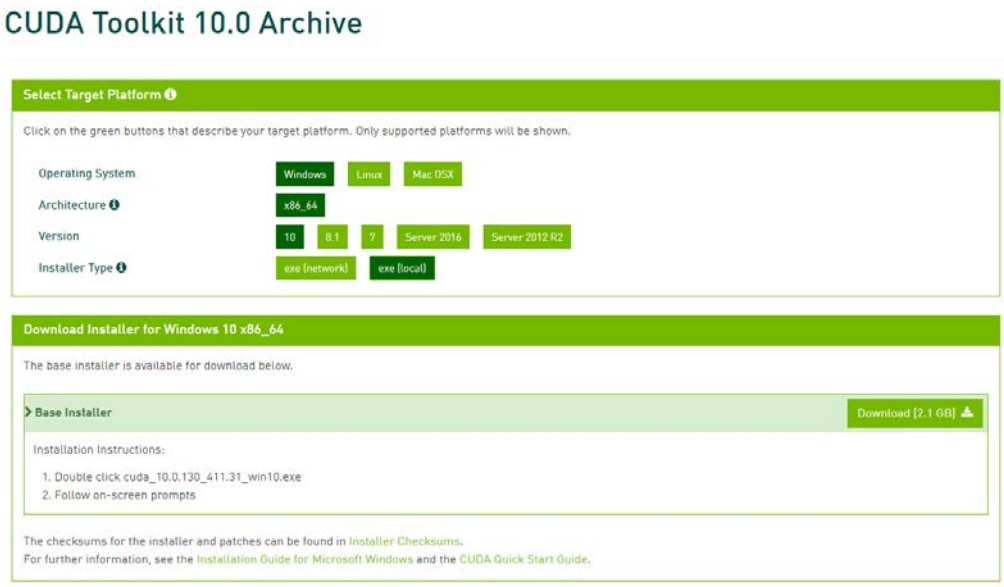
Şekil 3.9 : Çok işlemcili haritalama blokunu gösteren CUDA donanım modeli [36]

Görev paralelliğinde veya veri paralelliğinde performansı iyileştirmek için tek bir GPU'nun belleğine sığmayan verilerle çalışırken birden fazla GPU kullanılmaktadır. Şekil 3.10'da MATLAB programında gpuDevice komut satırı ile GPU sürücüsünün özellikleri gösterilmektedir.

```
Command Window
New to MATLAB? See resources for Getting Started.

>> gpuDevice(1)

ans =

  CUDADevice with properties:

                      Name: 'GeForce GTX 1050 Ti'
                     Index: 1
         ComputeCapability: '6.1'
            SupportsDouble: 1
             DriverVersion: 10.1000
            ToolkitVersion: 9.1000
        MaxThreadsPerBlock: 1024
          MaxShmemPerBlock: 49152
        MaxThreadBlockSize: [1024 1024 64]
               MaxGridSize: [2.1475e+09 65535 65535]
                 SIMDWidth: 32
               TotalMemory: 4.2950e+09
           AvailableMemory: 3.4566e+09
      MultiprocessorCount: 6
             ClockRateKHz: 1620000
              ComputeMode: 'Default'
      GPUOverlapsTransfers: 1
    KernelExecutionTimeout: 1
          CanMapHostMemory: 1
           DeviceSupported: 1
            DeviceSelected: 1

fx >>
```

Şekil 3.10: GPU Sürücüsünün Özellikleri

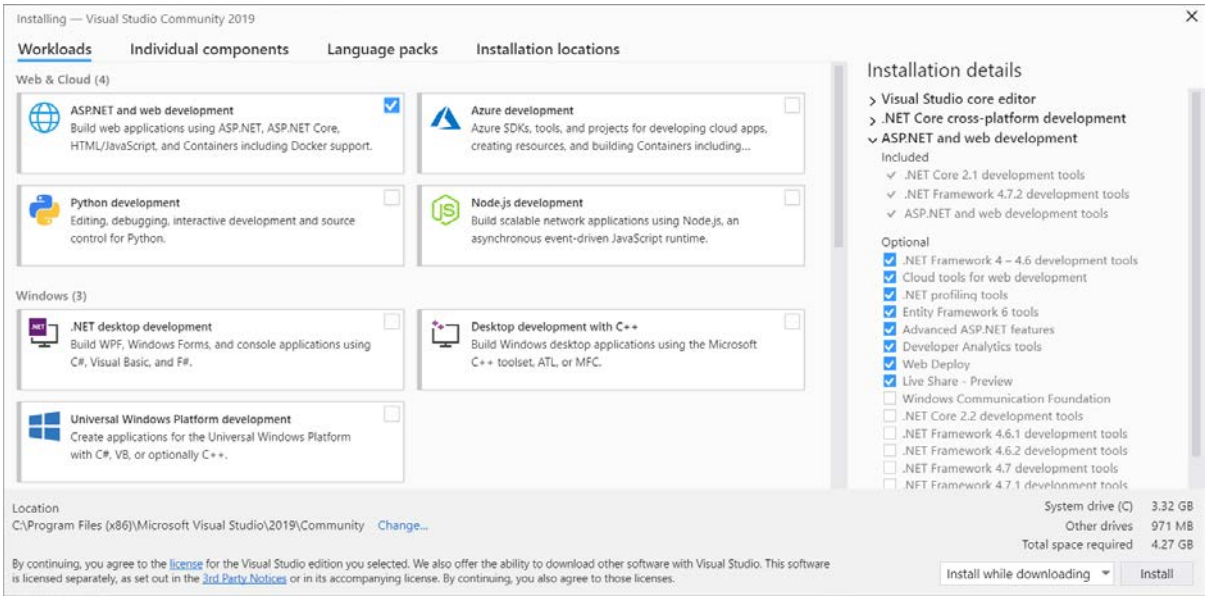
CUDA'dan faydalanabilmek için ekran kartının CUDA'yı destekliyor olması gerekmektedir CUDA kurulumu için CUDA Toolkit 10.0 Archive programını resmi sitesinden indirmemiz gerekmektedir. Şekil 3.11'de Windows sistemi için indirme ekranı gösterilmiştir.



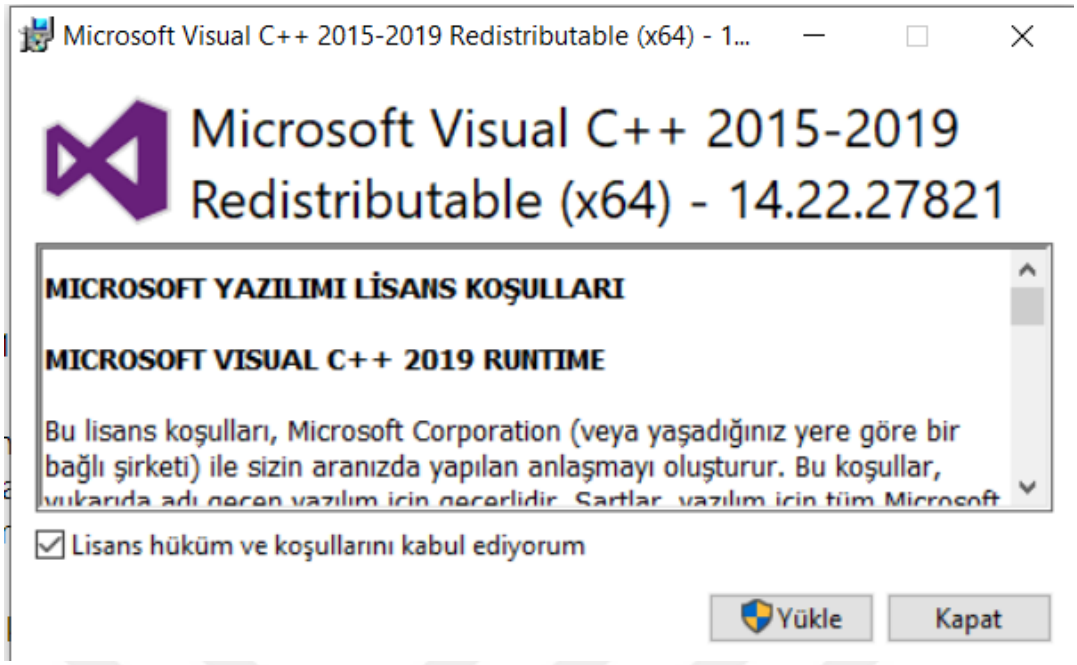
Şekil 3.11 CUDA yazılımı indirme

3.1.3 Visual Studio (Visual C++)

CUDA Toolkit kurulumu esnasında gerekmesi halinde Visual C++ kurulumunu gerçekleştirmeliyiz. Kurulum için Şekil 3.12 veya Şekil 3.13'ten yararlanmaktayız. Şekil 3.12'de Visual Studio resmi web sitesinden indirilmiş olan Visual Studio programının görüntüsü gösterilmektedir. Şekil 3.13'de Microsoft' un resmi web sitesinden indirilmiş olan Visual C++ programın görüntüsü gösterilmektedir.



Şekil 3.12 CUDA yazılımı için Visual Studio kurulumu



Şekil 3.13 CUDA yazılımı için Visual C++ kurulumu

3.1.4 Yusufçuk Algoritması MATLAB kodları

S. Mirjalili tarafından MATLAB firmasının sitesinde YA ve Araç Kutusu olarak yayınlanmıştır. Ayrıca kendisine ait özel web sitesinde de orijinal versiyonu, ikili sürümü ve çok amaçlı versiyonu mevcuttur.

3.1.5 Yusufçuk Algoritmasının CUDA mimarisiyle GPU tabanlı Paralel MATLAB kodları

S. Mirjalili tarafından MATLAB firmasının sitesinde yayınlanan YA ve CAYA algoritmalarının CUDA mimarisiyle GPU tabanlı paralel hali özgeçmiş kısmında belirttiğim kişisel web sitemde mevcuttur.

3.2. Yöntemler

Bu bölümde, tez çalışmasında kullanılan yöntemlere yer verilmiştir. YA 'nın karakteristik yapısından ve çalışma prensiplerinden bahsedilmiştir. Sürüdeki 5 davranış biçimi incelenerek matematiksek olarak modellenmesi anlatılmıştır. YA nın farklı problemlere çözüm olarak sunulan 3 farklı biçimi ile YA , IYA ve CAYA algoritmalarının akış şemasını ve detaylara indiğimizde çalışma mantığını ve daha sonra hangi problemlere çözüm aradığı anlatılmıştır. Tez çalışmasında YA nın önemini incelemek için kullanılan PSO ve YAKA algoritmasından da kısaca bahsedilmiştir.

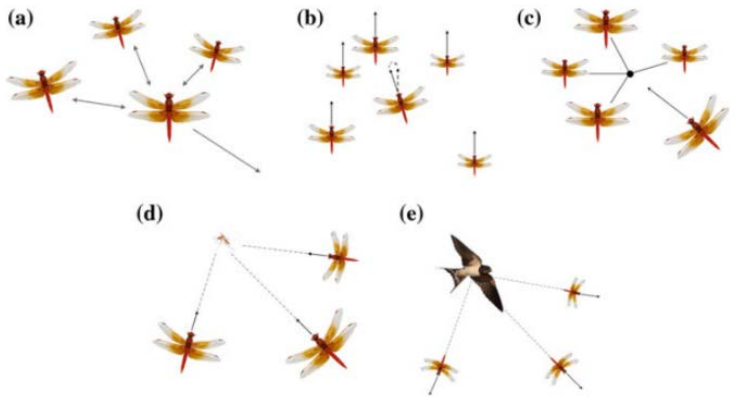
3.2.1. Yusufçuk Algoritması

Büyük birleşik gözleri, güçlü saydam kanatları olan yusufçuk sineği göz alıcı renkleriyle ve uzunca vücutlarıyla tanınır. Dünya da çok farklı türlere sahiptir. Yaşam döngüsü larva ve yetişkin olmak üzere iki önemli evreden oluşmaktadır. Avlanma özelliklerinde eşsiz ve nadir görülen zeki davranışlar görülmektedir. Statik sürüde yusufçuk sinekleri küçük gruplar halinde uçar ve avlanmak için ileri geri hareket ederler. Uçma yolu güzergahında yerel hareketler ve ani değişiklikler statik sürünün ana karakteristik özellikleridir. Bununla birlikte dinamik sürülerde çok sayıdaki yusufçuk büyük grup halinde göç için uzun yolculuk yaparlar [6].

Reynold'a göre sürü davranışları üç temel prensip içerisindedir [9]. Ayırma Komşuluk içerisindeki diğer kişiler ile çarpışmayı önlemek anlamına gelmektedir. Hizalama; Komşuluk içerisindeki diğer bireyler ile hız uyumunu göstermektedir. Ahenk; Komşuluk kütlesinin merkezine doğru bireylerin eğilimine atıf yapmaktadır Herhangi bir sürünün asıl amacı hayatta kalabilmektir ve bundan dolayı bireyler gıda kaynaklarına doğru yönelmelidirler. Ayrıca bu ana eylemin yanında sürü dışarıdan düşmanları tarafından rahatsız edilebilirler. Bu iki davranış da eklenirse, konum güncelleme işleminde beş ana faktör kullanılmaktadır.

Sürünün bu davranışı formüle edilerek optimizasyon problemlerinin çözümünde kullanılmaktadır

Şekil 3.14'de sürü içerisinde bireyler arasındaki beş temel faktör gösterilmektedir.
a Ayırma ,b Hizalama, c Ahenk, d Besin, e Düşman



Şekil 3.14 Sürü içerisinde bireyler arasındaki beş temel faktör [6]

Bu davranışların her birini matematiksel olarak aşağıdaki gibi modellenmektedir

$$S_i = -\sum_{j=1}^{N} X - X_j \tag{3.1}$$

Denklem (3.1) deki X mevcut bireyin pozisyonu, Xj ise j. komşu bireyin pozisyonunu göstermektedir. N komşu bireylerin sayısını vermektedir.

$$A_i = \frac{\sum_{j=1}^{N} V_j}{N} \quad (3.2)$$

Denklem (3.2) de yer alan Vj değeri j. komşu bireyin hızını göstermektedir.

$$C_i = \frac{\sum_{j=1}^{N} X_j}{N} - X \quad (3.3)$$

Besin kaynağına doğru çekim Denklem (3.3) de gibi hesaplanmaktadır.

$$F_i = X^+ - X \quad (3.4)$$

Denklem (3.4) deki X+ ifade besin kaynağının pozisyonunu göstermektedir. Bir düşman tehlikesi karşısındaki davranış biçimi aşağıdaki gibi hesaplanmaktadır.

$$E_i = X^- + X \quad (3.5)$$

Denklem (3.5) deki X- ifadesi tehdit eden düşmanın pozisyonunu göstermektedir.

$$\Delta X_{t+1} = (sS_i + aA_i + cC_i + fF_i + eE_i) + w\Delta X_t \quad (3.6)$$

Yapay yusufçuk sineklerinin pozisyonlarını güncellemek ve onların hareketlerine benzetmek için ΔX adım ve X pozisyon olmak üzere iki vektör düşünülmüştür.

Denklem (3.6) de ki s parametresi ayırma, a parametresş hizalama, c parametresi uyum, f parametresi besin, e parametresi ise düşman faktörlerinin ağırlık değerlerini temsil etmektedir. Ayrıca w değeri t. döngüdeki atalet ağırlığını göstermektedir.

$$X_{t+1} = X_t + \Delta X_{t+1} \quad (3.7)$$

Adım vektörünü hesapladıktan sonra aşağıda Denklem (3.7) de yer verilen pozisyon vektörü hesaplanmaktadır.

$$X_{t+1} = X_t + Levy(d) \times X_t \quad (3.8)$$

Rassallığı, stokastik davranışı ve yusufçukların keşif özelliklerini geliştirmek için rastgele yürüyüş (Le'vy uçuşu) kullanılmıştır. Bu durumda yusufçukların pozisyonları aşağıdaki eşitlik kullanılarak güncellenmiştir

Denklem (3.8) deki t parametresi mevcut döngüyü, d parametresi ise pozisyon vektörünün boyutunu temsil eder.

$$Levy(x) = 0.01 \times \frac{r_1 \times \sigma}{|r_2|^{1/\beta}} \quad (3.9)$$

Denklem (3.9) da Le'vy uçusu verilmiştir.

$$\sigma = \left(\frac{\tau(1+\beta) \times \sin\left(\frac{\pi\beta}{2}\right)}{\tau\left(\frac{1+\beta}{2}\right) \times \beta \times 2^{\left(\frac{\beta-1}{2}\right)}} \right)^{1/\beta} \tag{3.10}$$

$$\tau(x) = (x-1)! \tag{3.11}$$

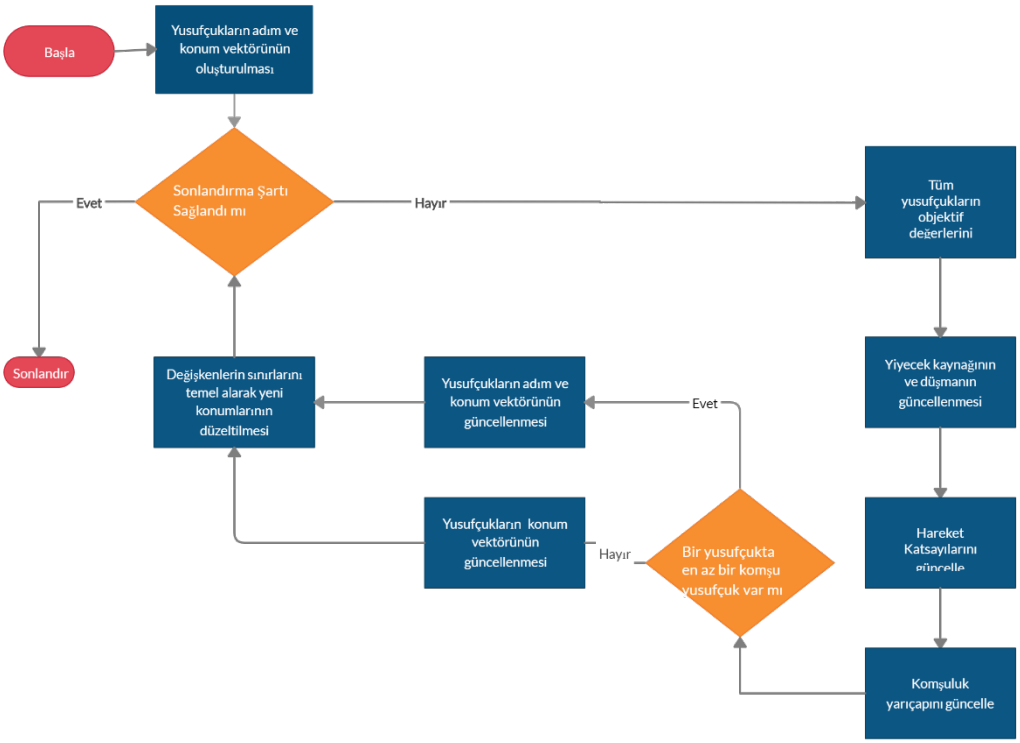
$$T(x) = \left| \frac{\Delta x}{\sqrt{\Delta x^2 + 1}} \right| \tag{3:12}$$

$$X_{t+1} = \begin{cases} -X_t, & r < T(\Delta X_{t+1}) \\ X_t, & r \geq T(\Delta X_{t+1}) \end{cases} \tag{3:13}$$

Denklem (3.9 - 3.13) da Le'vy uçusu parametreleri gösterilmektedir.

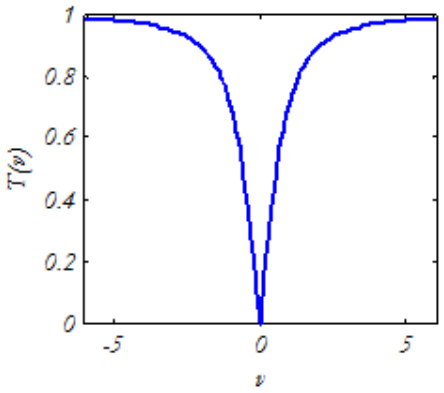
3.2.1.1. Tek Amaçlı Problemler için Yusufçuk Algoritması

Yalnızca sürekli problemlerin çözümünü sağlayan bir algoritma olarak geliştirildi. Şekil 3.15, YA 'nın akış şemasını göstermektedir:



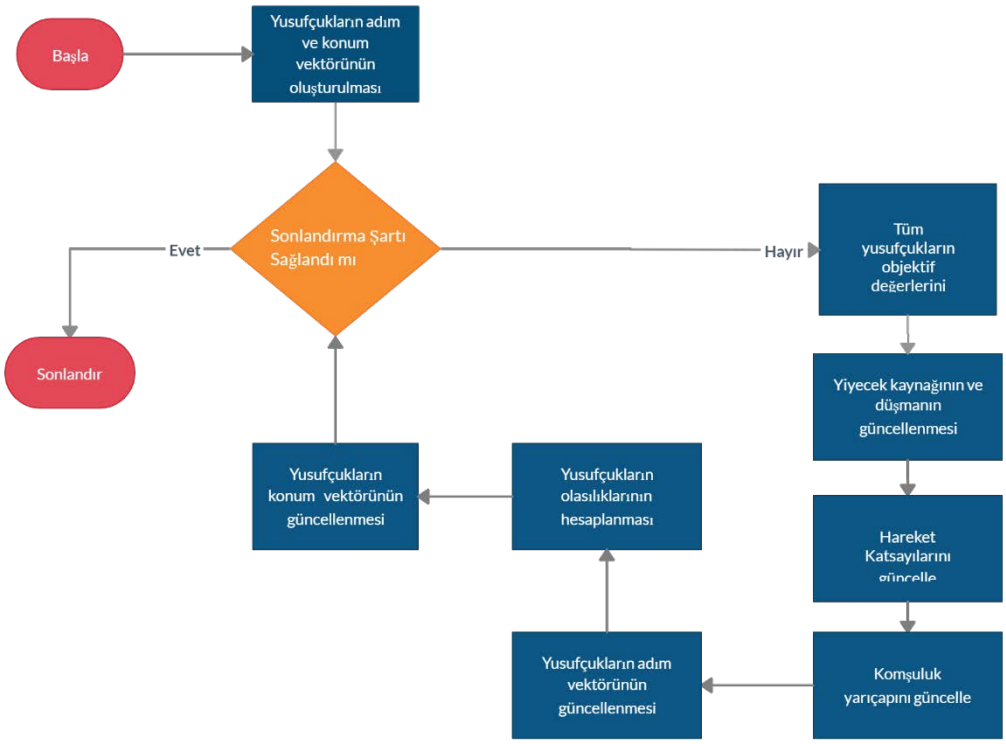
Şekil 3.15. YA ' ya ait akış şeması

YA yalnızca sürekli problemleri çözebildiğinden , ayrık problemleri çözmek için İkili Yusufcuk Algoritması (IYA) adı verilen algoritma geliştirilmiştir. Bu algoritmanın tasarımında Şekil 3.16'te gösterilen v şeklinde bir transfer fonksiyonu kullanılmıştır [2].



Şekil 3.16: IYA ' nın transfer fonksiyonu [2].

Şekil 3.17' de IYA ' nın akış şemasını göstermektedir:

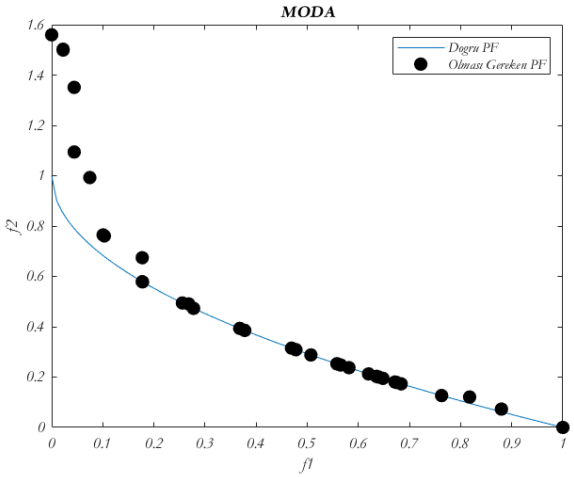


Şekil 3.17. IYA ' nın akış şeması

3.2.1.2. Çok Amaçlı Problemler için Yusufçuk Algoritması

Çok amaçlı problemleri çözmek için, ilk olarak, optimizasyon sırasında gerçek Pareto-optimal cephesine ait en uygun çözümleri saklamak ve almak için bir arşiv bulunur. Yusufcukların pozisyonunun güncellenmesi YA ile aynıdır, ancak gıda kaynakları arşivden seçilir. Pareto optimal cephesine iyi yayılmış olan çözümlemeleri bulmak için, literatürdeki Çok Amaçlı Parçacık Sürüsü Optimizasyonu (MOPSO) algoritmasına benzer şekilde, elde edilen Pareto optimal cephesinin en az nüfuslu bölgesinden bir gıda kaynağı seçilir. Pareto optimal cephesinin en az nüfuslu alanını bulmak için, arama alanının bölümlere ayrılması gerekir [17].

Şekil 3.18 'de ZDT1 fonksiyonunun örnek çözümü için pareto optimal ve pareto optimal olmayan değerleri gösterilmektedir. Çizgi üzerinde gösterilen noktalar pareto optimal çözümlerdir.



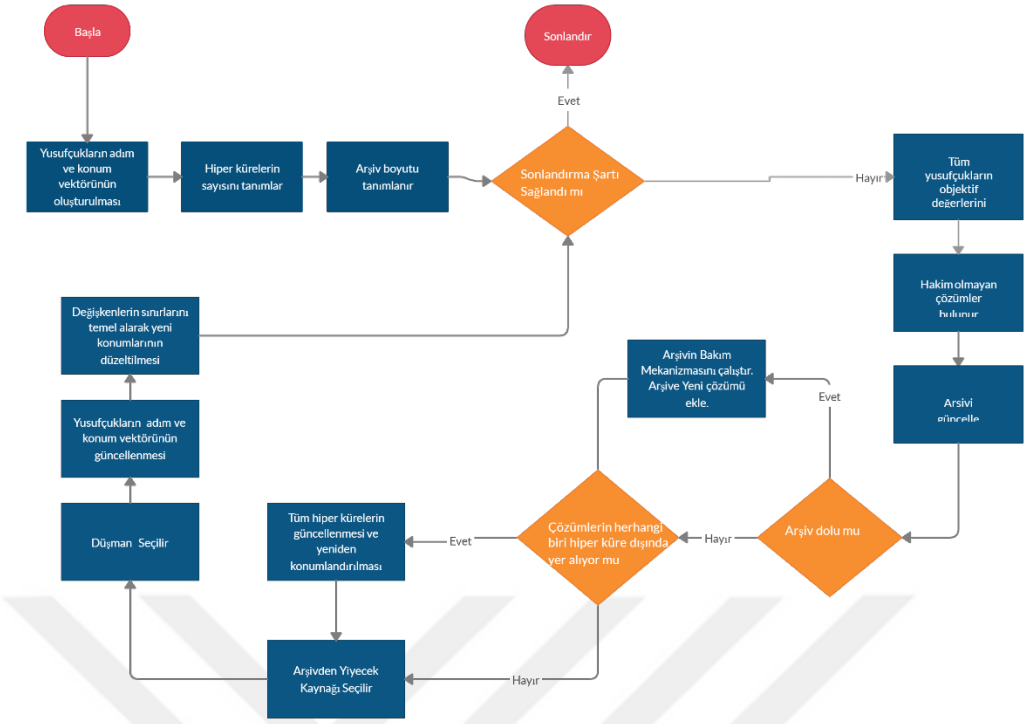
Şekil 3.18 : ZDT1 fonksiyonunun pareto optimal çözümü

Bu, elde edilen Pareto optimal cephesindeki çözümlerin en iyi ve en kötü olanlarını bulmak, tüm çözümleri kapsayacak bir hiper küreyi tanımlamak ve hiper küreleri her bir yinelemede eşit alt hiper kürelere bölmek suretiyle yapılır. Segmentlerin yaratılmasından sonra, seçim rulet tekerleği mekanizması ile yapılır. Bu mekanizma, Çok Amaçlı Problemler için Yusufçuk Algoritması 'nın (CAYA) daha az doldurulmuş segmentlerden gıda kaynaklarını seçme olasılığını arttırır. Bu nedenle, yapay yusufçuklar bu bölgelerin etrafında uçmaya ve tüm Pareto optimal cephesinin dağılımını iyileştirmeye teşvik edilecektir. Bununla birlikte, arşivden düşmanları seçmek için, yapay yusufçukların gelecek vaat eden kalabalık bölgeleri aramasını engellemek için en kötü (en kalabalık) hiper kürenin seçilmesi gerekir. Seçim yine rulet tekerleği mekanizması ile yapılır.

Arşiv, her yinelemede düzenli olarak güncellenmeli ve optimizasyon sırasında doldurulabilir. Bu nedenle arşivi yöneten bir mekanizma olmalı. Eğer arşivden en az birine hükmediyorsa arşive girmesi engellenmelidir. Eğer bir çözüm arşiv içerisindeki Pareto optimal çözümlerinin bazılarına hükmediyorsa, hepsinin arşivden kaldırılması ve çözümün arşive girmesine izin verilmesi gerekir. Bir çözüm, arşivdeki tüm çözümlerle ilgili olarak hâkim değilse, arşive eklenmelidir. Arşiv doluysa, arşivdeki yeni çözümleri barındırmak için en kalabalık bölümlerden bir veya daha fazla çözüm kaldırılabilir [37].

CAYA 'nın tüm parametreleri, maksimum hiper küre sayısını ve arşiv boyutunu tanımlamak için iki yeni parametre dışında YA parametreleriyle aynıdır [2].

Şekil 3.19'da CAYA 'nın akış şemasını göstermektedir:

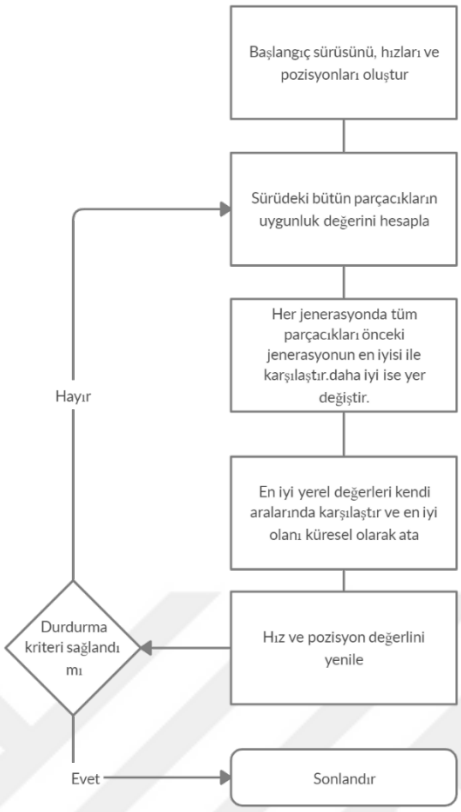


Şekil 3.19. CAYA nın akış şeması

3.2.2. Parçacık Sürü Optimizasyon Algoritması

1995 yılında Dr.Eberhart ve Dr.Kennedy tarafından geliştirilen Parçacık Sürü Optimizasyon Algoritması (PSO) bir sürü tabanlı sezgisel optimizasyon algoritmasıdır. Kuşların hareketlerinden esinlenmiştir. Sürüdeki üyelerin tek olarak yapamadıkları bir işi, birlikte hareket ederek yapmaktadır. Bir yandan kendine ait bilgilerini grubun bilgisinden yararlanıp geliştirirken diğer yandan da grup bilgisini optimize ederken elde edilen ortak bilgiye göre toplu yönelim sağlayarak hedefe doğru bir şekilde yönelmektedir [38].

Şekil 3.20'de PSO algoritması akış diyagramı gösterilmektedir.



Şekil 3.20. PSO algoritması akış diyagramı

3.2.3. Yapay Arı Kolonisi Algoritması

2005 yılında sayısal optimizasyon problemini çözmek için D. Karaboğa [21] tarafından geliştirilen YAKA algoritması, arıların yiyecek bulma tekniğinin modellendiği sürü tabanlı bir arama algoritmasıdır. B. Ata çalışmasında ters sarkaç için doğrusal bir ikinci dereceden optimal kontrolör tasarlamak için uygun ağırlık matrislerinin belirlenmesinde YAKA algoritmasını kullanmıştır [14].

Şekil 3.21'de YAKA algoritması akış diyagramı gösterilmektedir.

Şekil 3.21. YAKA algoritması akış diyagramı

3.3. Paralelleştirme İşlemi

YA iki ana aşama olarak veri paralelliği ve görev paralelliği çerçevesinde uygulanmıştır. Veri paralelliğini sağlamak için algoritmada her adımda üretilerek kullanılan rastgele sayılar toplu halde GPU belleğine gönderilmiş, her iterasyonda rastgele sayı üretmek için bir çağrı yapmak zorunda olan iş parçacıkları bu iş yükünden kurtarılarak hız kazanımı elde edilmiştir. Yusufcukların amaç fonksiyon değerlerinin bellekte yerleşimi ile en iyi yusufcugun parametrelerinin bellekte yerleşimi hem seri hem de paralel uygulamada aynı şekildedir. Görev paralelliği çerçevesinde ise kıyas fonksiyonları, yusufcukların pozisyonları , besin kaynağı ve düşmanın pozisyonlarının oluşturulması paralel olarak yeniden kodlanmış, adım adım işlenen süreçler eş zamanlı işlem görecek şekilde ayarlanmıştır. Bu çalışmada görev paylaşımlı bağlamında MATLAB'ın paralel döngü(parfor) yapısı kullanılmıştır. Paralel for döngüsü normal bir for döngüsünün yaptığı işi paralel yapıda gerçekleştirmektedir [39].

Kıyas fonksiyonları seri olarak Matlab ortamında yazılmıştır ve algoritmadaki yusufcuklar için aynı Matlab fonksiyon dosyası kullanılmıştır. Paralel uygulamada Yusufcuklar için Matlab ortamında Şekil 3.8'de gösterilen kod yardımıyla ve Şekil 3.3'te gösterilen GPU-Coder eklentisi yardımıyla ayrı ayrı fonksiyon dosyaları MEX uzantılarına dönüştürülmüştür. Üretilen MEX dosyaları da Matlab uygulamamızda kullanılmıştır.

4. BULGULAR ve TARTIŞMA

Bu bölümde YA' nın üstün performansı hem sonuç bazlı hem de süre bazlı incelenmiştir. Grafiksel ve tablo halinde detaylı olarak analiz edilmiştir.

İlk olarak sonuç odaklı performans için seri hali ile PSO, YAKA ve YA algoritmalarıyla yapılan çalışmadan elde edilen sonuçlar karşılaştırılarak Tablo 4.1 de verilmiştir. Süre odaklı çalışma için ise YA nın CPU ile seri ve paralel , GPU ile paralel işlemlerinin sonuçları ise Tablo 4.2 de verilmiştir.

Tablo 4.1. 500 döngüde gerçekleştirilen optimizasyon algoritmaları test sonuçları

	YA		PSO		YAKA	
	Min.	Ort.	Min.	Ort.	Min.	Ort.
F1	0,00E+00	2,70E+00	5,00E-16	1,13E-15	-2,48E-01	1,23E-01
F2	0,00E+00	9,92E-01	1,96E-10	1,05E+02	-3,61E-02	6,45E-03
F3	1,44E+00	3,58E+01	6,27E-15	8,47E-14	-3,89E-01	-2,59E-02
F4	1,24E-01	1,15E+00	4,24E-11	1,08E-10	-1,27E-01	7,99E-02
F5	8,78E+00	2,83E+02	2,90E-11	3,12E+02	-2,59E+01	-2,45E+00
F6	2,16E-08	8,25E-01	2,41E-15	5,69E-15	-6,93E-01	-3,84E-01
F7	4,46E-03	2,60E-02	5,40E-04	1,27E-03	-2,36E-01	-5,78E-02
F8	-3,65E+03	-3,03E+03	-1,80E+308	6,55E+04	4,18E+02	4,21E+02
F9	8,96E+00	2,24E+01	1,99E+00	3,68E+00	-2,04E-02	2,50E-03
F10	7,99E-15	1,48E+00	2,00E+01	2,00E+01	-1,42E-01	5,60E-03

4.1. Parçacık sürü optimizasyon algoritmasına yönelik bulgular

500 kuş sürüsü ile PSO'ya yönelik karşılaştırmada F1 ile F10 arasındaki tüm fonksiyonlar için Tablo 4.1'te 500 döngüde uygulanan algoritma 10 defa teste tabi tutulup sonuçları gösterilmiştir.

Tablo 4.1'te 500 yusufcuk ve 500 kuş sürüsü ile tüm fonksiyonlar üzerinden yapılan test işleminde PSO'nun F1,F3,F4,F7 ve F9 ile %50 başarı oranı gözlemlenmiş olup diğer fonksiyonlarda da orjinal YA nın % 50 başarısı gözlemlenmiştir

4.2. Yapay arı optimizasyon algoritmasına yönelik bulgular

500 arı sürüsü ile YAKA algoritmasına yönelik karşılaştırmada F1 ile F10 arasındaki tüm fonksiyonlar için Tablo 4.1'te 500 döngüde uygulanan algoritma 10 defa teste tabi tutulup sonuçları gösterilmiştir.

Tablo 4.1'te 500 yusufcuk ve 500 arı sürüsü ile tüm fonksiyonlar üzerinden yapılan test işleminde YAKA algoritmasının F1, F2, F3, F4, F5, F9 ve F10 ile %70 başarı oranı gözlemlenmiş olup diğer yandan da orjinal YA nın % 30 başarısı gözlemlenmiştir.

4.3. Paralelleştirmeye yönelik bulgular

Tek ve Çok amaçlı test fonksiyonlarında karşılaştırma süre bazlı yapılmıştır.

4.3.1. Tek Amaçlı Test Fonksiyonları üzerinde Paralelleştirmeye yönelik bulgular

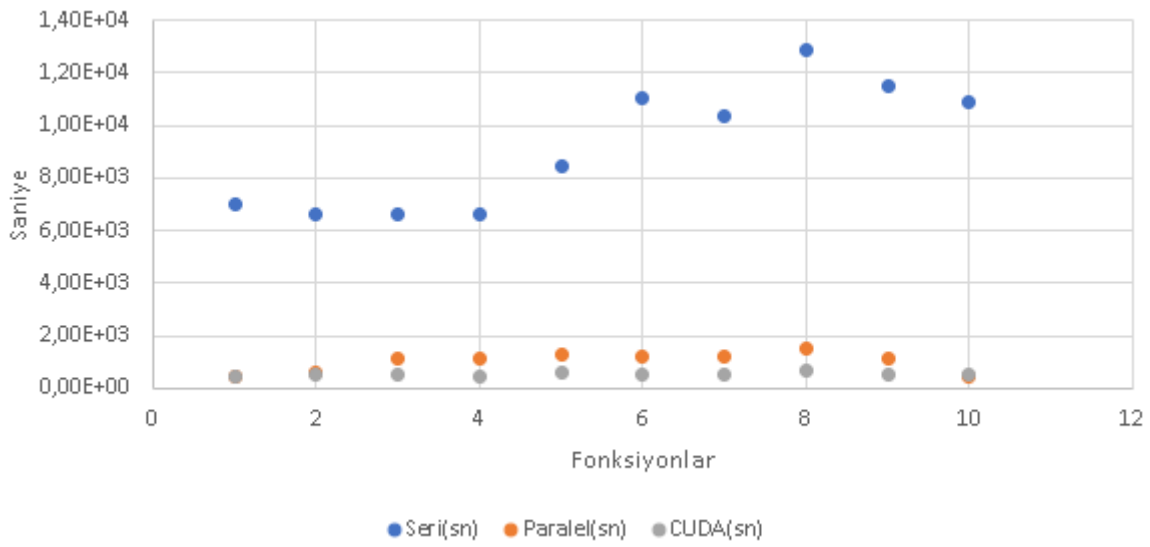
500 yusufçuk ile paralelleştirmeye yönelik karşılaştırma için tek amaçlı test fonksiyonlarında Tablo 4.2'de 500 döngüde uygulanan algoritma 10 defa hem CPU üzerinde hem de GPU üzerinde paralel olarak teste tabi tutulup sonuçları gösterilmiştir.

Tablo 4.2. 500 döngüde 500 Yusufçuk ile 10 defa Paralel YA Testi

Fonksiyonlar	CPU-Seri (saniye)	CPU-Paralel (saniye)	GPU-CUDA (saniye)
F1	7.029,87	448,17	443,76
F2	13.660,50	1.054,60	986,92
F3	20.251,45	2.213,35	1.467,76
F4	26.863,26	3.338,45	1.912,26
F5	35.336,06	4.621,52	2.537,01
F6	46.409,88	5.805,17	3.034,92
F7	56.732,39	6.998,58	3.539,47
F8	69.567,06	8.522,70	4.211,70
F9	81.037,07	9.649,62	4.741,81
F10	91.942,03	10.083,93	5.240,44
	%100	**%10,9**	**%5,7**

Tablo 4.2'de görüldüğü üzere 500 yusufcuk ile tüm fonksiyonlar için seri işlemlerde toplam 91.942 saniye, CPU üzerinde 10.083 saniye ,CUDA mimarisiyle GPU tabanlı paralel işlemlerde 5.240 saniye sürmüştür. Paralel olarak ele alınan YA 'nın, CPU üzerinde karşılaştırdığımızda sistem 9 kat, GPU üzerinde ise 20 kat hızlanmıştır

Grafiksel bir biçime dönüştürüldüğünde aşağıdaki Şekil 4.1 elde edilir. Şekil 4.1'de 500 döngüde 500 Yusufçuk ile 10 defa Paralel YA Testi grafiği gösterilmektedir.



Şekil 4.1. 500 döngüde 500 Yusufçuk ile 10 defa Paralel YA Testi

Mavi ile işaretlenen noktalarda CPU üzerindeki seri işlemlerin , turuncu ile işaretlenen noktalarda CPU üzerindeki paralel işlemlerin ve gri ile işaretlenen noktalarda ise GPU üzerindeki CUDA mimarisi ile yapılan paralel işlemlerin saniye cinsinden sonuçları gösterilmiştir.

4.3.2. Çok Amaçlı Test Fonksiyonları üzerinde Paralelleştirmeye yönelik bulgular

500 yusufçuk ile paralelleştirmeye yönelik karşılaştırma için çok amaçlı test fonksiyonlarında Tablo 4.3'de 500 döngüde uygulanan algoritma 10 defa hem CPU üzerinde hem de GPU üzerinde paralel olarak teste tabi tutulup sonuçları gösterilmiştir.

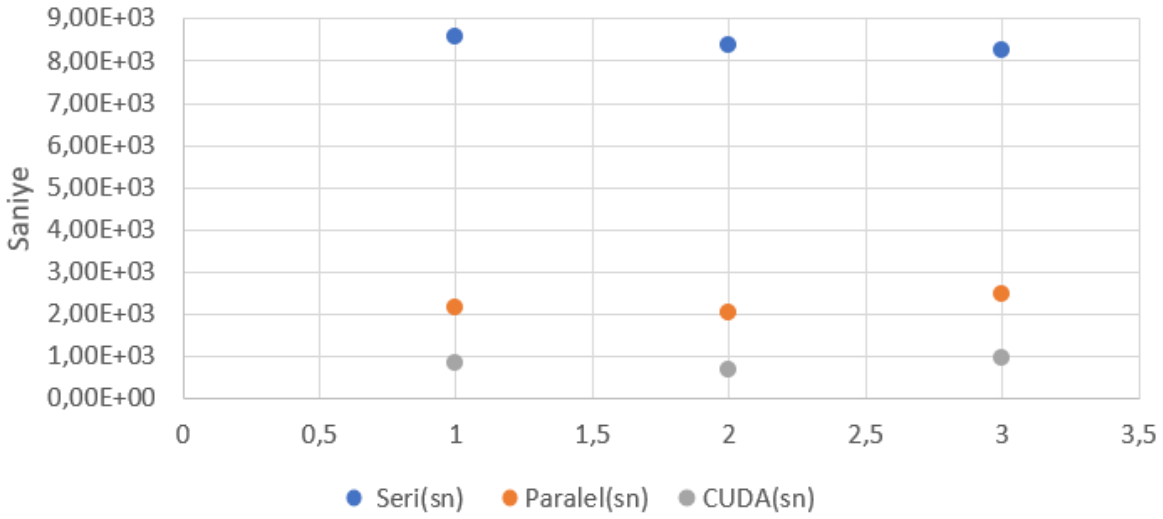
Tablo 4.3. 500 döngüde 500 Yusufçuk ile 10 defa Paralel CAYA Testi

Fonksiyonlar	CPU-Seri (saniye)	CPU-Paralel (saniye)	GPU-CUDA (saniye)
ZDT1	8.569,47	2.136,25	846,33
ZDT3	8.369,49	2.028,38	667,85
ZDT4	8.266,81	2.448,57	956,34

Tablo 4.3'de görüldüğü üzere 500 yusufcuk ile tüm fonksiyonlar için seri işlemlerde toplam 25.205 saniye, CPU üzerinde 6.613 saniye, CUDA mimarisiyle GPU tabanlı paralel işlemlerde 2.470 saniye sürmüştür. Paralel olarak ele alınan CAYA 'nın, CPU üzerinde karşılaştırdığımızda sistem 4 kat, GPU üzerinde ise 10 kat hızlanmıştır.

Grafiksel bir biçime dönüştürüldüğünde aşağıdaki Şekil 4.2 elde edilir. Mavi ile işaretlenen noktalarda CPU üzerindeki seri işlemlerin , turuncu ile işaretlenen noktalarda CPU üzerindeki paralel işlemlerin ve gri ile işaretlenen noktalarda ise GPU üzerindeki CUDA mimarisi

ile yapılan paralel işlemlerin saniye cinsinden sonuçları gösterilmiştir. Şekil 4.2'de 500 döngüde 500 Yusufçuk ile 10 defa Paralel CAYA Testi grafiği gösterilmektedir.



Şekil 4.2. 500 döngüde 500 Yusufçuk ile 10 defa Paralel CAYA Testi

5. SONUÇLAR ve ÖNERİLER

Bu tez çalışmasında doğa esinli metasezgisel optimizasyon algoritmalarının en başarılı ve en yenilerinden birisi olan YA optimizasyon algoritmasının PDB mimarileri üzerinde çalışılmıştır.

Bu tez çalışmasının amacı, YA 'nın, paralel programlama teknikleri kullanarak zaman ve bellek açısından daha yüksek başarımlı hale getirebilmektir.

GPU programlama tekniklerini kullanarak düşük maliyet ve yüksek performans ile daha hızlı çözüm üreten YA 'nın paralelleştirilmesi incelenmiştir. YA ' nın önemini belirtmek adına en önemli optimizasyon algoritmalarından olan Parçacık Sürü Optimizasyonu (PSO) ve Yapay Arı Kolonisi (YAKA) optimizasyon algoritmaları ile kıyaslanmıştır. YA seri ve paralel düzeyde olmak üzere CPU ve GPU üzerinde süre odaklı incelenmiştir. Algoritmalar her düzeyde 10 defa çalıştırılmış olup sonuçların ortalaması ile verimlilikleri karşılaştırılmış , süreleri toplamı ile hızları karşılaştırılmıştır. Test fonksiyonlarında problemin boyutu 10 olarak ele alınmış ve 500 yusufçuğun 500 itarasyonda performans analizi yapılmıştır. Seri olarak çalıştırılılan YA 91.942 saniye sürmüştür. Paralel olarak kodlanan YA CPU üzerinde 10.083 saniye iken GPU üzerinde 5.240 saniye sürmüştür. Paralel olarak ele alınan YA 'nın, CPU üzerinde karşılaştırdığımızda sistem 9 kat, GPU üzerinde ise 20 kat hızlanmıştır. Paralel olarak kodlanan CAYA CPU üzerinde 25.205 saniye iken GPU üzerinde 2.470 saniye sürmüştür. Paralel olarak ele alınan CAYA 'nın CPU üzerinde karşılaştırdığımızda sistem 4 kat, GPU üzerinde ise 10 kat hızlanmıştır. Sonuçları göz önüne aldığımızda optimizasyon problemlerinin daha hızlı çözülmesi için YA'nın GPU üzerinde çalışan CUDA tabanlı versiyonunun kullanılabilir oldugunu göstermektedir. Paralelleştirme süresi içinde CUDA mimarisi ile zamansal anlamda çok ciddi kazançlar elde edilmektedir. İşlemciler ve Grafik kartları giderek çok çekirdekli bir hal almasından dolayı NVIDIA firmasının güçlü desteği sürmektedir. MATLAB ortamında akademik çalışmaların artması ile GPU-Coder ve Paralel Hesaplama Araçları gibi kütüphanelerin kodlama açısından işlemlerin kolaylaştırılması sağlanmaktadır.

YA nın paralel olarak CUDA mimarisi ile çözümü uzun zaman alan gerçek dünya problemlerinde oldukça yararlı olabileceği öngörülmektedir.

KAYNAKLAR

[1]. Çelik, A., (2010). *Paralel bilgisayar sistemlerinin performans analizi Performance analysis of parallel computer systems.* Y. Lisans tezi, Dumlupınar Üniversitesi, Kütahya.

[2]. Mirjalili, S., (2016). *Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems.* Neural Comput. Appl., vol. 27, no. 4, p. 1053–1073

[3]. Aci, C. İ., (2019). *A Modified Dragonfly Optimization Algorithm for Single- and Multiobjective Problems Using Brownian Motion Comput*. Intell. Neurosci., vol. 2019, p. 1–17

[4].Mafarja, M. M., Eleyan, D., Jaber, I., Hammouri, A., Mirjalili, S. (2017). *Binary Dragonfly Algorithm for Feature Selection*. ICTCS 2017, vol. 2018-Janua, p. 12–17

[5].Vikram, K. A., Ratnam, C., Lakshmi, V., Sunny K., Ramakanth. R. (2018) *Application of dragonfly algorithm for optimal performance analysis of process parameters in turn-mill operations- A case study*. IOP Conf. Ser. Mater. Sci. Eng., vol. 310, no. 1, p. 012154

[6]. Katircioğlu, F. (2017) *Renkli Görüntüler İçin Yusufçuk Algoritması Kullanılarak Benzerlik Görüntüsüne Dayalı Eşikleme.* Düzce Üniversitesi, Bilim ve Teknoloji Dergisi p. 506-523

[7]. Kıran, M. S., Çınar, A. (2018) *Ağaç-tohum algoritmasının CUDA destekli grafik işlem birimi üzerinde paralel uygulaması.* Gazi Üniversitesi, Mühendislik-Mimarlık Fakültesi Dergisi, vol. 2018, no. 2018, p. 1397–1409.

[8]. Garland, M. (2008) *Parallel Computing Experiences with CUDA.* IEEE Micro, vol. 28, no. 4, p. 13–27

[9]. Reynolds, C. W., Craig, R. (1987) *Flocks, herds and schools: A distributed behavioral model* in Proceedings of the 14th annual conference on Computer graphics and interactive techniques*.*. SIGGRAPH '87, 1987, vol. 21, no. 4, p. 25–34.

[10]. Ab Wahab, M. N., Nefti-Meziani, S., Atyabi A. (2015) *A Comprehensive Review of Swarm Optimization Algorithms.* PLoS One, vol. 10, no. 5, p. e0122827.

[11]. Taylor, P., Chen, C., Ting, C., Chen, C. (2014) *An Improved Ant Colony System Algorithm For The Vehicle Routing Problem.* no. 2014, pp. 37–41.

[12]. Yu, B., Yang, Z., Xie, J. (2011) *An Improved Ant Colony System Algorithm For The Vehicle Routing ProblemA parallel improved ant colony optimization for multi-depot vehicle routing problem.* no. 1, p. 183–188.

[13]. Karaboğa, D. (2017). *Yapay Zeka Optimizasyon Algoritmaları*. Ankara: Nobel Kitap

[14]. Ata, B., Coban, R. (2014) *Linear quadratic optimal control of an inverted pendulum using the artificial bee colony algorithm.* in 2014 IEEE International Conference on Automation pp. 1–4

[15]. Sree Ranjini, S., Murugan, S. (2017) *Memory based Hybrid Dragonfly Algorithm for numerical optimization problems.* Expert Syst. Appl., vol. 83, p. 63–78.

[16]. Park, J., Jeong Y, Shin J, Lee K. (2010) *An improved particle swarm optimization for nonconvex economic dispatch problems.* J. Electr. Eng. Technol., vol. 8, no. 1, p. 80–89.

[17]. Coello Coello, C., Pulido, G., Lechuga, M. (2004) *Handling multiple objectives with particle swarm optimization.* IEEE Transactions on Evolutionary Computation, vol. 8, no. 3, pp. 256–279.

[18]. Yang, X. (2010) *Nature-Inspired Metaheuristic Algorithms.* Luniver Press, 115. doi: 10.1016/B978-0-12-416743-8.00005-1

[19]. Tsai, P., Pan, J., Liao, B., Tsai, M., Istanda, V. (2012) *Bat Algorithm Inspired Algorithm for Solving Numerical Optimization Problems.* Applied Mechanics and Materials vol. 148-149, p. 134-137

[20]. Yazdani, M., Jolai F. (2016) *Lion Optimization Algorithm (LOA): A nature-inspired metaheuristic algorithm.* Journal of Computational Design and Engineering, vol. 3, no. 1, p. 24–36

[21]. Karaboga, D. (2005) *An idea based on Honey Bee Swarm for Numerical Optimization.* Technical Report TR06, Erciyes University p. 10

[22]. Kaya, S., Fığlalı (2017) *Çok Amaçlı Optimizasyon Problemlerinde Pareto Optimal Kullanımı.* Sosyal Bilimler Araştırma Dergisi p. 9-18

[23]. Çunkaş, M. (2008) *Design optimization of electric motors by multiobjective fuzzy genetic algorithms.* Mathematical and Computational Applications vol. 13 no. 3 p. 153-163 doi: 10.3390/mca13030153

[24]. Damla, K. U. U. (2013). *Parçacık Sürü Optimizasyonu ile Mühendislik Optimizasyon Problemleri ve Başlangıç Değer Problemlerinin Çözümü.* Lisans tezi, İstanbul Teknik Üniversitesi, İstanbul.

[25]. Çunkaş, M. (1933) *The Use of Indifference Curves in the Analysis of Foreign Trade.* The Quarterly Journal of Economics vol. 47, no. 3, p. 493, doi: 10.2307/1883982

[26]. Sağ, T. (2008) *Çok kriterli optimizasyon için genetik algoritma yaklaşımları.* Yüksek lisans tezi, Selçuk Üniversitesi, Konya.

[27]. Atlas, M. (2008) *Çok Amaçlı Programlama Çözüm Tekniklerinin Sınıflandırılması.* Anadolu Üniversitesi Sosyal Bilimler Dergisi vol. 8, no. 1, p. 47-68

[28]. Molga, M., Smutnicki, C. (2005) *Test functions for optimization needs.* p. 1-43

[29]. Akçay, M., Şen, B., Orak, İ. M., Çelik, A. (2011) *Parallel Computing And CUDA.* 6. Uluslar arası İleri Teknolojiler Sempozyumu

[30]. Wilkinson, B., Allen, C. M. (2005). *Parallel programming : techniques and applications using networked workstations and parallel computers*. Pearson: Prentice Hall

[31]. Pacheco, P. S. (2011). *An introduction to parallel programming*. Elsevier Inc., p. 1-370, doi: 10.1016/C2009-0-18471-4

[32]. Çetin,N. M., Hacıömeroğlu, M. (2013) *GPU Hızlandırmalı Veri Demetleme Algoritmalarının İncelenmesi.* Online Academic Journal of Information Technology, vol. 4, no. 12, doi: 10.5824/1309-1581.2013.2.002.x

[33]. Sinan, K. (2009). Matlab Nedir. 15 Ağustos 2019 tarihinde http://ekologo.blogspot.com/2009/01/matlab-nedir.html adresinden erişildi.

[34]. MathWorks, Inc. (2019). Introducing Mex Files - Matlab Simulink. 17 Ağustos 2019 tarihinde https://www.mathworks.com/help/matlab/matlab_external/introducing-mex-files.html adresinden erişildi.

[35]. Uçkan, T., Dal, D. (2016) *Image Quality Improvement On Opengl-Based Animations By Using Cuda Architecture. Uludağ Üniversitesi Mühendislik Fakültesi Dergisi, vol. 21, no. 1,* doi: 10.17482/uujfe.97021

[36]. Peker, M., Özkaraca, O. (2018) *Büyük ölçekli veri setleri için GPU hızlandırmalı melez bir GA-SVM: Cu-GA-SVM.* Gazi Üniversitesi Fen Bilimleri Dergisi Part C, p. 581-591, doi: 10.29109/gujsc.388244

[37]. Menchaca-Mendez, A., Coello, C. A. C. (2013) *A new selection mechanism based on hypervolume and its locality property.* IEEE Congress on Evolutionary Computation, p. 924-931, doi: 10.1109/CEC.2013.6557666

[38]. Campo, A., Nouyan, S., Birattari, M., Groß, R., Dorigo, M. (2006) *Negotiation of Goal Direction for Cooperative Transport.* Lecture Notes in Computer Science, p. 191-202, doi: 10.1007/11839088_17

[39]. MathWorks, Inc. (2019). Parallel Computing ToolboxTM User's Guide R2019a. 17 Ağustos 2019 tarihinde https://www.mathworks.com/help/pdf_doc/parallel-computing/distcomp.pdf adresinden erişildi.

[40]. Aci, Ç. İ., Polat, R., Akdağlı, A. (2017) *A Modification of Dragonfly Algorithm by means of Exploration Behaviour for Single Object Problems. International Conference on Theoretical and Applied Computer Science and Engineering*

ÖZGEÇMİŞ

Adı ve Soyadı **:** Ramazan POLAT

Doğum Tarihi **:** 14/09/1986

E-mail **:** ramazanpolat@mersin.edu.tr

Web adresim :www.ramazanpolat.info

Öğrenim Durumu **:**

Derece	**Bölüm/Program**	**Üniversite**	**Yıl**
Yuksek Okul	Bilgisayar Proğ.	Mersin Üniversitesi	2005-2007
Lisans	Bilgisayar Mühendisliği	Çukurova Üniversitesi	2010-2013
Yüksek Lisans	Elektrik Elektronik Müh.	Mersin Üniversitesi	2015-2019

ESERLER (Makaleler ve Bildiriler)

1. R. Polat, A. Akdağlı , Ç. İ. Acı – "A Modification Of Dragonfly Algorithm By Means Of Exploration Behaviour For Single Object Problems" , International Conference On Theoretical And Applied Computer Science And Engineering 2017 (Icetasce) , November 10,2017 , Houston Hotel, Ankara[40]