YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

PERFORMANS ARTIRMAYA YÖNELİK PARALEL MİMARİLERİN YAPAY SİNİR AĞLARI YAKLAŞIMI İLE DEĞERLENDİRİLMESİ

Bilgisayar Yük. Müh. Sırma Yavuz

FBE Bilgisayar Mühendisliği Anabilim Dalında Hazırlanan

DOKTORA TEZİ

Tez Savunma Tarihi	**:** 24 Mart 2006
Tez Danışmanı	**:** Prof. Dr. Oya KALIPSIZ (YTÜ)
Jüri Üyeleri	**:** Prof. Dr. Coşkun Sönmez (YTÜ)
	: Prof. Dr. Fikret Gürgen (BÜ)
	: Prof. Dr. Mustafa Bayram (YTÜ)
	: Doç. Dr. Selim Akyokuş (DÜ)

İSTANBUL, 2006

İÇİNDEKİLER

Sayfa

SİMGE LİSTESİ

α	İleti hazırlama süresi
α_2	Paket hazırlama süresi
β	Başlangıç süresi
τ	Bant genişliği
p_s	İleti paket boyutu
T_{comm}	Haberleşme gecikme süresi
D	İşlemciler arası mesafe
H	Paket başlığı boyutu
l	İleti boyutu
T_{comm}	Haberleşme gecikme süresi
$a_{1:i}$	$\{F_{1:i}\}$ düğümünün çıktısı
a_Q	Yapay sinir ağının çıktısı
$b_{i:j}$	bias
$\mathcal{E}_T$	Hata fonksiyonu
$F_{i,j}$ veya $\sigma_{i:j}$	*i.* katmandaki *j.* düğümün fonksiyonu
i	*i.* katman
L	Son katman
Q	Veri kümesindeki eleman adedi
s_i	*i.* katmandaki düğüm sayısı
t_Q	Hedeflenen çıktı değeri
$\tau_{i:j}$	Eşik değeri
$w_{i:,j,k}$	*i*-1 seviyesindeki *k* düğümünü *i* seviyesindeki *j* düğümüne bağlayan ağırlık
$\underset{-}{w}$	Tek kolonlu ağırlıklar vektörü
x_i	Veri kümesinin *i.* elemanı
$\underset{-m}{x} = \{x_i^m\}$	*dxn* boyutundaki bir *S* matrisinin *m.* kolonu, yapay sinir ağının girdilerini temsil eder

KISALTMA LİSTESİ

AIMS	Automated Instrumentation and Monitoring System
ATM	Asynchronous Transfer Mode
BFGS	Broyden-Fletcher-Goldfarb-Shanno
$CHIP^3S$	Characterization Instrumentation for Performance Prediction of Parallel Systems
CISC	Complex Instruction Set Computer
CMOS	Complementary Metal Oxide Semiconductor
Cmr	Characterisation Measurement Routines
CRUV	Communication Resource Usage Vector
EPIC	Explicitly Parallel Instruction Computing
flops	Floating-Point Operations per Second
FFT	Fast Fourier Transform
GAA	Geniş Alan Ağları
GNU	Gnu's Not UNIX (Free Software Foundation)
HiPPI	High Performance Parallel Interface
HMCL	Hardware Modelling And Configuration Language
IEEE	Institute of Electrical and Electronics Engineers
FDDI	Fiber Distributed Data Interface
IORUV	Input-Output Resource Usage Vector
LU	L-Lower Triangular Matrix, U-Upper-Triangular Matrix
LVQ	Learning Vector Quantization
Mbps	Megabytes per Second
MCP	McCulloch and Pitts
MFLOPS	Million Floating Point Instructions Per Second
MIPS	Millions of Instructions per Second
MPI	Message Passing Interface
MPICH	Message Passing Interface Chameleon
MRPS	Millions of Results per Second
MWIPS	Million Whetstones İnstructions per Second
NAS	NASA Advanced Supercomputing Division
NPB	NAS Parallel Benchmarks
PACE	Performance Analysis and Characterisation Environment
PBNN	Probabilistic Neural Networks
PICL	Portable Instrumented Communication Library
PRUV	Processor Resource Usage Vector
PVM	Parallel Virtual Machine
RBN	Radial Based Networks
RISC	Reduced Instruction Set Computers
RNN	Recurrent Neural Network
RUV	Resource Usage Vector
SCI	Scalable Coherent Interface
SDDF	Self-Defining Data Format
SPARC	Scalable Processor Architecture
SPEC	The Standards Performance Evaluation Corporation

SPLASH	Stanford Parallel Applications for Shared Memory
SPARC	Scalable Processor Architecture
TCP	Transmission Control Protocol
UTP	Unshielded Twisted Pair
VLIW	Very Large Instruction Word
YAA	Yerel Alan Ağları
YSA	Yapay Sinir Ağı

ŞEKİL LİSTESİ Sayfa

ÇİZELGE LİSTESİ Sayfa

ÖNSÖZ

Benim için hiç hayal etmediğim bir maceraya dönüşen doktora çalışmam boyunca, doktora çalışma alanımın ötesinde de pek çok konuda öğrenme şansım oldu. Bu zorlu yolculuk boyunca, her konuda, öğrenmeme büyük katkıları olan pek çok kişi ile karşılaşmam da kaçınılmazdı. Buradan,

Tez çalışmam süresince beni yönlendiren, görüşlerini aktaran ve çok ihtiyacım olduğu zamanlarda beni motive eden danışmanım Sayın Prof. Dr. Oya KALIPSIZ'a;

Çok sevdiğim mesleğimden ve kariyerimden vazgeçmeyi düşünecek kadar umutsuz olduğum bir dönemde tekrar çalışmalarıma başlamamı sağlayan ve aslında meslek seçimimde de önemli bir rolü olan, değerli hocam Sayın Prof. Dr. Durul ÖREN'e;

Doktora çalışmamın başlangıcında danışmanım olan, sağlık problemleri yaşadığım ve hayatımın şimdiye kadarki en zor dönemi olarak nitelendirdiğim dönemde, bana en büyük desteği veren hocam Sayın Prof. M. Yahya KARSLIGİL'e;

Bilgi ve deneyimlerini benimle paylaşıp, yol gösteren, tez izleme komitesinin değerli üyeleri Sayın Prof. Dr. Fikret GÜRGEN ve Sayın Prof. Dr. Mustafa BAYRAM'a;

Benimle deneyimlerini paylaşan, desteklerini esirgemeyen Sayın Prof. Dr. Ertuğrul ERİŞ, Sayın Prof. Dr Galip CANSEVER ve Doç. Dr. Tülay YILDIRIM'a;

Bana her zaman destek olan ve sabır gösteren, sevgili arkadaşlarım Yrd. Doç.Dr. Banu DİRİ, Yrd. Doç Dr. Songül ALBAYRAK, Dr. Ömer Özgür BOZKURT, Göksel BİRİCİK, Ekin Su UĞURLU ve Yıldız Teknik Üniversitesi Bilgisayar Mühendisliği Bölümündeki tüm arkadaşlarıma;

Warwick Üniversitesi Bilgisayar Bilimleri Bölümündeki hocalarım ve arkadaşlarıma;

Artık aramızda olmayan, ama güvenini ve sevgisini daima yanımda hissettiğim sevgili babama;

Benden sevgisini ve desteğini hiç bir zaman esirgemeyen, sevinçlerimi ve üzüntülerimi benden daha yoğun yaşayan sevgili anneme, kardeşime ve tüm ailemize;

Bu uzun ve zorlu süreç boyunca, desteği ve sevgisi ile daima yanımda olan ve çok az kişinin yapabileceğine inandığım fedakârlıklar gösteren sevgili eşime;

Teşekkürlerimi sunarım.

ÖZET

Gelişen teknolojiler ve paralel sistemlerin yaygınlaşması sayesinde performans analizi giderek artan bir gereklilik haline gelirken geleneksel yöntemler arasında oluşan boşluğu dolduracak, evrensel ve kullanımı daha kolay olan modellere ihtiyaç da artmaktadır. Bu tezin amacı özellikle haberleşme ağları üzerinde paralel çalışan kümelenmiş bilgisayarlar için yapay sinir ağlarını kullanarak bir performans tahmin ve analiz yöntemi geliştirmektir.

Performans tahmini alanında istatistiksel yöntemler şimdiye kadar yaygın olarak kullanılmış olmasına rağmen, yapay sinir ağlarının bu amaçla kullanımı ilk kez bu çalışmada önerilmiştir. Elde edilen sonuçlar yapay sinir ağlarının ve özellikle geri dönüşümlü ağların bu alanda başarı ile kullanılabileceği yönündedir.

Yapay sinir ağı modelleri, gerçek kullanıcı kodlarının kullanılmasına ve sunulan modellerin girdilerini oluşturan donanım ve yazılım parametreleri arasındaki etkileşimleri izlemeye olanak verdiği için, karşılaştırmalı değerlendirmelerden daha sağlıklı ve detaylı sonuçlar vermektedir. Modeller, gerçeklenmelerinin kolaylığı ve modellerin oluşturulması sırasında birtakım varsayımlara ihtiyaç bırakmaması açısından, simülasyon ve analitik yöntemlere de alternatif oluşturmaktadır.

Oluşturulan yapay sinir ağı (YSA) modelleri, farklı platformlar üzerinde çalıştırılan paralel programların aritmetik işlem ve haberleşme performanslarını tahmin etmek için kullanılmıştır. Kullanılan modellerden ilki tek gizli katmanlı, ileri beslemeli, geri yayılımlı YSA modeli olup, sinir ağının eğitim yöntemi olarak Levenberg-Marquardt tercih edilmiştir. Tasarlanan ikinci model ise, beş adet içerik elemanına sahip, kısmi geri dönüşümlü Elman ağıdır ve BFGS eğitim algoritması ile birlikte kullanılmıştır. Testlerde kullanılmak üzere iki ayrı uygulama seçilmiştir. Bunlardan ilki işlemciler arasında yoğun veri alışverişi gerektiren, 2-boyutlu bir Hızlı Fourier Dönüşümü (FFT) uygulamasıdır. Seçilen ikinci uygulama ise tipik bir kayan noktalı aritmetik uygulaması olarak sınıflanabilecek, Monte Carlo yöntemini kullanan bir uygulamadır. YSA modellerinin eğitilmesi, testi ve doğrulanması için kullanılan veriler iki şekilde elde edilmiştir. Verilerin önemli bir kısmı seçilen paralel uygulamaların Sun Sparc iş istasyonu üzerinde çalıştırılması ile, diğer kısmı ise farklı donanım ve komünikasyon sistemlerinin, PACE (Performance Analysis and Characterisation Environment) yardımı ile oluşturulan modelleri kullanılarak elde edilmiştir.

Anahtar Kelimeler: Yapay sinir ağları; Performans analizi; Performans değerlendirme; Paralel bilgisayarlar; Paralel hesaplama

ABSTRACT

The requirements of new applications always motivate the development of new systems. The enormous performance range offered by today's systems adds a level of difficulty to performance evaluation methods, which must consider the relative values and contributions of various components. It is complicated to predict and validate the contributions of these interrelated factors only analytically. Benchmarking is a popular way but equally open to misuse. There are few accurate performance analysis and visualization tools, however they are usually either too complex or system dependent.

PACE is one of the easy to use and reliable performance analysis toolsets. It allows users to model their own system and application easily and gives an accurate prediction of execution time. Modeling user specific systems or adapting user's own codes into PACE still involves a level of effort.

This thesis investigates the possibility of predicting performance of real applications by using artificial neural network models. Neural networks can learn to approximate any function and behave like associative memories by using just example data that is representative of the desired task. The models presented here are aimed to be simple and usable in general cases.

The contribution of this thesis is to present two neural network models for performance prediction. These models can be incorporated into a characterization tool, such as PACE, or can be used separately. They are potentially robust and the prediction results are proved to be accurate. Although, the artificial neural networks have been used for various prediction tasks, their use in performance evaluation area is a novel approach.

Keywords: Neural networks; Performance analysis; Performance evaluation; Parallel computers; Parallel computing

1. GİRİŞ

Kümelenmiş paralel bilgisayarlar, sahip oldukları performans potansiyeline rağmen, farklı sebeplerle yaygın olarak kullanılmamışlardır. Bu sistemlerin performans açısından gösterdikleri değişkenlik, benimsenmemelerinin önde gelen sebeplerinden biridir. Sistemin teorik performansına bazı uygulamalarla yaklaşmak bile mümkün değilken, sistemin mimari özelliklerini iyi kullanabilen diğer uygulamalarla, bu değere yaklaşılabilir. Elde edilen performans, kullanılan mimarinin ve uygulama programının parametrelerine bağlı olarak, önemli değişiklikler gösterebilir. Bahsedilen parametrelere, sistemlerin paralel çalıştırılması da eklendiğinde performans analizi ve kıyaslaması daha da gerekli ve önemli hale gelmiştir.

Son on yılda, bilgisayar sistemlerinin kapasitelerinin yanı sıra, haberleşme ağı teknolojileri de hızla gelişmiş ve özellikle kümelenmiş paralel sistemlerin potansiyelleri artmıştır. Bu sistemlerin sunduğu geniş performans aralığı, bağıl değerler ve çeşitli bileşenlerin katkılarını da dikkate alma zorunluluğu, performans değerlendirme uzmanlarının işini daha da zorlaştırmıştır. Bu tür bağıl değerlerin ve birbiri ile ilişkili faktörlerin performans üzerindeki etkilerini, simülasyonlar ve analitik yöntemlerle tahmin etmek ve doğrulamak oldukça güç bir iştir. Simülasyonların ve analitik yöntemlerin en önemli dezavantajları, geliştirilmelerinin uzun zaman alması ve karmaşık sistemler için birtakım varsayımlar yapmanın gerekliliği, başka bir deyişle, evrensel olamamalarıdır. Bu sorunlar, üreticileri, mimari tasarımcılarını ve kullanıcıları karşılaştırmalı değerlendirme programlarını kullanmaya yöneltmiştir.

Analitik modellerin hem geliştirilmesi zordur hem de takip edilebilir olmak adına birtakım varsayımlara yer verildiğinden evrensel olmaktan uzaktırlar. Dally (1990) tarafından, kübik ağların performans analizi için, önerilen model gibi stokastik bağımsızlık varsayımına veya kuyruk teorisine dayanan modeller bir noktaya kadar başarılı olmuşlardır ancak çoğu zaman uygulanabilir değillerdir.

Simülasyonlar, çoğunlukla gerçeği daha iyi yansıtan, güçlü modellerdir ancak bu modellerin geliştirilmesi uzun bir zaman alır, çalıştırılmaları pahalıdır ve yine de gerçek ölçümlerin yerini tutmazlar. Simülasyon yöntemi ile geliştirilen modellerin her zaman analitik modeller veya gerçek ölçümler ile doğrulanması da gereklidir.

Simülasyonlar da genellikle modelleyebildikleri sistemler ve çalıştırılabilecekleri platformlar açısından kısıtlamalar getirirler. Örneğin, paralel sistemlerin performans analizi için geliştirilmiş olan PROTEUS (Brewer vd., 1991), güvenilir ve hızlı bir simülatör olmakla birlikte, kodun lokal bellekte olduğunu varsaydığı için cep bellek etkilerini

yakalayamamaktadır ve sanal bellek kullanımına destek vermemektedir.

ClusterSim (Góes vd., 2004), Java tabanlı, kümelenmiş bilgisayarların oluşturduğu yükün simülasyonu ve görsel olarak modellenmesi için geliştirilmiş bir araçtır ancak kullanılan iş yükleri sentetik olarak yaratılmıştır. Netsim (Lewis, 1993) Ethernet performans simülatörü, gibi haberleşme performansını ölçmeye yönelik diğer simülasyon modelleri de mevcuttur.

Performans analistleri için en önemli ölçü zamandır. Hem uygulama geliştirenler hem de mimari tasarımcıları, belirli mimarilerde kodların ne kadar hızlı çalışacağını bilmek isterler. Bu iki grubun ihtiyaçları aynı olmakla birlikte, amaçları oldukça farklıdır. Uygulama geliştirenler tabandaki mimarinin tüm olanaklarını kullanabilmek için kodlarını nasıl yazacakları konusunda geri besleme isterken, mimari tasarımcıları, sonraki nesil mimarilerde performansı artırabilmek için darboğazları ortadan kaldırmak isterler.

Karşılaştırmalı değerlendirmelerin sonuçlarının, içerdikleri programların çalışma sürelerinin temsil ettiği, tek bir sayı ile ifade edilmesi de bu yöntemlerin güvenilirliğini tartışmaya açmaktadır (John, 2004). Yeni mimariler tasarlanırken, popüler karşılaştırmalı değerlendirme programlarında başarılı sonuçlar alacak iyileştirmelere öncelik verilmesi de, son kullanıcıları yanlış yönlendirebilmekte, kendi uygulamaları ile benzer performansı yakalayamamalarına neden olabilmektedir.

Karşılaştırmalı değerlendirmelerin ilk örnekleri arasında, Whetstone (Curnow ve Wichmann, 1976) ve Dhrystone (Weicker, 1984) sentetik programları sayılabilir. Sentetik karşılaştırma programlarından sonra çekirdek program parçacıklarından oluşan LINPACK ve bütün algoritmaların yanı sıra basit program parçacıkları da içeren EuroBen (Steen ve Jack Dongarra, 2002) karma karşılaştırmalı değerlendirmeleri geliştirilmiştir.

Özellikle mimari tasarımcılar tarafından farklı nesil mimarileri karşılaştırmak için, SPEC (Reilly vd., 1996; Henning, 2000) gibi ticari veya ticari olmayan karşılaştırmalı değerlendirme programlarını yaygın olarak kullanılmaktadırlar. Ancak, bu tür karşılaştırmalı değerlendirme programlarının, bazı kısıtları vardır ve elde edilen sonuçların güvenilirliği tartışmaya açıktır (Dongarra vd., 1987). Söz konusu olumsuzluklara rağmen, gelecek nesil mimariler oluşturulurken, önce benzetilmiş modeller için karşılaştırmalı değerlendirme programlarının çalıştırılması yaygın bir durumdur ve elde edilen sonuçlar tasarımları büyük ölçüde etkiler.

Kullanıcılar açısından bakıldığında ise, karşılaştırmalı değerlendirme programları genellikle kullanıcı uygulamalarını temsil etmediği için, bu değerlendirmelere dayanarak satın aldıkları

mimarilerde kabul edilebilir bir performansa ulaşana kadar kodlarını en iyileştirebilmekle uğraşmak zorunda kalabilmektedirler.

Üreticiler, kodlarını analiz edebilmeleri için kullanıcılara bir takım üst-düzey araçlar sunarlar. Ancak, bu tür araçlar genellikle belirli bir mimariye özgüdür ve uygulama geliştiren kişilerin farklı mimariler için farklı araçlar kullanmayı öğrenmesini gerektirir. Bu eksikliği gidermek için, varolan mimariler arasındaki bu farklılıklardan etkilenmeyecek ve kod ile mimari arasındaki etkileşimi sezgisel denebilecek bir biçimde temsil edebilecek model ve araçlara ihtiyaç vardır.

Yapay sinir ağları, normal dağılım, doğrusallık ve değişkenlerin bağımsızlığı gibi katı varsayımlarda bulunan geleneksel yöntemlere mükemmel bir alternatif oluşturur. Yapay sinir ağları, başka yollarla açıklanması zor olabilecek çok çeşitli ilişkiyi yakalayabildiği için olayların modellenmesin hızlı ve nispeten kolay bir olanak sağlar (Jain vd., 1996).

Yapay sinir ağı modelleri, finansal tahmin ve analizler, üretimde iş akış kontrolü, ses ve görüntü tanıma gibi pek çok uygulamada başarı ile kullanılmıştır (Chunrong ve Niemann, 2000; Neji ve Beji, 2000). Ancak, performans tahmin ve analizi alanında uygulamaları bulunmamaktadır.

Bu çalışmada, paralel sistemlerin performansının değerlendirilmesi için ve bunu yaparken mimari ve program parametrelerinin performansa etkilerini de tespit etmek için, yapay sinir ağlarının nasıl kullanılabileceği incelenmiştir. İleri beslemeli ve geri dönüşümlü olmak üzere iki ayrı tip yapay sinir ağı modeli oluşturulmuştur. Bu modellerin eğitilmesi, doğrulanması ve test edilmesi için kullanılan veriler deneysel yollarla ve mevcut olmayan bazı teknolojiler için modeller oluşturularak toplanmıştır.

Elde edilen sonuçlar, yapay sinir ağlarının performans analizi alanında başarıyla kullanılabileceğini destekler yöndedir.

Sunulan tez çalışmasının kalan kısmı, şu şekilde düzenlenmiştir:

2. Bölümde, performans analizi ve değerlendirilmesi alanında kullanılan geleneksel yöntemlerden ve uygulama alanlarından bahsedilmiş, yaygın olarak kullanılan karşılaştırmalı değerlendirme programları incelenmiştir. Bu bölümde, diğer performans değerlendirme ve görsel izleme araçlarından da en yaygın kullanılanlar kısaca tanıtılmıştır.

3. Bölümde, bu çalışma kapsamında kullanılan, yüksek performanslı mimariler ve haberleşme teknolojileri tanıtılarak, temel özellikleri açısından karşılaştırılmıştır. Kümelenmiş bilgisayarların, paralel bir kaynak olarak kullanılmasına olanak veren ve seçilen

uygulamaların kodlarının paralelleştirilmesinde kullanılmış olan PVM (Parallel Virtual Machine) ve MPI (Message Passing Interface) ileti geçirme kütüphaneleri de bu bölümde kısaca tanıtılmıştır.

4. Bölümde, yapay sinir ağları ve öğeleri hakkında genel bilgi verilerek, en çok kullanılan sinir ağı topolojileri, eğitim strateji ve algoritmaları kısaca tartışılmıştır.

5. Bölümde, deneysel ölçümlerin yapıldığı ortam, ölçümlerde kullanılan uygulamalar ve mevcut olmayan sistemlere ait verilerin elde edilmesinde kullanılan PACE ortamı ve bileşenleri tanıtılmıştır. Seçilen uygulamaların, PACE ortamına nasıl uyarlandığı, PACE üzerinde donanım ve haberleşme modellerinin nasıl oluşturulduğu anlatılmıştır. Seçilen uygulamaların farklı iş istasyonları üzerinde çalıştırılması ile elde edilen ölçüm sonuçlarının bir kısmı da bu bölümde verilmiş ve PACE üzerinde modeller ile elde edilen verilerin güvenilirliğini göstermek amacı ile karşılaştırmalar yapılmıştır.

6. Bölümde, bu çalışmada önerilen yapay sinir ağı modelleri, kullanılan algoritmalar ve seçilen giriş-çıkış değişkenleri tanımlanmıştır. Toplanan verilerin analizi ve işlenmesinde kullanılan yöntemler ile oluşturulan modellerin test edilmesinde kullanılan veri kümelerinin nasıl oluşturulduğu da bu bölümde anlatılmıştır.

7. Bölüm, önerilen modeller ile Matlab ortamında, elde edilen sonuçları ve ilgili grafikleri içermektedir. Bu bölümde yapay sinir ağları ile elde edilen sonuçlar, ölçüm ve PACE modelleri ile elde edilen değerlerle kıyaslanarak hata hesapları yapılmıştır.

8. Bölüm olan, sonuç bölümünde ise, çalışmada elde edilen sonuçlar özetlenerek, önerilen yöntemin diğer performans analiz yöntemleri ile karşılaştırılması verilmiştir.

2. PERFORMANS DEĞERLENDİRME YÖNTEMLERİ

Performans değerlendirmesindeki amaç, sistemlerin performansını belirli açılardan artıracak fırsatları belirlemek ve daha etkin mimarilerin yaratılmasında rehberlik etmektir. Ancak, sistem performansını çeşitli faktörler bir arada etkilediği ve bu faktörlerin bazılarının değiştirilmesi diğerlerini de etkilediği için, çok işlemcili sistemlerin performansını analiz etmek oldukça karmaşık bir iştir

Çok işlemcili sistemlerin performansını analiz etmede yaygın olarak kullanılan üç yöntem vardır: analitik yöntemler, simülasyon ve deneysel yöntemler. Her üç yaklaşımın da kendi avantajları ve sınırlamaları vardır. Analitik modeller, örgütsel parametreler ile performans arasında analitik bağlantı kurmaya olanak vermeleri bakımından son derece güçlüdürler. Ancak bu modellerin evrensel olarak kabul görmemektedir. Kolay takip edilebilir olmak adına bu modellerde mimari ve uygulama özellikleri ile ilgili pek çok varsayım yapılmaktadır ve bu varsayımlar her zaman gerçeği tam olarak yansıtamayabilir. Örneğin çok işlemcili sistemlerde kuyruk teorisine dayalı hafıza arabirim modelleri, genellikle rasgele dağılımlı hafıza istem akışı olduğunu varsayar. Bu varsayım, son derece düzenli veri erişim modelleri sergileyen pek çok bilim ve mühendislik uygulaması için başarısız olur.

Simülasyonlar çoğunlukla gerçeği daha iyi yansıtır; ancak, bunların çalıştırılması pahalıdır ve gerçek ölçümlerin yerini tutmazlar. Dahası, gerçek sistemde bu tür modeller ile yakalanması güç olan girişimler bulunabilir.

Deneysel performans analizinin avantajı sistemin modeli yerine gerçek sistem üzerinde çalışılmasıdır. Bu tür çözümlerin dezavantajı ise, analiz edilen kodların kısıtlı olması ve genellikle karşılaştırmalı değerlendirmeler için kullanılan kodlara bakarak herhangi başka bir uygulamanın performansını tahmin etmeye yönelik bir yöntem sunamamalarıdır. Çok basit karşılaştırmalı değerlendirmeler kullanıldığında dahi, kodun özellikleri ile gözlenen performans arasında bağlantı kurmaya yarayacak genel yöntemler bulunmamaktadır.

Analitik ve simülasyon gibi modelleme tekniklerinin en fazla kabul gördükleri aşama, sistem davranışını tahmin etmeyi kolaylaştırdıkları, donanımın gerçeklenmesinden çok önceki sistem tasarım aşamasıdır. Bu, önemli kaynakların yetersiz bir tasarıma harcanmasını engelleyecek mantıklı kararların alınmasına yardımcı olur. Merkezi işlem biriminin analitik performans modelleri ile ilgili çalışmalar (Basu vd., 1990; Gloria vd., 1997) buna örnek gösterilebilir. Belirli bir mimari üzerinde uygulamaların çalışması ile ilgili analitik modeller de asimptotik ölçeklenebilirlik çalışmalarına yardımcı olabilir (Foster vd., 1991). Ancak, bu tür

modellerdeki donanımsal parametrelerin deneysel ölçümlerle ayarlanması gerekir.

Çok işlemcili sistemlerdeki mimari yaklaşımların çeşitliliği nedeni ile bu makinelerin verilen iş yükü için gerçek performansını ölçebilen modellerin geliştirilmesi son derece karmaşıktır.

Mimari ve uygulama parametreleri birbirine bağlı olduğundan ve bazı faktörlerin değiştirilmesi diğerlerini de etkileyebileceği için, tüm performans etkilerini içeren üstün ve takip edilebilir bir analitik model oluşturmak mümkün değildir. Çok işlemcili programların yürütüm süresi içindeki dinamik davranışını güvenilir bir şekilde analitik modeller ile yakalamak mümkün değildir.

Bu zorluklar, paralel bilgisayarların performansının tanımlanması ve değerlendirilmesi için karşılaştırmalı değerlendirme programlarının kullanılmasında öncülük etmiştir. Her ne kadar bu karşılaştırmaların kullanılması, yaygın olarak zor ve sonuçları tartışmaya açık kabul edilse de aynı zamanda, karmaşık programların çalıştırıldığı gelişmiş bilgisayarların performansı ile ilgili faydalı bilgiler sağlayabilecek, birkaç kabul edilmiş yöntemden biridir (Martin, 1987; Neves ve Simon, 1987). Bilgisayarların kıyaslanmasında kullanılan en yaygın yöntemler ve karşılaşılabilecek sorunlar (Dongarra vd., 1987) tarafından tanımlanmıştır.

2.1 Karşılaştırmalı Değerlendirme Programları

Performans değerlendirmesinde kullanılan en yaygın yöntem, karşılaştırmalı değerlendirme yapacak bir grup program oluşturmaktır, ancak bu karşılaştırmalı değerlendirmeler bazen bize test edilen donanımdan ziyade yazılım hakkında bilgi verir. Karşılaştırmalı değerlendirmeler, sonuçları bilgisayar sisteminin donanım özellikleri ile ilişkilendirilebilirse daha anlamlı ve faydalı olabilirler (Hockney, 1996).

Standart karşılaştırmalı değerlendirmeler, verilen programların belirli makineler üzerindeki yürütüm sürelerini verirler, ancak makine ve program karakteristikleri açısından neden bu sonuçların elde edildiğini açıklamakta yetersiz kalırlar. Dahası, mimariye bağımlı olan değerlendirmeler farklı programların farklı mimariler üzerindeki yürütüm sürelerini dahi veremediklerinden tümüyle yetersiz kalırlar (Saavedra ve Smith, 1996).

Çoklu bilgisayarların performansının ifade edilmesinde kullanılacak uygun ölçütlerin ne olacağı konusu da önemlidir. Bugün varolan mikroişlemcilerin çeşit ve farklılıkları göz önüne alındığında, MIPS (saniyede milyon komut) gibi tek bir ölçüt anlamsız kalmaktadır. MFLOPS (saniyede milyon kayan noktalı işlem), bilimsel uygulamalar için daha uygun bir ölçüt olmakla birlikte yine de yetersizdir. Son kullanıcı açısından bakıldığında ise, MRPS (saniyede milyon sonuç) tercih edilen bir ölçüt olabilir; ancak bu da evrensel bir ölçüt değildir.

Genellikle karşılaştırmalı değerlendirmelerin sonuçları, performansı bir "ortalama" ile ifade edecek şekilde özetlenir. Bu ortalamanın nasıl hesaplanacağı ise performans değerlendirme alanında en çok tartışılan konulardan biri olmuştur (Fleming ve Wallace, 1986). Siegel ve arkadaşları (1982) çalışmalarında, çoklu bilgisayarların performansının ölçümünde kullanılan diğer ölçütler ile detaylı bir tartışmaya yer vermektedir.

Gerçek uygulamalar haricinde, Hennessy ve Patterson (1996) karşılaştırmalı değerlendirme programlarını, çekirdek, sentetik ve oyuncak olmak üzere üç kategoriye toplamaktadır. Tek bir tip program kullanarak sistem performansının ölçülmesi zor olduğundan, bu kategorilerdeki programların bazılarını veya tümünü birlikte kullanarak oluşturulan karma (hibrid) karşılaştırmalı değerlendirmeler de vardır. Gerçek uygulamaların kullanıldığı karşılaştırmalı değerlendirmeler ise iyi tanımlanmış bilimsel problemleri çözen programlardır.

2.1.1 Çekirdek Karşılaştırmalı Değerlendirme Programları

Çekirdek değerlendirme programları tipik olarak gerçek uygulamalardan alınmış, yoğun aritmetik işlem içeren küçük kod parçalarıdır. Bu program parçacıklarının, karşılaştırmalı değerlendirme kümesinin bir parçası olarak kendi başlarına çalışabilmesi amaçlanmıştır.

LINPACK, yoğun doğrusal denklem sistemlerinin çözümünü yapan ve yüksek oranda kayan noktalı aritmetik işlem içeren, nümerik bir değerlendirme programıdır (Dongarra, 1993, 1995). Programın yürütüm süresinin %75'den fazlası, içsel vektör çarpımı yapan, tek-hassasiyetli sürümünde *saxpy* ve çift-hassasiyetli sürümünde *daxpy* olarak adlandırılan, bir alt yordamda harcanır. Bu karşılaştırmalı değerlendirme programının sonuçları MFLOPS (saniyede milyon kayan noktalı işlem) olarak rapor edilir.

Livermore Fortran çekirdek programları (Lawrence Livermore Çevrimleri), fizik biliminin çeşitli dallarından alınmış 24 nümerik hesaplama (iç döngü) yer almaktadır (McMahon, 1988). Her bir döngünün kod uzunluğu birkaç satır ile bir sayfa arsında değişmektedir. Bu döngüler çok miktarda kayan noktalı işlem ve yüksek oranda dizi erişimi gerektirmektedir. Program üç farklı vektör uzunluğu için her bir çekirdek programın MFLOPS oranını hesaplar.

NAS Paralel değerlendirme kümesinde (NPB versiyon1.0), 5 çekirdek program ve 3 akışkanlar dinamiği uygulaması olmak üzere 8 problem yer almaktadır (Bailey vd., 1991). Bunların hepsi bilgisayar sisteminin vektör performansını vurgular. Performans MFLOPS olarak ölçülür. NPB sürüm 2.0, bu 8 problemden 5 tanesini içerir. NPB sürüm 2.3 ise bu uygulamaların kodlarının MPI tabanlı gerçekleştirimlerini içerir.

Bunların dışında genel kullanıma girmemiş, konuya özel yordamlardan oluşturulmuş, AIM

[1] ve Business Benchmarks (Taheri, 1990) gibi değerlendirmeler de vardır.

2.1.2 Sentetik Karşılaştırmalı Değerlendirme Programları

Sentetik karşılaştırmalı değerlendirmelerde, tipik bir programın ortalama komut dağılımını belirlenmeye çalışır, sonra da aynı dağılımdaki komutlar karışımı işlemcide çalıştırılır. Bunlar gerçek programlar değildir ve bu kodlar herhangi birinin hesaplamak isteyeceği bir şeyi de hesaplamaz.

Whetstone, literatürde karşılaştırmalı değerlendirme yapmak için tasarlanmış ilk programdır (Curnow ve Wichmann, 1976). Bu sentetik program, her biri belirli tipte komutlar içeren (tam sayı aritmetiği, kayan noktalı aritmetik, koşul komutları vb.) dokuz küçük döngüden oluşmaktadır. Programda çoğunlukla evrensel değişkenler kullanılmıştır ve çok miktarda kayan noktalı işlem vardır. Yürütüm süresinin yarısı matematiksel kütüphane fonksiyonları için kullanılmaktadır. Karşılaştırmalı değerlendirme sonuçları MWIPS (saniyede mega Whetstone komutu) olarak bildirilir.

Dhrystone ise 1984 yılında Reinhold Weicker tarafından ADA ile geliştirilmiş bir başka sentetik karşılaştırmalı değerlendirme programıdır (Weicker, 1984). Dhrystone, hiç kayan noktalı işlem içermez ve yürütüm süresinin önemli bir kısmı dizgi (string) fonksiyonlarında harcanır. Whetstone ile farklı olarak, çok az evrensel değişken kullanılmıştır. Karşılaştırmalı değerlendirme sonuçları saniyede Dhrystone sayısı (bir saniyedeki döngü (iterasyon) sayısı) olarak bildirilir.

Sentetik karşılaştırmalı değerlendirmelere örnek olarak verilebilecek diğer uygulamalar arasında Hartstone, IOBENCH ve PAR-Bench sayılabilir.

Oyuncak karşılaştırmalı değerlendirmelere örnek olarak ise Hanoi kuleleri, Eratosthenes kalburu ve hızlı sıralama gibi programlar sayılabilir.

2.1.3 Karma Karşılaştırmalı Değerlendirme Programları

Karma değerlendirme örneklerinden birisi olan EuroBen, belirli uygulamalar için önem taşıyan basit programlardan bütün algoritmalara kadar bir seri programdan oluşur (Steen, 1991). Değerlendirmenin ilk aşamasında, basit programlar, işlem performansı ile ilgili temel bilgi verir. Bu seviye yeterli bulunmaz ise, bir sonraki seviyedeki rasgele sayı üreteci, hızlı Fourier dönüşümü gibi basit ama sık kullanılan algoritmaları içeren testler çalıştırılmalıdır. İstenen bilgi miktarına göre daha fazla modül çalıştırılır. Bu yöntem "kademeli yaklaşım" olarak adlandırılır. EuroBen değerlendirme kümesi dört modülden oluşur.

Genesis değerlendirme programı, Alman Suprenum dağıtık-bellek bilgisayarının

performansını diğer paralel bilgisayarlara göre değerlendirmek için geliştirilmiştir (Addision vd., 1993). Genesis, üç seviyeden oluşur. İlk seviye; haberleşme, eş zamanlama ve vektör işleme ile ilgili makinenin temel özelliklerini ölçmeye yönelik sentetik kod parçalarından oluşur. LU çarpanlarına ayırma, hızlı Fourier dönüşümü gibi çekirdek kodlar, hesaplama ağırlıklı uygulamaları temsil etmek üzere Genesis değerlendirme kümesinde yer almaktadır. Paralel sistemlerin tam bir değerlendirmesini yapabilmek, bellek veya giriş-çıkış birimlerinde oluşabilecek darboğazları ve paralel hale getirilemeyen kod parçalarının etkilerini gözlemlemek için ise fizik, meteoroloji vb. alanlardan seçilmiş uygulamalar kullanılmıştır. Genesis değerlendirmeleri, problem boyutu ve işlemci sayısına bağlı olarak performansın değişimini gözlemeye olanak verecek yaklaşık bir model sağlar.

2.1.4 Gerçek Uygulamalara Dayalı Karşılaştırmalı Değerlendirmeler

Perfect karşılaştırmalı değerlendirmeleri (Blume ve Eigenmann, 1992; Cybenko vd., 1990), Kuck ve Sameh'in önerisi ile performans değerlendirmede uygulamaya dayalı bir yöntem ortaya koymak amacı ile geliştirilmiştir. Perfect karşılaştırmalı değerlendirmeleri, çeşitli bilim ve mühendislik alanlarından seçilmiş ve 60.000 satırın üzerinde Fortran koduna sahip 13 programdan oluşur. Kullanılan yöntem, bir grup taban ölçümleri takiben, her bir koda ait en uygun hale getirilmiş ölçümlerin yapılmasını gerektirir.

SPLASH (Singh vd., 1991), Perfect karşılaştırmalı değerlendirmelerine benzer şekilde, çeşitli bilim ve mühendislik alanlarından seçilmiş 7 uygulamadan oluşur. Bu uygulamalar paylaşılmış bellekli çoklu işlemciler üzerinde çalışan mimari tasarımcılar ve yazılımcılara yardımcı olması amacı ile tasarlanmıştır.

Gerçek uygulamalara dayalı karşılaştırmalı değerlendirmeler arasında yer alan SLALOM (Gustafson, 1991), bir kutunun iç yüzeylerinin optik ışınsallığını (radiosity), bunların birbirlerini ne kadar ve nasıl etkilediğini hesaplayan problemi çözer. Sadece problem çözüm kısmı değil; girişler, problemin oluşumu ve çıkış da zamanlanır. SLALOM, sabit problem yerine sabit zamanın kıyaslanmasına dayanan ilk değerlendirmedir.

SPEC (Standart Performans Değerlendirme Kurumu), son dönemlerin en önemli performans değerlendirme girişimlerinden birisidir (Dixit ve Reilly, 1991; Henning, 2000). Amacı, karşılaştırmalı değerlendirmeler için kullanılabilecek büyük uygulama programlarını bir araya toplamak, düzenlemek ve dağıtmaktır.

SPEC değerlendirme kümesinde, kayan noktalı ve tamsayı işlem yoğunluğu olan programlar yer alır. Satıcı veya üreticiler, programları çalıştırmadan önce kaynak kodlarını alarak

kendileri derlerler. Üreticilerin, taban performans ölçümü ve en uygun hale getirilmiş performans ölçümü olmak üzere iki grup sonuç bildirmesi gerekir. Taban performans ölçümleri, aynı dilde yazılmış programlar için tek grup bayrak ve tek bir derleyici sınırlaması getirmektedir. Performansı ifade etmek için, uygun hale getirilmiş kodlara ait performans ölçüm sonuçlarının geometrik ortalaması alınır. Sonuçlar, referans olarak alınan bir makineye göre, bağıl olarak ifade edilir.

Ancak, karşılaştırmalı değerlendirmelerde elde edilen iyi sonuçlar, gerçek uygulamalar için aynı başarıyı ifade etmeyebilir. Dolayısı ile gerçek bir ortamdan seçilmiş uygulamalar her zaman için en iyi değerlendirmelerdir. Endüstriyel standart değerlendirmeler yapay bir ortamı temsil eder ve makine mimarilerinin belirli bir endüstriyel karşılaştırmaya uygun hale getirilebilirler. Karşılaştırmalı değerlendirmeler belirli bir mimarinin en iyi yönlerini vurgularken rakip mimarinin en zayıf özelliklerini ortaya koyacak şekilde düzenlenebilir.

2.2 Diğer Performans Değerlendirme ve Görsel İzleme Araçları

Son 20 yıl içinde, paralel ve dağınık sistemler için pek çok performans analiz ve görsel izleme aracı ortaya çıkmıştır. Bu tür performans ölçüm araçları, programın çalışmasını izleyerek performans kaybı olan kısımları belirlemek ve anlamak üzere bir performans verisi üretmeyi amaçlamaktadır. Bu bölümde, bu araçlardan yaygın olarak kullanılan bir kaç tanesi hakkında kısaca bilgi verilmektedir.

2.2.1 ParaGraph

ParaGraph (Heath ve Etheridge, 1991) ilk performans izleme araçlarından biridir. Hedeflenen mimaride homojen dağıtık bellekli paralellik varsayımına dayanır ve uygulamanın çalışma modeli direk olarak programlama modeline eşlenir. ParaGraph, işlemci kullanımını, işlemciler arası haberleşmeyi ve işlemler ile ilgili genel bilgileri görsel hale getirmek için, PICL (Worley, 1992) tarafından oluşturulan yürütüm izlerini kullanır. ParaGraph, ileti geçirme kodlarının iyileştirilmesini ve haberleşme ağı topolojilerini vurgulayan, iz-tabanlı dinamik ilk görsel izleme araçlarının örneklerinden biridir.

ParaGraph'ın işlemci kullanım ekranları Kiviat diyagramlarını, Gnatt grafiklerini ve şerit grafikleri içerir. Bunlara ek olarak, etkileşim matrislerini içeren bir grup haberleşme trafiği gösterimi, ileti kuyruk uzunluklarına ait histogramlar ve işlemciler arası haberleşme şablonları görüntülenebilir.

2.2.2 IPS-2 ve Paradyn

IPS-2 (Miller vd., 1990; Foster vd., 2001) ise, kullanıcıların hem paralel hem de dağıtık

uygulamaları tanımlamalarına ve performans bilgilerini toplamalarına olanak veren hiyerarşik bir yaklaşıma sahiptir. Bu hiyerarşi, kökü programı, dalları da makine ve işlemcileri temsil eden bir ağaç olarak gösterilir.

Paradyn (Miller vd., 1995), IPS-2'nin devamı olan bir projedir. Paradyn, performans darboğazlarını arama işini büyük oranda otomatik hale getirmiştir. Paradyn, öncelikle çok sayıda senkronizasyon engelleme, giriş-çıkış engelleme veya bellek gecikmeleri gibi üst düzeydeki problemleri inceler. Genel problem belirlendikten sonra, problemin nedenlerini tespit etmek için, daha fazla araç seçici olarak devreye sokulur. Araçların yerleştirilmesini otomatik olarak yönlendiren Performans Danışman modülü, darboğazları belirli sebepler ve programın belirli bölümleri ile ilişkilendirebilmek için, performans darboğazları ve program yapısı ile ilgili bir bilgi tabanına sahiptir. İlk yeni nesil akıllı performans araçlarından biri olan Paradyn, performans bilgisini seçilen uygulama çerçevesinde sunar.

2.2.3 Pablo ve SvPablo

Pablo (Reed vd., 1993), portatif ve genişletilebilir bir performans analiz aracıdır. Pablo, MPI programlarını izlemek, düzenlemek ve düzenlenmiş işletilebilir kodlar tarafından oluşturulmuş izlerin analizi için çeşitli bileşenler içerir. Çeşitli bileşenlerin birlikte çalışabilmesi, performans iz dosyalarında kullanılan veri formatına (SDDF: Self-Defining Data Format) dayanır. Pablo izleme bileşenleri çıktı olarak SDDF formatındaki izleme dosyalarını oluşturur ve analiz bileşenleri de SDDF formatındaki dosyaları girdi olarak kabul eder. Pablo dağıtım paketinde, diğer izleme kütüphaneleri ile oluşturulan izleme dosyalarının da Pablo ile incelenebilmesi için, PICL veya AIMS gibi dosya formatlarından SDDF formatına çeviren dönüştürücülere de yer verilmiştir.

Pablo başlangıçta dağıtık bellekli paralel sistemleri desteklemek üzere geliştirilmiş olsa da, altyapının bazı bölümleri veri-paralel programların analizini de destekleyecek şekilde genişletilmiştir. Illinois Üniversitesi ile Rice Paralel Hesap Araştırma Merkezi tarafından ortak olarak yürütülen bir projeden ortaya çıkan bu girişimlerin sonucunda SvPablo (Rose ve Reed, 1999) ortaya çıkmıştır.

SvPablo basit renk kodları kullanarak donanım ve yazılım performans verilerini uygulamanın kaynak kodu ile ilişkilendirmek için, derleme sürecinde programdaki değişimlerden elde edilen verileri kullanır. Paradyn'de olduğu gibi, etkileşimli bir sorgu ara yüzü kullanıcıların üst düzeydeki özet performans bilgilerinden daha detaylı, belirli ölçevlere ve işlemcilere ait nicel verilere ulaşmalarına olanak verir.

AIMS (Yan vd., 1995) ise Fortran 77 ve C dillerinde, NX, PVM veya MPI kütüphaneleri kullanılarak yazılmış programların ölçüm ve analizinde kullanılan bir araçtır. Diğer görsel performans analiz araçları arasında Nupshot (Browne vd., 1998) ve daha sonra Intel Trace Analyzer olarak adlandırılan VAMPIR [3] sayılabilir.

2.2.4 PACE Performans Analiz Aracı

Warwick Üniversitesi'nde geliştirilmiş olan PACE (Nudd vd., 1999) aracı, katmanlı bir yapıya sahip olup, sistemin yazılım ve donanım bileşenlerinin ayrı ayrı modellenmesine olanak verir. PACE ile elde edilen sonuçların hata oranı %10 un altında olarak tespit edilmiştir. Bu çalışmada da, PACE hâlihazırda elimizde bulunmayan yüksek performanslı sistemlerin modellenerek verilerin nicelik ve nitelik olarak artırılması amacıyla kullanılmıştır.

3. YÜKSEK PERFORMANSLI BİLGİ İŞLEME ORTAMLARI

1980'li yıllarda, daha hızlı ve etkin işlemciler yaratmanın, performansı artırmak için en iyi yol olduğu düşünülüyordu. İşlemcilerin ve haberleşme ağlarının hızla gelişmesi ile paralel bilgi işleme alanındaki gelişmeler yüksek performanslı bilgi işleme anlayışını tümüyle değiştirmiştir. Bundan sonra kümelenmiş iş istasyonları tercih edilen süper bilgi işleme ortamları haline gelmiştir.

Bu bölümde, kümelenmiş bilgi işlem ortamlarını oluşturmada kullanılan mimari ve ağ teknolojilerinden, sadece bu çalışma ile ilgili olanlar kısaca tanıtılacaktır. Sırası ile önce kümelenmiş bilgi işlem ortamlarını oluşturmada kullanılan mimarilerden, daha sonra da ağ teknolojileri ve ileti geçirme ara yüzlerinden bahsedilecektir.

3.1 İşlemci Mimarileri

1970'lerde üst düzey dil özelliklerinin işlemci komut kümelerine dahil edilmesi ile, donanım yazılımın yerini almaya başlamıştır. Bu gelişmenin sonucu; çok sayıda komut, adresleme tipi ve komut formatı içeren CISC (Karmaşık Komut Takımlı Bilgisayarlar) teknolojisidir. Bundan sonra CISC'e alternatif olarak, sadece en sık kullanılan komutlar için donanım desteği veren ve diğer komutları bu komutları ardı ardına kullanarak gerçekleyen, RISC (İndirgenmiş Komut Takımlı Bilgisayarlar) teknolojisi doğmuştur (Patterson ve Sequin, 1981). CISC ve RISC teknolojileri daha iyi performans sergileyebilmek için, birbirlerinden de fikir alarak yarışmaya devam etmektedirler. RISC tasarımcıları, fiziksel bağlantı yolu ile kontrol ederek her bir çevrim süresinde bir komut vermeyi başarmışlardır.

Çoklu komut verebilen, süperskalar işlemciler ise her bir çevrim süresinde birden fazla komutun getirilmesi, çözümlenmesi, yürütülmesi, çekilmesi ve sonuçlarının yazılması işlemlerini tamamlayabilmektedir (Sima, 1997).

Dataflow bilgisayarlar, programdaki tüm paralellik olanaklarından yararlanma kapasitesine sahiptir. Dataflow mimariler, makine dili olarak veri akış diyagramlarını kullandıkları için von Neumann mimarisine radikal bir alternatif oluştururlar. Veri akış diyagramları, komutların yürütülmesi ile ilgili sadece kısmen bir sıra belirtir ve bu şekilde bireysel komutlar seviyesinde paralel ve boru hattında yürütmeye olanak sağlarlar.

Süperskalar, VLIW (Very Large Instruction Word) ve dataflow işlemciler, komut seviyesinde paralelliği kullanabilme ve her çevrim süresinde birden fazla komut verme kapasitesine sahiptir. Ancak süperskalar işlemciler ile VLIW işlemciler, paralelliği kullanma açısından farklıdır. Geleneksel süperskalar tasarımında, yazılımcı programını yüksek seviyeli bir dil ile

ardışık komut dizisi halinde kodlar. Bu kodlar derleyici tarafından makine diline derlenir. Derleme işleminin sonucu da bir ardışık komut dizisidir. Bu komutlar işlemciye beslendikçe, işlemci bu komutlar üzerinde bağımlılık testi, yeniden sıralama gibi gerek transistör olarak gerek saat çevrimi olarak pahalı olan işlemler uygular ve komutları paralel olarak işler. Sonuçta da komutlar sıralarını kaybetmiş olarak işlemciden çıkmış olur.

VLIW tasarımında ise komutların paralel işletilmesini sağlayacak yeniden sıralama gibi işlemler bir yazılım tarafından yapılır. Son yıllarda VLIW tekniği, İntel'in IA-64 mimarili işlemcileri için önerdiği EPIC (Explicitly Parallel İnstruction Computing) tasarımı ile tekrar gündeme gelmiştir.

RISC ve CISC gibi EPIC de, bir dizi kural ve tasarım belirtimlerinden öte bir teknolojiler bütünü ve tasarım felsefesidir. Bu mimari, Intel ve HP'nin RISC ve CISC mimarilerinden edindiği tecrübeyi en son derleyici teknolojileri ile birleştirerek geliştirdiği yeni bir yaklaşımdır.

3.1.1 SPARC Komut Kümesi Mimarisi

RISC mimarisine dayanan belli başlı örnekler arasında HP/Compaq Alpha, Sun SPARC ve IBM/Motorola PowerPC sayılabilir. Bunlardan en eski olan SPARC (Scalable Processor ARChitecture), Sun firması tarafından, başlangıçta iş istasyonlarında kullandıkları Motorola 680x0 işlemcilerinin yerine almak üzere geliştirilmiştir. Bu mimari daha sonra SPARC Uluslararası konsorsiyumu tarafından desteklenen açık bir IEEE-standart mimarisi haline gelmiştir.

SPARC sürüm 9 (SPARC-V9), 1987'de SPARC mimarisinin çıkından itibaren en önemli değişikliklerin yapıldığı uygulamadır. SPARC-V9 ile gelen önemli gelişmeler arasında, 64-ikil adres ve veri üzerinde işlem yapabilme, artırılmış sistem performansı, en iyileştirici derleyiciler ve yüksek performanslı işletim sistemlerini destekleme özelliği, süperskalar gerçekleştirim ve hata toleransı sayılabilir.

Bu mimarinin ikinci nesil uygulaması olan UltraSPARC-II ise, yeni işlemci teknolojisini kullanmasının yanı sıra, daha yüksek saat frekansı, çoklu SRAM kipi ve UltraSPARC-I tabanlı sistemlerle uyum özelliklerini taşır [2]. UltraSPARC-II ile, SPARC-V9 komut kümesi mimarisinde tanımlanan ancak UltraSPARC-I mimarisinde uygulanmamış olan, önceden komut getirme özelliği de uygulanmıştır. Aynı ailenin üçüncü nesil uygulaması olan UltraSPARC-III ise, 0.18 μm CMOS teknolojisi ile 600 MHz'den başlayan hızlarda üretilmiştir (Normoyle vd., 1998). Bu tasarımın en önemli özelliklerinden biri büyük miktarda

cep bellek içermesidir.

3.1.2 IA-64 Komut Kümesi Mimarisi

Itanium işlemcisi, IA-64 komut kümesi mimarisinin ilk uygulamasıdır. İşlemci, geniş bir yelpazedeki ihtiyaçları karşılamak üzere en iyileştirilmiştir. Internet sunucuları ve iş istasyonları için yüksek performans sağlaması, 64-ikil adresleme desteği olması, güvenilirlik, donanım açısından IA-32 komut kümesi ile tamamen uyumlu olması, farklı platformlar ve işletim sistemleri için ölçeklenebilir olması bu mimarinin özellikleri arasındadır.

İşlemci, yazılım ve donanım arasında daha iyi bağlantı sağlamak için EPIC tasarım prensiplerini kullanmaktadır. EPIC yapısı, yazılımın geniş bir zamanlama aralığı için genel en iyileştirmeler yapmasına olanak veren güçlü mimari semantikler sağlar, dolayısı ile mümkün olan komut seviyesi paralellikten yararlanır.

Bu çalışmadaki testlerde kullanılan iş istasyonları ve modellenen makinelerden bazılarının temel özellikleri yukarıda gösterilmiştir (Çizelge 3.1).

Çizelge 3.1 SPARC ve IA-64 mimarili bazı ürünlerin temel özellikleri

Model	**SparcStation 5**	**Ultra 1**	**Ultra 10**	**Sun Blade 1000 Model 1750/1900**	**Hp Server rx4610**
Mimari	**UltraSPARC (MicroSPARC-II uyumlu) SPARC V-8**	**UltraSPARC-I SPARC V-9**	**UltraSPARC-II i SPARC V-9**	**UltraSPARC-III SPARC V-9**	**Itanium IA-64 EPIC Technology**
CPU İşlem Hızı	170 MHz	143 MHz	300/333 MHz	750/900-MHz	733/800 MHz
Ana Bellek Hızı	90 MHz	71.5 MHz	100/111 MHz	100/111 MHz	
Ana Bellek Kapasitesi	64 Mb	64 Mb	128 Mb	1 Gb	8 Gb
L1 Cep Bellek Kapasitesi	16Kb Veri 16Kb Komut	16Kb Veri 16Kb Komut	16Kb Veri 16Kb Komut	34Kb Veri 64Kb Komut	16Kb Veri 16Kb Komut
L2 Cep Bellek Kapasitesi	512Kb	512Kb	2Mb	8Mb	96Kb
L3 Cep Bellek Kapasitesi	-	-	-	-	4Mb
Adresleme/Veri Yolu	32 bit	64 bit	64 bit	64 bit	64 bit
Fonksiyonel Birimler	En iyilenmiş tümleşik Floating Point işlem birimi 1 Integer işlem birimi	3 Floating Point işlem birimi 2 ALU 4 Integer işlem birimi 2 Grafik işlem birimi	3 Floating Point işlem birimi 2 ALU 4 Integer işlem birimi 2 Grafik işlem birimi	3 Floating Point işlem birimi 2 ALU 4 Integer işlem birimi 2 Grafik işlem birimi	2 adet çifte duyarlı ve 2 adet tek duyarlı FMAC 4 adet tek çevrimli integer ALU. 4 MMX 3 dallanma birimi
Pipeline Kademeleri	5 Kademeli; her çevrimde 4 komut verilebilir	9 Kademeli; her çevrimde 4 komut verilebilir	9 Kademeli; her çevrimde 4 komut verilebilir	14 Kademeli; her çevrimde 4 komut verilebilir	10 Kademeli; her çevrimde 6 komut verilebilir

3.2 Kümelenmiş Sistemler ve Yüksek Performanslı Haberleşme Teknolojileri

Bir yerel ağ üzerinde kümelenmiş iş istasyonları veya PC'lerin kullanımı, özellikle 1994 yılında ilk Beowulf tasarımının sunulmasından sonra aniden artmıştır. Bu sistemlerin en cazip yönleri yazılım ve donanım olarak ekonomik olmaları ve kullanıcıların sistem üzerinde kontrol sahibi olabilmeleridir.

Çoğu durumda, ara bağlantılar standart Ethernet ile yapılmaktadır. Ethernet teknolojisi, bant genişliği ve gecikme süresi göz önüne alındığında, performans açısından oldukça eski olmakla birlikte ucuzdur. Ethernet için, 100 Mbps bant genişliğine karşılık, 100 µs civarında yüksek bir gecikme süresi söz konusudur. Gigabit Ethernet ise 10 kat daha fazla bant genişliği sağlamasına rağmen gecikme süresi aşağı yukarı aynıdır (Rash vd., 2000).

Alternatif olarak, kullanıcı alanından yürütülen, Myrinet [4], Giganet cLAN [5] ve SCI (Scalable Coherent Interface) [6] gibi ağ teknolojileri de mevcuttur. Myrinet ve Giganet cLAN için bant genişlikleri 1Gbps dolayındadır. SCI için ise bant genişliği teorik olarak 4-5 Gbps arasında ve gecikme süresi 10 µs'nin altındadır (James vd., 1990).

Myrinet, Giganet cLAN ve özellikle SCI tarafından sağlanan haberleşme hızları bazı tümleşik paralel sistemler ile aynıdır. İşlemci ve yazılım hızları dışında, süper bilgisayar sınıfındaki paralel makineler ile bu ağ teknolojilerini kullanan kümelenmiş bilgisayarlar arasındaki fark oldukça azdır ve günden güne azalmaya devam etmektedir.

ATM (Asynchronous Transfer Mode) ise başlangıçta telekomünikasyon endüstrisi için geliştirilmiş bir anahtarlamalı sanal-ağ teknolojisidir (ATM Forum, 1995). ATM hem yerel alan ağları (YAA) hem de geniş alan ağları (GAA) ile kullanılmak için tasarlanmış bir ağ teknolojisidir.

Bu çalışmadaki testlerde kullanılan ağ teknolojilerine ait ürünlerden bazılarının temel özellikleri Çizelge 3.2'de gösterilmiştir.

Çizelge 3.2 Farklı ağ teknolojilerine ait ürünlerden bazılarının temel özellikleri

Ağ Tipi	Ürün	Donanım ve Yazılım Gereksinimleri	Ağ Arayüzü	Teorik Bant genişlikleri veya transfer oranları
Ethernet	SunSwift PCI Adaptörü	Solaris 2.5.1 Donanım: 4/97 sürümü	10BASE-T ve 100BASE-TX ara yüzleri (otomatik pazarlık kapasite belirleme ve tam çift yönlü)	Otomatik algılama protokolü, adaptör hızını seçer (10 Mbps veya 100 Mbps)
Fast Ethernet	Sun FastEthernet PCI Adaptörü	Solaris 2.5.1 HW 4/97 veya sonraki sürümler	10BASE-T ve 100BASE-TX arayüzleri (otomatik pazarlık kapasiteli ve tam çift yönlü)	10 Mbps veya 100 Mbps
FDDI	SunFDDI/P 1.0 Adaptörü SunFDDI/P PCI kartları	Sun-4u PCI-tabanlı mimariler Solaris 2.5.1 HW 4/97	Sun FDDI	100 Mbps'e kadar veri transfer oranlarını destekler
ATM	SunATM 155 3U Compact PCI Adaptörü	Sun adaptörünü ATM anahtara bağlamak için, SC bağlantılı çok-düğümlü fiber kablo kullanılmalı	LAN erişimi için fiber optik alıcı vericili 155 Mbps ATM arayüz kartı	Çok kipli fiber optik veya kategori 5 UTP kablo üzerinden 155 Mbps
ATM	SunATM 622 3U Compact PCI Adaptörü	Sun adaptörünü ATM anahtara bağlamak için, SC bağlantılı çok-düğümlü fiber kablo kullanılmalı.	622 Mbps ATM arayüz kartı	62.5/125 m çok kipli fiber kablo üzerinden 622 Mbps
Gigabit Ethernet	Sun GigaSwift Ethernet UTP Adapter	Sun Ultra 5, 10, 60, 80 Sun Enterprise 220R, 250, 420R, 450, 3000/3500, 4000/4500, 5000/5500, 6000/6500, 10000 Sun Blade™ 1000	Tek portlu gigabit Ethernet bakır-temelli PCI Bus kartı	1 GBit/s (çift yönlü -1000 Mbps)
HiPPI	GENROCO SHP-6464 PCI GSN controller	SPARC PCI üzerinde Solaris 2.x desteği Compaq Alpha üzerinde Digital UNIX desteği	HiPPI-6400 fiziksel katmanı için ANSI standartlarına uyumlu	HIPPI-6400 için 6.4 GBits/s
Myrinet	Myricom M2M-PCI32	Herhangi bir topoloji için, anahtarları ve uç düğümleri birleştiren noktadan noktaya bağlantı	Myrinet ağ teknolojisi	çift yönlü 1.2Gbit/s

3.3 İleti Geçirme Arayüzleri

Paralel programlamanın kolaylığının gittikçe artması ve bu tür programların farklı ortamlara taşınabilir olması bilim adamları ve mühendisler için önemli bir faktördür. FORTRAN veya C programlama dilleri için, ileti geçirme arayüzü sağlayan kütüphane fonksiyonları kullanılarak, bir ağ üzerindeki iş istasyonları paralel bir kaynak olarak kullanılabilir.

Paralel programlama kütüphaneleri arasında tartışmasız en başarılı olanlar MPI (Pacheco, 1997) ve PVM (Geist vd., 1994) standartlarıdır. PVM 1989 yılında başlayarak Oak Ridge Ulusal laboratuarı ve Tennessee üniversitesi tarafından geliştirilmiştir, MPI ise, birkaç yıl

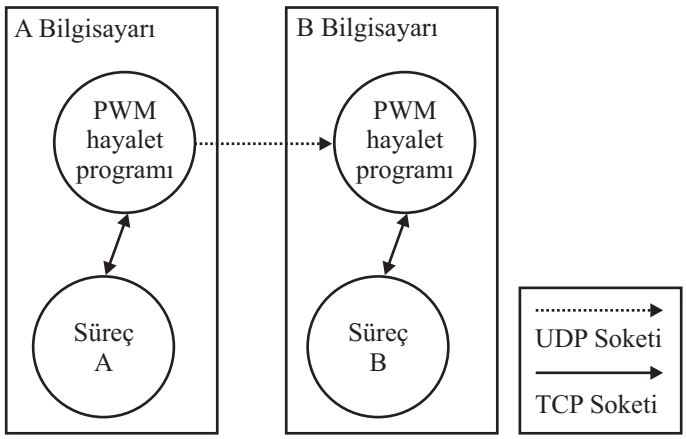
sonra, çeşitli üniversiteler ve ulusal laboratuarların katıldığı ve ileti geçirme kütüphaneleri için bir standart oluşturmayı hedefleyen bir konsorsiyum sonucunda ortaya çıkmıştır.

3.3.1 PVM (Paralel Sanal Makine)

PVM, farklı bilgisayarların sayısından ve konumundan bağımsız olarak, kullanıcılara uygulamalarını çalıştıracakları paralel bir ortam sağlar. PVM bir ağ üzerindeki aynı özellikte olmayan bilgisayarların kaynaklarını bir araya getirip bunlardan yararlanma ve yüksek performans ve fonksiyonellik sağlama kapasitesine sahiptir.

PVM yapısındaki ana fikir, isminden de anlaşılabileceği gibi bir grup farklı özellikteki bilgisayara tek bir paralel sanal makine gibi bakılmasıdır. PVM, birbiri ile uyumsuz bir bilgisayar topluluğu arasındaki tüm ileti yönlendirme, veri dönüştürme ve görev programlama işinin üstesinden gelir. Bu çalışmada yer alan testler için, PVM sürüm 3'e göre daha fazla makine desteği olan ve daha iyi haberleşme performansı sağlayan sürüm 3.3 kullanılmıştır.

PVM haberleşme mimarisi Şekil 3.1 de gösterilmiştir. PVM mimarisi, paralel sanal makinenin bir parçası olan her bilgisayarda yerleşik olan PVM hayaletleri (pvmd) üzerine kuruludur. Bir bilgisayardaki süreç, başka bir bilgisayardaki süreç ile haberleşmek istediğinde, bunu kendi üzerinde çalışan PVM hayalet programı vasıtası ile yapar.

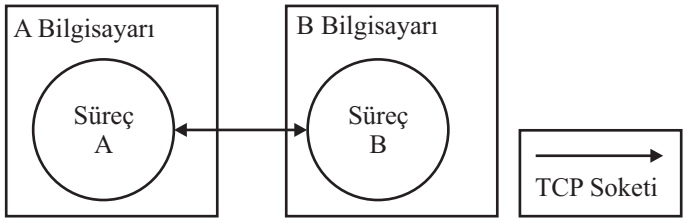


Şekil 3.1 PVM haberleşme mimarisi

3.3.2 MPI (İleti Geçirme Arayüzü)

MPI, programcılara ileti geçirme arayüzü ile birlikte, herhangi bir uygulamada özelliklerinin nasıl davranması gerektiğini belirten semantik tanımlama ve protokolü de sağlar. MPI, paralel görevler arasında hem noktadan noktaya hem de kolektif haberleşmeyi sağlayacak zengin bir kütüphaneye sahiptir. Ancak, MPI standartında, görevlerin işlemcilere dağıtımı ile ilgili bir yöntem belirtilmemiş ve bu iş üreticiye özel gerçekleştirimlere bırakılmıştır.

MPI haberleşme mimarisi PVM mimarisinden daha basittir ve başka bir işlemcideki süreçle haberleşmek isteyen herhangi bir süreç bunu UNIX TCP soketi vasıtası ile direk yapabilir (Şekil 3.2). Aynı bilgisayar üzerinde yer alan süreçler ise, iletiyi diğer sürecin ileti yastık belleğine koyarak haberleşebilir.



Şekil 3.2 MPI haberleşme mimarisi

MPICH, MPI şartnamesinin, etkin ve portatif olmak üzere tasarlanmış, tam bir gerçekleştirimidir. MPICH pek çok üreticinin iş istasyonunda çalışmaktadır ve WinMPI adı altında Windows işletim sistemine de uyarlanmıştır.

Bu çalışmada, Argonne ulusal laboratuarı ve Mississippi üniversitesi'nde geliştirilmiş olan, MPICH 1.0.13 sürümü kullanılmıştır.

4. YAPAY SİNİR AĞLARI

Yapay Sinir Ağları (YSA) kavramı, biyolojik sinir sistemi ile ilgili temellere dayanmaktadır. Yapay sinir ağları, insan beynindeki nöronların çalışmasını taklit eder. İlk nöron, nöropsikolog Warren McCulloch ve mantıkçı Walter Pits tarafından, 1943'de üretilmiştir (Kröse ve Smagt, 1996). Tek katmanlı bir algılayıcı olan, McCulloch ve Pitts (MCP) modeli yapay sinir ağlarının temelini oluşturur. Minsky ve Papert tarafından yazılan ve tek katmanlı algılayıcıların kısıtlılıklarını anlatan kitap (Minsky ve Papert, 1969), pek çok araştırmacının bu konuya ilgilerini kaybetmelerine neden olmuştur. 1980li yılların başında hata geriye yayma yönteminin keşfine kadar, çok az araştırmacı çalışmalarına devam etmiştir. Hopfield (1982) tarafından yapılan çalışmalardan sonra, yapay sinir ağları uygulamaları tekrar ortaya çıkmaya başlamış ve o zamandan bu yana endüstriyel, ticari ve bilimsel uygulamalarda başarı ile kullanılmıştır (Rumelhart vd., 1994; Atiya, 2001; Atiya vd., 1999).

Bir yapay sinir ağı, birbirine hiyerarşik olarak bağlı ve birbiri ile ağırlıklı bağlantılar vasıtası ile haberleşen bir grup işleme biriminden (nöron) oluşur. Bu bağlantıların ağırlıkları, ön bilgiler kullanılarak ayarlanabilir veya belirli bir öğrenme kuralına göre değişebilecek şekilde eğitilerek belirlenir. Bu durum, Rumelhart (1994b) tarafından yapılan tanımlamada şu şekilde ifade edilmiştir: "*yapay sinir ağlarında öğrenme problemi ağın istenilen işlemi yapmasına olanak verecek bağlantı güçlerini bulmaktan ibarettir*".

Yapay sinir ağları, gerçek değerli, ayrık değerli ve vektör değerli fonksiyonların öğrenilmesinde genel ve pratik bir yöntem sağlamaktadır. Bu çalışmada, performans değerlendirme alanında önemli bilgiler sağlayabilecek, işlem ve haberleşme sürelerine ait verilerin tahminde kullanılacak yapay sinir ağı modelleri önerilmektedir.

4.1 Temel Yapay Sinir Ağı Kavramları

Yapay sinir ağları, bilgiyi insan beynine benzer bir şekilde işler ve örnek yolu ile öğrenir. Bir yapay sinir ağı, çok sayıda ve birbiri ile bağlı, belirli bir problemi çözmek için paralel çalışan işlem elemanlarından (nöron) oluşur. Yapay sinir ağı modelleri, kullanılan topoloji, öğrenme stratejisi (öğretmenli, öğretmensiz vb.) ve öğrenme algoritması ile tanımlanır.

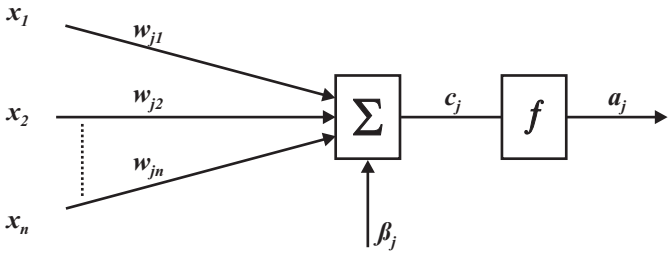
Bu bölümde, yapay sinir ağları ile ilgili kavramlar, mimariler ve bu çalışmada kullanılan algoritmalar ile ilgili temel bilgiler yer almaktadır.

4.1.1 Yapay Sinir Hücresi (Nöron)

Yapay sinir hücresi, komşularından veya dış kaynaklardan çeşitli girdiler alan ve bunları diğer birimlere de yayılan belirli bir çıktıyı hesaplamak için kullanan işleme birimidir. Şekil 4.1'de,

$x_1, x_2, \ldots, x_n$ şeklinde *n* adet girdisi olan bir yapay sinir hücresi gösterilmiştir. Girişlerin her biri w_{j1}, $w_{j2}, \ldots, w_{jn}$ ağırlıklarındadır. Toplama fonksiyonu $\sum$, hücreye gelen net girdiyi hesaplar. Ağırlıklı toplama fonksiyonu kullanıldığında, *a* nöronunun çıktısı şu şekilde ifade edilebilir:

$$f(\sum_{i=1}^{n} w_{ji} x_i + \beta_j) \tag{4.1}$$



Şekil 4.1 n adet girdisi olan basit bir nöron

Bu durumda aşağıdaki eşitlik yazılabilir:

$$c_j = \sum_{i=1}^{n} w_{ji} x_i + \beta_j \quad , \quad a_j = f(c_j) \tag{4.2}$$

Pozitif w_{j1} değerinin katkısı uyarım, negatif w_{j1} değerinin katkısı ise çekince olarak algılanır. Bu yayılım kuralına uyan birimler *sigma* birimleri olarak adlandırılır.

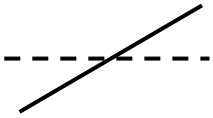
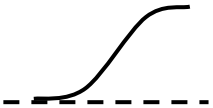
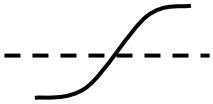
Bazı durumlarda, girdilerin bir araya getirilmesinde (toplanmasında) daha karmaşık kurallar uygulanır. Örnek olarak *sigma-pi* yayılım kuralı aşağıdaki gibi ifade edilir:

$$c_j = \sum_{i=1}^{n} w_{ji} \prod_{k=1}^{m} x_{ik} + \beta_j \tag{4.3}$$

4.1.2 Aktivasyon Fonksiyonu

Her bir yapay sinir hücresi, girdilerinin adedi dışında *f* aktivasyon fonksiyonu ile tanımlanır. Bu fonksiyon, hücreye gelen girdiyi işleyerek, hücrenin bu girdiye karşılık üreteceği çıktıyı belirler. Toplama fonksiyonunda olduğu gibi, aktivasyon fonksiyonu olarak da farklı fonksiyonlar kullanılmaktadır. Ağdaki işlem elemanlarının tümünün aynı aktivasyon fonksiyonunu kullanması da şart değildir. Çizelge 4.1'de en yaygın olarak kullanılan aktivasyon fonksiyonlarının grafikleri, formülleri ve alabilecekleri değerler verilmiştir.

Çizelge 4.1 Yaygın olarak kullanılan aktivasyon fonksiyonları

Eşik Değer Fonksiyonu (Hardlimit)	**Doğrusal Fonksiyon (Purelinear)**	**Sigmoid Fonksiyonu**	**Hiperbolik Tanjant Fonksiyonu (Tansigmoid)**
			
$f(x) = \begin{cases} 1, x \geq 0 \\ 0, x \leq 0 \end{cases}$	$f(x) = x$	$f(x) = \frac{1}{1+e^{-x}}$	$f(x) = \frac{2}{1-e^{-2x}} - 1$
$f(x) \in \{0,1\}$	$f(x) \in (-\infty,+\infty)$	$f(x) \in (0,1)$	$f(x) \in (-1,1)$

Bu fonksiyonlardan hangisinin seçileceği yapılarına ve uygulamaya göre belirlenir. Örneğin, eşik değer fonksiyonu $(-\infty,+\infty)$ aralığında yer alan herhangi bir değeri $\{0,1\}$ şeklinde iki değere dönüştürür, dolayısı ile sınıflama işlemi yapan çok katmanlı bir yapay sinir ağı için tercih edilebilir. Sigmoid ve tanjant sigmoid fonksiyonları, sıkıştırma fonksiyonları olarak bilinir ve $(-\infty,+\infty)$ aralığında yer alan herhangi bir değeri, sırası ile [0,1] veya [-1,1] aralığındaki değerlere dönüştürür. Doğrusal fonksiyon ise gelen girdileri aynen hücrenin çıktısı olarak yansıtır ve genellikle yapay sinir ağının çıktısı ile ilişkili olan hücrelerde kullanılır.

4.1.3 Katmanlar

Temel olarak, tüm yapay sinir ağları benzer bir yapıya sahiptir. Bu yapı içinde, bazı hücreler dış dünyadan girdileri alırken diğerleri dışarıya çıktıları vermekle görevlidir. Yapay sinir ağları, rasgele bağlanabilecek bir grup hücreden ibaret değildir. Bir yapı oluşturmanın en kolay yolu bir grup elemandan oluşan katmanlar yaratılmasıdır. Bu katmanlar arasındaki bağlantılar, toplama ve aktivasyon fonksiyonları işleyen bir yapay sinir ağını meydana getirir. Sadece tek katmandan oluşan, kullanışlı sinir ağları var olmakla birlikte, çoğu uygulama için sinir hücreleri, 3 katman halinde ve her katman içinde paralel olarak bir araya gelerek ağı oluştururlar. Bir yapay sinir ağında, en az bir girdi katmanı, sıfır veya daha fazla gizli (ara) katman ve bir çıktı katmanı bulunur. Girdi katmanı dışındaki tüm katmanlar, sinir hücreleri veya nöronlardan oluşur.

Girdi katmanı nöronları, verileri ya girdi dosyalarından ya da gerçek zamanlı uygulamalarda sensörlerden alırlar. Çıktı katmanı ise bilgiyi dış dünyaya, ikinci bir bilgisayar sürecine veya mekanik kontrol sistemlerinde olduğu gibi diğer cihazlara gönderir. Bu iki katman arasında

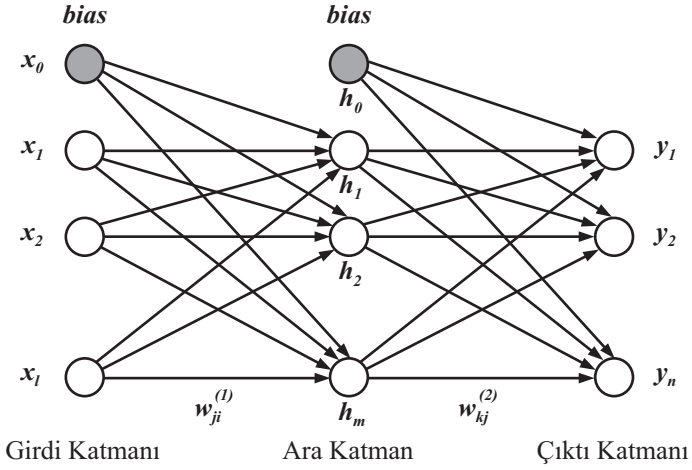
pek çok gizli katman olabilir, bu ara katmanlar çeşitli şekillerde bağlanmış pek çok sinir hücresinden oluşur. Yapay sinir ağının girdi katmanındaki nöron sayısı, sahip olunan girdi örneklerinin boyutuna eşittir. İstenen çıkışların sayısı ise, çıktı katmanındaki nöronların sayısını belirler. Genel olarak, yapay sinir ağındaki gizli katmanların sayısı ve her gizli katmandaki nöron sayısı ağın karmaşık fonksiyonları benzetme oranını belirler. Ancak bu, karmaşık ağların her zaman daha başarılı olacağı anlamına gelmez. Ağın temsil gücü ile, hesaba katacağı gürültü arasında bir denge vardır. Ağ ne kadar karmaşık hale gelirse, gürültüye karşı da o kadar hassas olur ve temeldeki fonksiyon ile birlikte girdilerdeki gürültünün de öğrenilmesi kolaylaşır.

4.2 Yapay Sinir Ağı Topolojileri

Tüm yapay sinir ağları; nöronlar, bağlantılar, toplama ve aktivasyon fonksiyonu kavramlarına dayandığı için, farklı tipteki ağların topolojileri bile birbirine benzer. Çeşitlemelerin çoğu, farklı öğrenme kuralları ve bu kuralların ağın tipik topolojisini değiştirme biçimi ile ortaya çıkar. Yapay sinir ağı topolojileri, ileri beslemeli ve geri dönüşümlü (geri beslemeli) ağlar olmak üzere, bağlantıların şekline göre, iki ana grupta toplanabilir. Bu çalışmada ileri beslemeli ve geri dönüşümlü topolojilere sahip iki yapay sinir ağı modeli kullanılmıştır.

4.2.1 İleri Beslemeli Ağlar

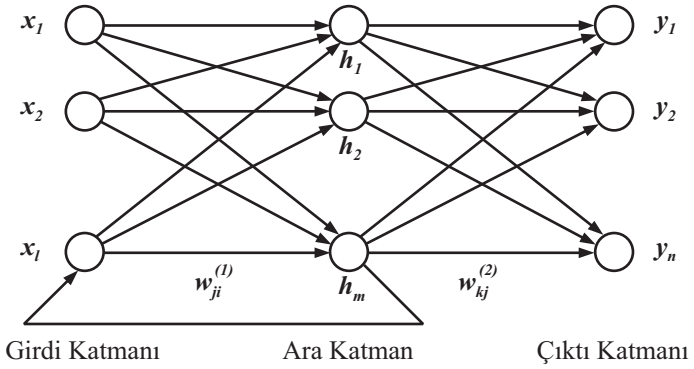
İleri beslemeli yapay sinir ağlarında, giriş birimlerinden çıkış birimlerine, veri akışı tamamen ileri doğrudur (Şekil 5.2). Birimlerin çıkışlarından, aynı katmanda veya önceki katmanlarda yer alan nöronların girişlerine bağlantı yapılmasına izin verilmez. Bu topoloji, yapay sinir ağına problemle ilgili tüm bilgilerin bir kerede giriş olarak verilebileceği, sonucun (çıktının) tamamen o andaki girdi değerlerine bağlı olduğu durumlarda kullanılır.



Şekil 4.2 İleri beslemeli yapay sinir ağı topolojisi

4.2.2 Geri Dönüşümlü Ağlar

Geri dönüşümlü ağlarda, önceki girişler ile ilgili bilgiler, ara veya çıkış katmanındaki nöronlardan geriye doğru yapılan bağlantılar vasıtası ile tekrarlanır ve girdilerle birleşir (Şekil 5.3). Geri dönüşümlü ağlar, ağa verilecek girişlerin sırasının önemli olduğu, önceki girişlerin kaydının bir şekilde saklanıp yeni girişler ile etkileşime sokulması istenen durumlarda kullanılır. Dinamik sistemlerin modellenmesinde ve öğrenilmesinde geri dönüşümlerin olması, özellikle zaman gecikmelerini dikkate almak için önemlidir.



Şekil 4.3 Geri dönüşümlü yapay sinir ağı topolojisi

Geri dönüşümlü ağlar, tam geri dönüşümlü ve kısmi geri dönüşümlü olmak üzere ikiye ayrılır. Tam geri dönüşümlü ağlarda, hepsi eğitilebilir olan, ileri ve geri doğru rasgele bağlantılar vardır. Bazı durumlarda, birimlerin aktivasyon değerleri bir dinlenme sürecine girer ve ağ artık aktivasyonların değişmediği kararlı bir duruma geçer. Ancak, ağın aynı cevapları veren durumları tekrarlayarak ya da her seferinde farklı bir çıktı üreterek kararlı bir duruma gelememesi de söz konusudur. Bağlantıların ağırlıklarına bazı sınırlamalar getirilerek ağın kararlı duruma geçmesi sağlanabilir. Tam geri dönüşümlü ağlar, daha çok optimizasyon problemlerinde, çağrışımsal bellek gibi kullanılır.

Kısmi geri dönüşümlü ağlarda, ağın işlem elemanlarına ek olarak, ara katman elemanlarının geçmiş durumlarını hatırlamak için kullanılan içerik elemanları vardır. Geri dönüşüm sadece içerik elemanları üzerinde yapılır ve bu bağlantılar eğitilemezler. Ancak ağın çıktıları, hem önceki durumlara hem de o anki durumlara bağlı olarak oluşmaktadır. Geçmiş durumları hatırlayabilmeleri nedeni ile bu ağların dinamik bir belleğe sahip olma özelliği vardır. İleri beslemeli ağların aksine, geri dönüşümlü ağların dinamik özellikleri önemlidir.

Kısmi geri dönüşümlü yapay sinir ağları sınıfında iki önemli mimari vardır. Elman (1990), ara katmandaki nöronlardan, ek girdi olarak kullanılacak bir grup içerik elemanlarına geri besleme yapılmasını önermiştir. Jordan (1986) ise, çıkış elemanlarından içerik elemanlarına

geri besleme yapan bir ağ tanımlamıştır. Kısmi geri dönüşümlü ağlar, popüler geriye yayma algoritmasının da kullanılmasına olanak verdiğinden, ileri beslemeli ağların basitliği ile tam geri dönüşümlü ağların karmaşıklığı arasında bir noktayı temsil eder.

4.3 Yapay Sinir Ağalarının Eğitilmesi

Yapay sinir ağı modellerinin gücü, bağlantı ağırlıklarının nasıl ayarlanma şeklinden kaynaklanır. Bu ağırlıkların, belirli bir veri kümesine göre, ayarlanması süreci ağın eğitilmesi olarak tanımlanır. Bu şekilde kullanılan veri kümesi ise eğitim kümesi olarak adlandırılır. Eğitimin temel amacı, ağın eğitim kümesinde varolan birtakım modelleri öğrenecek şekilde bağlantı ağırlıklarının ayarlanmasıdır. Bu şekilde ayarlanan ağ, gelecekteki durumları (bilinmeyen verileri) tahmin etme yeteneğine sahip olacaktır. Yapay sinir ağlarının eğitiminde, öğretmenli, öğretmensiz, destekleyici veya karma stratejiler kullanılır.

4.3.1 Öğrenme Stratejileri

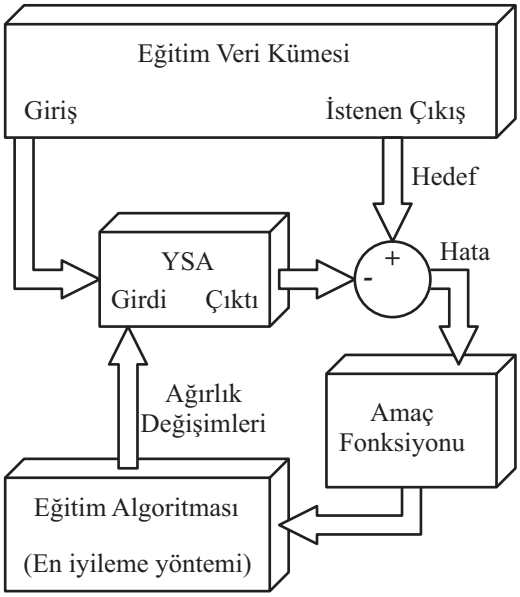
Öğretmenli öğrenmede, eğitim kümesi olarak hem girdiler hem de beklenen çıktılar ağa gösterilir. Ağ, girdileri işledikten sonra bulduğu çıktıları istenen çıktılar ile karşılaştırır. Hatalar, sistemde geri yayılarak ağı kontrol eden bağlantı ağırlıkları yeniden ayarlanır. Bir yapay sinir ağının eğitimi sırasında aynı veriler defalarca işlenerek bağlantı ağırlıkları düzeltilir. Öğretmenli öğrenme, pek çok yapay sinir ağı modelinde en yaygın olarak kullanılan stratejidir.

Öğretmensiz öğrenmede ise, eğitim kümesi olarak girdiler ağa gösterilir ancak istenen çıktılar verilmez. Sistem, verilen girdileri gruplamak ya da ilişkilendirmek için hangi özellikleri kullanacağına kendisi karar verir. Bu strateji genellikle, kendini örgütleme ya da uyarlama olarak adlandırılır. Öğretmensiz öğrenme alanında önde gelen araştırmacılardan biri olarak, Kohonen ağının (Kohonen, 1995) geliştiricisi, Tuevo Kohonen sayılır. Kohonen ağı, tek katmanlı ve pek çok bağlantı içeren, kendini örgütleyen bir ağ modelidir.

Destekleyici öğrenme stratejisinde, öğrenen sisteme bir öğretmen yardımcı olur. Öğretmen, sisteme çıktıları göstermek yerine, sistemin kendi kendine çıktı üretmesini bekleyerek bu çıktıların doğru veya yanlış olduğunu gösteren bir sinyal üretir. Bu stratejiyi kullanan sistemlere örnek olarak LVQ (Learning Vector Quantization) ağı gösterilebilir (Kohonen, 1988).

Kısmen öğretmenli, kısmen öğretmensiz olarak öğrenme yapan ağlar karma stratejili olarak sınıflanabilir. Bu tür ağlara örnek olarak radyal tabanlı ağlar (RBN) ve olasılık tabanlı ağlar (PBNN) verilebilir.

Bir yapay sinir ağının, öğretmenli öğrenme stratejisi ile, nasıl eğitildiği Şekil 4.4'de gösterilmiştir. Eğitim kümesindeki veriler, girdi ve hedef olmak üzere iki kısımdan oluşur. Başlangıçta, ağ bağlantılarının ağırlıkları rasgele atanabilir. İlk örnek verinin girdi kısmı ağa gösterildiğinde, ağ bu başlangıç ağırlıklarına göre bir çıktı hesaplar. Bu hesaplanan çıktı, örnek verinin hedef çıktısı ile kıyaslanır ve aradaki fark seçilen bir algoritma tarafından ağırlıkları düzeltmek için kullanılır. Bu süreç, verilen *x* girdilerine karşılık hedeflenen *t* çıktılarını veren bir *f(x)* fonksiyonunun bulunmasına benzetilebilir.



Şekil 4.4 Bir yapay sinir ağının eğitilmesi işlemi

Yapay sinir ağlarının eğitiminde, tüm eğitim verilerinin bir kerede ağa gösterildiği *toptan* (batch) ve her devirde bir örneğin ağa gösterildiği *artımlı* (incremental) olmak üzere iki tarz vardır (Mitchell, 1997). Aynı eğitim kümesinin tekrar işlenmek üzere ağa verilme sayısı ise *devir sayısı* (epoch) olarak adlandırılır.

4.3.2 Hata Fonksiyonu

Bir yapay sinir ağını eğitmek ve ne kadar başarılı olduğunu ölçebilmek için bir hata fonksiyonu tanımlanmalıdır. Ağ bağlantılarının ağırlıkları, bu hata fonksiyonunu azaltacak yönde ayarlanmalıdır. En yaygın olarak kullanılan hata fonksiyonlarından biri olan ortalama karesel hata (okh) aşağıdaki gibi ifade edilebilir:

$$okh = \frac{1}{Q}\sum_{k=1}^{Q} e(k)^2 = \frac{1}{Q}\sum_{k=1}^{Q} (t(k) - a(k))^2 \qquad (4.4)$$

p_Q ağın girdileri ve t_Q buna karşılık beklenen hedef çıktı olmak üzere, eğitim kümesi

$[p_1.t_t][p_2.t_2].....[p_Q.t_Q]$ şeklinde verilmiş olsun. Her girdi ağa gösterildiğinde, ağın hesapladığı a_Q çıktısı, hedef t_Q çıktısı ile kıyaslanır. Hata, hedef çıktı t ile ağın çıktısı a arasındaki fark olarak hesaplanır. Bundan sonra ağ bağlantılarının ağırlıkları bu farkların karelerinin toplamının ortalamasını küçültecek şekilde ayarlanır.

4.3.3 Öğrenme Kuralları

Yapay sinir ağlarının öğrenmesinde pek çok kural kullanılmakla birlikte, bunların çoğu, en iyi bilinen ve en eski öğrenme kuralı olan, Hebb kuralının (Hebb, 1949) çeşitlemeleridir. Öğrenme kurallarının en önemlilerinden bir kısmı aşağıda açıklanmıştır.

Hebb Kuralı: Bu kural ilk olarak Donald Hebb tarafından 1949 yılında yazdığı kitabında tanımlanmıştır. Bu kurala göre eğer bir nöron başka bir nörondan bir bilgi alırsa ve bu nöronların her ikisi de aktif ise, yani matematiksel olarak aynı işareti taşıyor ise, bu iki nöron arasındaki bağlantı güçlendirilmelidir.

Hopfield Kuralı: Hebb kuralına benzer ancak, kuvvetlendirme veya zayıflatmanın miktarı da belirlenir. Bu kural, her iki hücre de aktif veya her iki hücre de pasif ise bağlantı ağırlığının öğrenme katsayısı (oranı) kadar artırılması ya da azaltılmasını ister. Öğrenme katsayısı genellikle kullanıcı tarafından belirlenen 0–1 arasında sabit bir değerdir.

Delta Kuralı: Bu kural, Hebb kuralının biraz daha geliştirilmiş ve oldukça yaygın olarak kullanılan şeklidir. Delta kuralı, işlem biriminin çıktısı ile hedeflenen çıktı değeri arasındaki farkı (delta) azaltacak yönde, giriş bağlantılarının ağırlıklarının sürekli değiştirilmesi fikrine dayanır. Bu kural, bağlantı ağırlıklarını, ağın ortalama karesel hata değerini düşürecek şekilde değiştirir. Aynı zamanda, Widrow-Hoff öğrenme kuralı veya en küçük ortalama kareler (LMS) öğrenme kuralı olarak da adlandırılır. Delta kuralının çalışmasında, çıktı katmanındaki delta hatası, aktivasyon fonksiyonunun türevi ile dönüştürüldükten sonra, bir önceki katmandaki giriş bağlantı ağırlıklarının ayarlanmasında kullanılır. Diğer bir deyişle, her seferinde bir katmana olmak üzere, hata önceki katmanlara geri yayılır ve ilk katmana ulaşılana dek bu geri yayma devam eder. İleri beslemeli, geri yayılımlı yapay sinir ağları, ismini, bu şekilde hata terimi hesaplanmasından almıştır. Delta kuralı kullanılırken, ağın istenen doğruluk noktasına yakınsayabilmesi için, giriş verilerinin rasgele olmasına dikkat edilmelidir.

Eğim Düşümü Kuralı: Delta kuralına benzeyen eğim düşümü yönteminde de, ağırlıklara uygulanmadan önce, delta hatasının değiştirilmesinde aktivasyon fonksiyonunun türevi kullanılır. Ancak bu kuralda, ağırlıklara uygulanacak öğrenme katsayısına bir orantısal sabit

daha eklenir. Bu kural kullanıldığında kararlı duruma yakınsamanın gerçekleşmesi uzun sürmektedir. Ancak, çıkış katmanına daha yakın olan katmanlar için öğrenme katsayısının giriş katmanına yakın olanlardan daha küçük olacak şekilde seçilmesi halinde, yakınsamanın daha hızlı gerçekleştiği de gözlenmiştir.

Kohonen Kuralı: Teuvo Kohonen (Kohonen, 1988, 1995), bu kuralın geliştirilmesinde, biyolojik sistemlerin öğrenmesinden esinlenmiştir. Bu yöntemde, işlem elemanları ağırlıklarını değiştirmek ve öğrenmek için birbirleri ile yarışırlar. En büyük çıktıyı üreten hücre, kazanan hücre olarak yakınındaki hücrelerden daha kuvvetli hale gelir ve hem kazanan hücrelerin, hem de komşusu olan diğer hücrelerin ağırlıklarını değiştirebilir. Komşuluğun sınırları eğitim süresince değişim gösterebilir, genellikle büyük bir komşuluk ile başlanarak eğitim süreci ilerledikçe komşuluk sınırı daraltılır. Bu yöntem, verilerin istatiksel ve topolojik modellemesinde kullanılır ve kendini örgütleyen yapılar veya topolojiler olarak da adlandırılırlar.

5. VERİLERİN ELDE EDİLMESİ VE KULLANILAN YÖNTEMLER

5.1 PACE Ortamı ve Bileşenleri

Bu çalışmada bazı donanım ve ileti geçirme modelleri eklenerek genişletilen ve veri toplanmasında da kullanılan PACE (Nudd vd., 1999), yeni nesil performans analiz araçlarından biridir. PACE projesinde kullanılan yöntem, katmanlı yapıya sahiptir ve sistemin yazılım ve donanım bileşenleri bir paralelleştirme katmanı vasıtası ile birbirinden ayrılır (Nudd vd., 1993; Papaefstathiou, 1995; Papaefstathiou vd., 1994). PACE ortamında platformlar arası kıyaslamaların yapılması oldukça kolaydır, hali hazırda varolan modeller deneysel analiz amacı ile kolaylıkla değiştirilebilir ve yeniden kullanılabilirler. Farklı paralel ve dağıtık mimariler için elde edilen doğrulama sonuçları, PACE ile elde edilen sonuçlarda kabul edilebilir, %10 civarında, bir hatanın söz konusu olduğunu göstermektedir.

PACE içinde yer alan en önemli bileşen, katmanlı yapının gerçeklenmesi için yaratılmış olan, CHIP^3 S (Characterization Instrumentation for Performance Prediction of Parallel Systems) performans tanımlama dilidir.

5.1.1 Katmanlı Tanımlama Yapısı

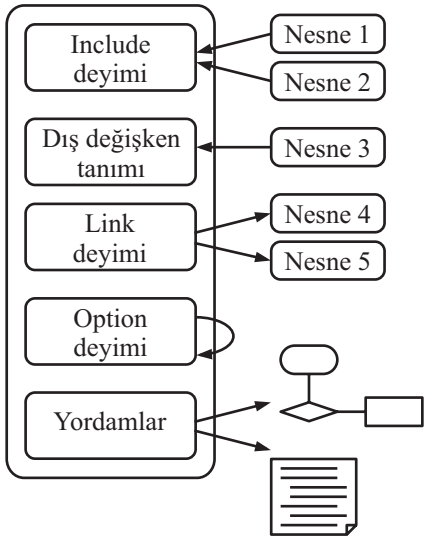
PACE projesinin amacı, oluşturulması nispeten kolay olan ve tekrar kullanılabilir modeller üzerine yoğunlaşarak, performans analizi alanında uzman olmayan kişilerin de kullanabileceği bir yöntem geliştirmektir.

PACE, tabanda yer alan dil vasıtası ile birbirine bağlanmış olan pek çok bileşenden oluşur.

Modüler yaklaşım, farklı kullanıcıların ihtiyaçlarına göre farklı seviyelerde tekrar kullanılabilir modeller oluşturulmasına olanak verir. PACE içinde yer alan nesneler, uygulama, alt görev, paralel şablon veya donanım tiplerindedir. Bu nesnelerin her biri ayrı bir katmanı temsil eder. Katmanlar, performansı tahmin etmek için değerlendirme aşamasında birleştirilen iyi tanımlanmış ara birimlerdir.

Donanım katmanının görevi, programın yürütüm süresini tahmin etmek için alt görev ve paralel şablon katmanlarından gelen kaynak kullanım bilgilerini değerlendirmektir. Kaynak kullanım bilgisi, uygulama ve paralelleştirme katmanlarının donanım ihtiyaçlarını sistem özelliklerinden bağımsız olarak tanımlar. Sistemdeki işlemciler, işlemciler arası haberleşme ağları ve giriş-çıkış birimleri gibi aygıtların modelleri donanım katmanı tarafından işleme dahil edilir. Böylelikle, tanımlanan sisteme ait performans sonuçlarını belirlemek üzere, değerlendirme motorunun kullanacağı hiyerarşik nesneler yaratılmış olur.

Paralel şablon veya donanım nesnelerinden herhangi biri farklı alt görev nesneleri tarafından tekrar kullanılabilir (Şekil 5.1).



Şekil 5.1 PACE'de yer alan nesnelerin yapısı

PACE içinde bulunan nesne tipleri ve bunların birbirleri ile olan ilişkileri aşağıda tanımlanmıştır.

Uygulama Tipi Nesne: CHIP^3 S performans tanımlama dilinde sadece bir uygulama tipi nesne kullanılabilir. Bu nesne otomatik olarak çalışma biriminden çağırılır. Kullanıcı, performans çalışmasına ait parametreleri uygulama nesnesine ait dış arayüz vasıtası ile değiştirebilir. Uygulama tipi nesne alt görev ve donanım nesnelerine ait dış değişkenleri değiştirebildiği gibi alt görev nesnelerinin tümünü kullanır.

Alt Görev Tipi Nesne: CHIP^3 S performans tanımlama dilinde birden fazla alt görev nesnesi kullanılabilir. Alt görev nesnesine ait arayüz uygulama nesnesi tarafından değiştirilebilir. Alt görev nesneleri, paralel şablon nesnelerini kullanır ve bu nesnelere ait dış değişkenlerin değiştirilmesine olanak verir.

Paralel Şablon Tipi Nesne: CHIP^3 S performans tanımlama dilinde birden fazla paralel şablon nesnesi kullanılabilir. Paralel şablon nesnesine ait arayüz alt görev nesnesi tarafından değiştirilebilir. Paralel şablon nesnesi ise başka hiçbir nesneye ait arayüzde değişiklik yapamaz. Bu nesne haberleşme ve işlem şablonlarını ve diğer donanım kaynaklarının kullanımını tanımlar.

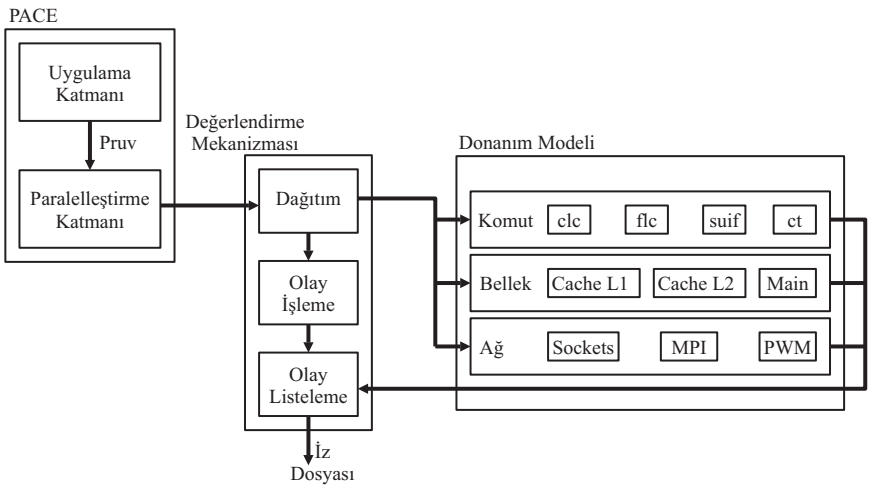
Donanım Tipi Nesne: CHIP^3 S performans tanımlama dili, donanım nesnelerinin tanımlanmasını desteklemez. Donanım nesneleri, diğer analitik ve simülasyon araçları

kullanılarak tanımlanmalıdır. Donanım nesnesinin yapısı ve parametreleri bir donanım simge dosyasında tanımlanmalıdır. Donanım nesnelerinin yapısı Şekil 5.2'de gösterilmiştir.

Donanım katmanı sistem bağımlı modelleri kapsarken, uygulama katmanı ve paralel şablon yazılım bağımlı modellerin tanımını yapar. Kaynak kullanım vektörü (RUV), bir görevi gerçekleştirmek için yazılımın ihtiyaç duyduğu donanım kaynaklarını gösterir. Uygulama katmanı hesaplama ihtiyaçlarını gösteren işlemci kaynak kullanım vektörünü (PRUV) tanımlar. Haberleşme kaynak kullanım vektörü (CRUV) ve giriş-çıkış kaynak kullanım vektörü (IORUV) ise paralel şablon katmanında tanımlanır ve ek yüklerin getirdiği ihtiyaçları gösterir.

Kaynak kullanım vektörünün (PRUV) oluşturulmasında farklı yöntemlerden yararlanılabilir. Kaynak kullanım vektörü olarak, yürütüm grafiğinin her bir düğümündeki işlem sayısı (flops) kullanılabilir ve belirli bir sistem için işlem oranı (Mflop/s) kullanılarak yürütüm süresi tahmin edilebilir. Bu çalışmada, üst düzey dil işleyici, alt düzey değerlendirmeler (Cmr) ile elde edilen maliyet bilgisini ve PRUV tarafından belirlenen işlem sıklığı bilgisini kullanır. Kullanılan alt düzey değerlendirme yordamları (Cmr) ile ilgili detaylı bilgi bölüm 5.3.5'de verilmiştir.

İşlemciler arası haberleşme ağı söz konusu olduğunda, kaynak kullanım vektörü tek bir formdadır. Ancak, bu formu oluşturmak için daha farklı haberleşme modelleri kullanılması da mümkün olabilir. Haberleşme modelleri, iletilerin uzunluklarını, haberleşme mesafesini ve ağ üzerindeki çekişmeleri (contention) dikkate alır. Bu modeller istatistiksel yollarla oluşturulur; regresyon parametreleri ölçümlerle tespit edilir, ileti uzunluk ve mesafeleri ise haberleşme kaynak kullanım vektörü CRUV tarafından sağlanır.



Şekil 5.2 PACE donanım nesnelerinin yapısı

5.1.2 Donanım ve Model Düzenleme Dili (HMCL)

PACE içindeki donanım modelleri, sisteme özel mikro değerlendirme ölçümlerinden, ilgili aygıtların analitik modellerine kadar değişik formlarda olabilir. Bu modellerin her biri, sistemin bireysel parçalarını tanımlayan, daha küçük bileşen modellerinden oluşur. Varolan bileşen modelleri, hesaplama, haberleşme, bellek ve giriş-çıkış olarak sıralanabilir. Bu modellerin esnek yapısı, donanım modellerinin kolaylıkla geliştirilmesine olanak verir.

Donanım ve Model Düzenleme Dili (HMCL), sistem bağımlı parametreleri belirleyerek, yeni donanım nesneleri oluşturmalarına olanak verir. Modeller, CHIP^{3}S çalıştırma birimine müdahale etmeye gerek kalmaksızın değiştirilebilir. HMCL dilinin sözdizimi oldukça basittir. Düzenleme dosyasının bir bölümü Şekil 5.3'de gösterilmiştir.

```
config SunUltra5_360 {
hardware {
Tclk = 1 / 360,
Desc = "SUN Ultra 5, U-SPARC II/360MHz, SunOS 5.7",
Source = "budweiser.dcs.warwick.ac.uk";
}
cache {
L1_CAPACITY = 16 * 1024,
L1_LINE_SIZE = 32,
L1_SECTOR_SIZE = 16,
……
clc {
SISL = 0.00584824,
SISG = 0.00595824,
SILL = 0.00623157,
component network ethernet  {
pvm {
DD_COMM_A = 4096 ,
DD_COMM_B = 3593.8 ,
DD_COMM_C = 1.40984 ,
}
}
SunUltra10 cluster_a[32];
ethernet net_a;
connectcluster_a[1..32] by net_a;
……
```

Şekil 5.3 Örnek HMCL parçası

5.1.3 PACE Ortamında Haberleşme Modellerinin Oluşturulması

İşlemciler arası haberleşme ağlarının performansını incelemek üzere farklı haberleşme modelleri geliştirilmiştir (Agarwal, 1991; Kleinrock, 1993). Bu haberleşme modelleri, ölçümler ile regresyon modellerinin çıkarıldığı istatistiksel tabanlı modeller veya ağın içsel çalışmasının işleyişine dayanan analitik modeller olabilir.

PACE metodolojisinin temelini oluşturan katmanlı yapı, diğer kısımlarda bir değişiklik yapmaya gerek kalmadan haberleşme modellerinin de değiştirilmesine olanak verir. İhtiyaç duyulan doğruluk oranına bağlı olarak, düşük hesaplama ve gerçekleme maliyeti olan basit modeller veya daha gelişkin analitik ve simülasyon modelleri kullanılabilir.

Bu çalışmada kullanılan modeller arasında, ileti boyutunu dikkate alan tek değişkenli regresyon modeli ve hem ileti boyutu hem de işlemciler arasındaki mesafeyi dikkate alan çift değişkenli model kullanılmıştır. Ağ üzerindeki çekişmeler sebebi ile meydana gelen, ek haberleşme gecikmelerini dikkate almak için de (Kerbyson vd., 1995) tarafından tanımlanan model uygulanmıştır.

5.1.3.1 İleti Boyutu Regresyon Modeli

Paralel sistemlerin haberleşme performansının incelendiği çalışmaların bazılarında (Hockney, 1993), haberleşme gecikme süresi ileti boyutuna bağlı olarak tahmin eden doğrusal regresyon modelleri kullanılmıştır. Bu modelin ifadesi, l ileti boyutu ve T_S başlangıç gecikmesi olmak üzere, şu şekilde verilebilir:

$$T_{comm}(l) = T_S + \frac{l}{b} \tag{5.1}$$

5.1.3.2 Mesafe Regresyon Modeli

Mesafe regresyon modeli, yukarıda verilen doğrusal regresyon modelinin uzantısı olup, sistemdeki herhangi iki işlemci arasındaki haberleşmeyi modellemek için kullanılabilir. Bu model, haberleşme gecikme süresini ileti boyutuna ve işlemciler arasındaki mesafeye bağlı olarak ifade eder:

$$T_{comm}[l,d] = T_S + r * d + f(b) * l \tag{5.2}$$

Verilen denklemde, l ileti boyutunu, T_S başlangıç gecikmesini, d işlemciler arasındaki mesafeyi, r yol atama modeline ait parametreyi ve b bant genişliğini ifade etmektedir. Bu model çeşitli yol atama modelleri için doğrulanmıştır (Bomans ve Roose, 1989).

Bu genel model daha sonra (Tron ve Plateau, 1994) aşağıda verildiği şekilde genişletilmiştir:

$$T_{comm} = \begin{cases} \alpha + d(\beta + \tau(l+h)) & l \le p_S \\ \alpha + d(\beta + \tau \cdot p\ ) + \left(\frac{1}{p_S}\right) \cdot (\alpha_2 + \beta + \tau \cdot \rho) & l > p_S \end{cases} \tag{5.3}$$

Verilen denklemde, β başlangıç gecikmesini, p_S paket boyutunu, α ileti hazırlama süresini,

α_2 paket hazırlama süresini, τ bant genişliğini, *h* paket başlığının boyutunu, *l* ileti boyutunu ve *d* işlemciler arasındaki mesafeyi ifade etmektedir.

Belirli bir sistem için, bu parametreler, Cmr yordamları ile elde edilen haberleşme ölçümlerinin regresyonu ile elde edilebilir. Haberleşme ölçümleri için kullanılan cmr yordamları, bölüm 5.3.5'de incelenmiştir.

5.1.4 PVM ve MPI İçin İşlemciler Arası Haberleşme Modellerinin Oluşturulması

PACE içinde yer alan haberleşme modelleri PVM veya MPI kütüphanelerinin kullanımını destekler. Paketleme, paket açma ve veri iletimi için kullanılan PVM komutları çeşitli fonksiyonlar ile temsil edilmektedir. Bu fonksiyonların çoğu, modellenen işlemi temsil eden tamsayı bir argüman alır.

"pvminitsend" fonksiyonu tampon belleği (buffer) harekete geçirmek için kullanılır ve programın kaynak kodunda "pvm_initsend" çağrısının bulunduğu yerlerde kullanılmalıdır. "pvmpack" ve "pvmunpack" fonksiyonları ise, çeşitli PVM paketleme ve paket açma çağrılarının modellenmesinde kullanılır.

PVM kullanılarak ileti geçirme, "pvmcom" fonksiyonu kullanılarak tek adımda veya gönderme ve alma için iki ayrı fonksiyon kullanılarak iki adımda olmak üzere, PACE içinde iki şekilde ifade edilebilir. "pvmcom" fonksiyonunun kullanılışı Şekil 5.4'de gösterilmektedir.

```
#include <pvmdefs.h>

...
            step pvminitsend {
                confdev PVM_DataDefault;
            }

...

    step pvmpack {
            confdev COUNT, TYPE;
    }

...
    step pvmcom {
            confdev SOURCE, DEST;
    }

...
```

Şekil 5.4 `pvmcom' fonksiyonunun kullanıldığı örnek PVM paralel şablonu

Alternatif olarak haberleşmenin iki basmağı, gönderim için "pvmsend" ve alım için

“pvmrecv” fonksiyonları kullanılarak, Şekil 5.5’de görüldüğü gibi ayrı olarak modellenebilir.

```
step pvmsend {
        confdev SOURCE, DEST [, MSG-SIZE [, TRANS-TYPE] ];
    }
   ...

    step pvmrecv {
        confdev SOURCE, DEST [, MSG-SIZE [, TRANS-TYPE] ];
    }
```

Şekil 5.5 `pvmsend' ve `pvmrecv' fonksiyonlarının kullanıldığı örnek PVM paralel şablonu

Her iki durumda da ileti boyutunun belirtilmesine gerek yoktur. İleti boyutu son “pvminitsend” fonksiyonundan itibaren paketlenen verilerin boyutlarından otomatik olarak hesaplanır.

MPI modelleri ise ileti boyutunun otomatik olarak hesaplanmasına olanak vermez. Dolayısı ile MPI haberleşme çağrıları ileti boyutunun “bayt-sekizli” olarak belirtilmesini gerektirir. PACE, MPI modeli fonksiyonlarının ihtiyaç duyacağı ileti parametreleri için, sadece “MPI_COMM_WORLD” ileticisini desteklemektedir.

Ayrıca, PACE işlemcileri 1 den Nproc’a (toplam işlemci sayısı) kadar numaralandırırken, MPI işlemcileri sıfırdan Nproc-1’e kadar sayar. Dolayısı ile MPI programlarını modellerken tüm işlemci kimlik numaraları PACE’in kullandığı düzene uygun olarak verilmelidir.

PACE, temel haberleşme yöntemi olarak “mpisend” fonksiyonu ile “MPI_Send” çağrısını ve “mpirecv” fonksiyonu ile de “MPI_receive” çağrısını modeller. Bu iki fonksiyonun gönderme ve alma şeklinde tam bir haberleşmeyi ifade ettiği durumlarda ikisinin yerine sadece “mpicom” fonksiyonu da kullanılabilir.

MPI kütüphanesi `MPI_Send' ve `MPI_Recv' dışında pek çok üst düzey çağrı içermektedir, bu çağrıların da büyük bir kısmı PACE içinde gerçekleştirilmiştir. Bu gerçekleştirimler, paralel şablonun kullanabileceği prosedürler şeklindedir. Bu prosedürler, ilgili işlemi birbirini izleyen “mpisend” ve “mpirecv” fonksiyonları ile gerçekleştirmektedir.

MPI kullanıcı kitabından (Gropp, 2005) verilen `particles.c' isimli örnek programın PACE ortamında kullanılmak üzere uyarlanan kodlar aşağıda Şekil 5.6’da gösterilmiştir.

```
#include <mpidefs.h>
    partmp ParParticles {
       include hardware;
       var compute: Tx;
       var numeric: N;
       option {
          seval = 1, nstage = 1;
       }
       proc exec init {
          var numeric :i;
          for (i = 1; i <= hardware.Nproc; i = i + 1) {
             call MPI_Send (i, 1, 20);
          }
          for (i = 1; i <= hardware.Nproc; i = i + 1) {
             call MPI_Recv (i, 1, 20);
          }
          call MPI_Allgather (4, 4, MPI_COMM_WORLD);
          call MPI_Gather (4, 4, 1, MPI_COMM_WORLD);
          call MPI_Allgatherv (N*16, N*16, MPI_COMM_WORLD);
          step cpu {
             confdev Tx;
          }
          call MPI_Barrier (MPI_COMM_WORLD);
       }
    }
```

Şekil 5.6 Örnek MPI paralel şablonu (Bu örnek, `particles.c' isimli örnek MPI programına karşılıktır)

5.1.5 Cmr Ölçüm Yordamları

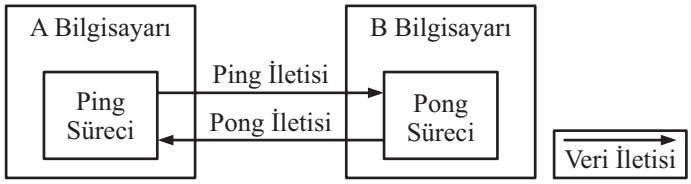
Cmr nitelik ölçüm yordamları, PACE donanım modellerini oluşturmakta da kullanılan, paralel programların performans tahmininde kullanılmak üzere ileti geçirme verilerini toplanması için oluşturulmuş bir yazılımdır. Bu yazılım hem PVM hem de MPI çağrılarının zamanlanması için kullanılabilir.

Cmr yazılımı, C komutlarının zamanlanması ve PVM ya da MPI haberleşmelerinin zamanlanması olmak üzere iki tip ölçüm yapar. Haberleşme zamanlarının ölçümü için kullanılan testler, noktadan noktaya iletişim, çoklu ileti çekişme yükü ve çoğa gönderilen ileti süreleri olmak üzere üç bölümden oluşur. Yazılım ayrıca, ileti gönderilirken PVM hayalet programının (deamon) oluşturduğu arka plan etkilerini ve eşzamansız ileti sürelerini de ölçmektedir. Kullanılan temel testler aşağıda kısaca açıklanmaktadır.

5.1.5.1 Pinpon (Ping-Pong) Testi

Şekil 5.7'de gösterilen basit pinpon testi Cmr yazılımının temelini oluşturur. Farklı uzunluklardaki iletileri iki süreç arasında geçirmek için, "Ping" anasistemindeki süreç bir zamanlayıcı tutmaya başlar ve "Pong" sürecine bir ileti yollar. "Pong" süreci, iletiyi alır almaz

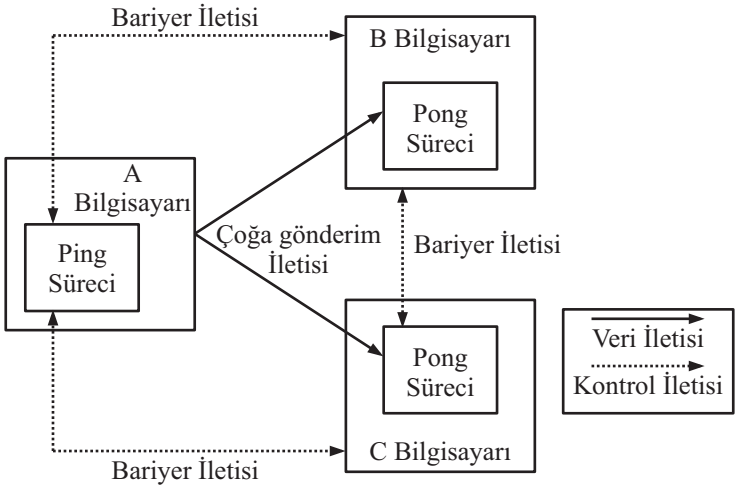
derhal “Ping” sürecine geri yollar ve “Ping” süreci geri gönderilen iletiyi alır almaz zamanlayıcıyı durdurur. Zamanlayıcı artık iki ileti yollamak için harcanan süreye ait değeri tutmaktadır.



Şekil 5.7 Pinpon Testi

5.1.5.2 Çoğa Gönderim Testi

PVM veya MPI süreçleri arasında, farklı uzunluktaki iletilerin birden fazla hedefe gönderilmesi için geçen süreyi ölçer. Bu testlerin işleyişi, birden fazla hedef süreç kullanılması dışında pipon testi gibidir. İki “Pong” sürecinin kullanıldığı bir çoğa gönderim testinin işleyişi Şekil 5,8’de gösterilmiştir. Bir “Ping” süreci, iletiyi tüm “Pong” süreçlerine gönderir. Ancak, sadece bir adet “Ping” süreci olduğu için sistem simetrik değildir ve tüm “Pong” süreçlerinin iletiyi geri yollamasından önce bir bariyer kullanılarak tüm süreçler eşzamanlı hale getirilir. Bariyerin oluşturulması için harcanan süre de ölçümlere dahil edilmekte olsa da, tüm sistemler benzer bir işlem gerçekleştireceği için, karşılaştırmalarda bu bir sorun yaratmayacaktır.



Şekil 5.8 Çoğa gönderim testi

5.1.5.3 Çekişme Yükü Ölçümü

PVM veya MPI süreçleri arasında, farklı uzunluktaki iletilerin hedefe gönderilmesi için geçen süreyi ölçer. Bu testlerde çekişme yükünün ölçülebilmesi için aynı anda birden fazla haberleşme yapılmaya çalışılmaktadır.

5.1.5.4 Arka Plan Yükünün Ölçümü

Cmr yazılımının bu kısmı, ileti gönderilirken, pvm_send() çağrısından hemen sonra, PVM hayalet programının (deamon) oluşturduğu arka plan etkilerini ölçmek için tasarlanmıştır. Kullanılan yöntem, koda pvm_send() çağrısından hemen sonra yoğun bir hesaplama işlemi ekleyip harcanan toplam süreyi ölçer. Daha sonra, aynı hesaplama işlemi boş bir sistem üzerinde çalıştırılarak zamanlanır ve aradaki fark hesaplanır.

5.1.5.5 Eşzamansız İleti Sürelerinin Ölçümü

Bu test, pvm_send() ve tıkanmasız pvm_recv() çağrılarının aldığı süreleri ölçer. Tıkanmasız haberleşme kullanımı, ileti geçiren programların performansında önemli iyileştirmeler sağlayabileceği için, önemlidir.

Bu testlerin her biri üç tip farklı zamanlayıcının kullanılmasına olanak verir:

- Gerçek: Toplam duvar saati süresini ölçer
- Sanal: Görevlerin ana işlemcide aktif oldukları süreyi ölçer
- Profil: Üst görevin aktif olduğu ve işletim sisteminin bu görev için harcadığı süreyi ölçer, bu süre alt öğe için beklenen süreyi de kapsar.

Ayrıca, byte, short, integer, long, float, double olmak üzere altı farklı ileti tipi kullanılabilir.

Bu testlerden elde edilen sonuçlar, PACE modellerinin oluşturulmasında olduğu gibi, çeşitli istatiksel analizler için kullanılabilir.

5.1.6 Regresyon Modellerinin Oluşturulması: En Küçük Kareler Yöntemi

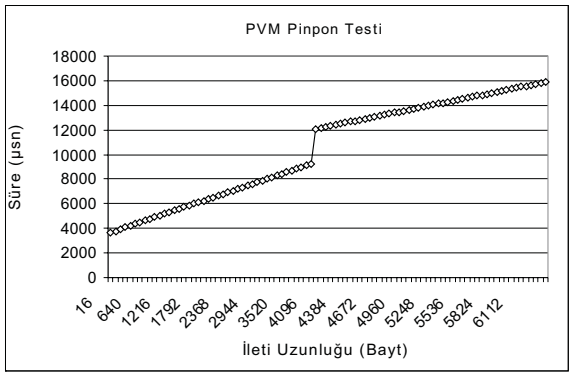
Cmr yazılımı ile toplanan test verilerine en iyi oturan doğruya ait denklemin bulunmasında en küçük kareler yöntemi kullanılmıştır. Bu yöntem, Cmr ileti geçirme testleri ile elde edilen verilerin her aşamasının sadece iki değer ile ifade edilmesine olanak verir. En Küçük Kareler yöntemi kullanılarak, $y = mx + b$ formundaki doğrusal modelin, b sabit terimi (düşey ekseni kesen nokta) ve m eğim parametresi hesaplanabilir. Her bir ileti büyüklüğüne (x) karşılık gelen mesaj geçirim haberleşme yükü (y) şeklinde N çift veri için b sabit terimi ve m eğim değeri (5.4) ve (5.5) eşitlikleri ile hesaplanır.

$$m = \frac{N(\sum xy) - (\sum x)(\sum y)}{N\left(\sum x^2\right) - (\sum x)^2} \tag{5.4}$$

$$b = \frac{(\sum y)\left(\sum x^2\right) - (\sum xy)(\sum x)}{N\left(\sum x^2\right) - (\sum x)^2} \tag{5.5}$$

En küçük kareler yönteminin güvenilirliği verilerin dağılımına bağlıdır, veriler ne kadar doğrusal ise elde edilen model o kadar güvenilir olacaktır. En küçük kareler modeli ile, veri kümesindeki her bir nokta için, hesaplanan y değeri ile veri kümesindeki gerçek y değeri arasındaki karesel fark alınarak "kalıntı hata oranı" ve toplam hata oranı hesaplanarak modelin ne kadar başarılı olduğu değerlendirilmelidir.

Şekil 5.9'da ve Çizelge 5.1'de, Cmr testleri ile elde edilen örnek veriler gösterilmektedir. Bu verilere en küçük kareler yöntemi uygulanarak, mesaj geçirme işleminin getireceği yükler sadece iki katsayıya bağlı denklemler ile ifade edilebilir (Çizelge 5.2).



Şekil 5.9 Sunsparc2 iş istasyonları ile yapılan PVM pinpon testine ait sonuçlar

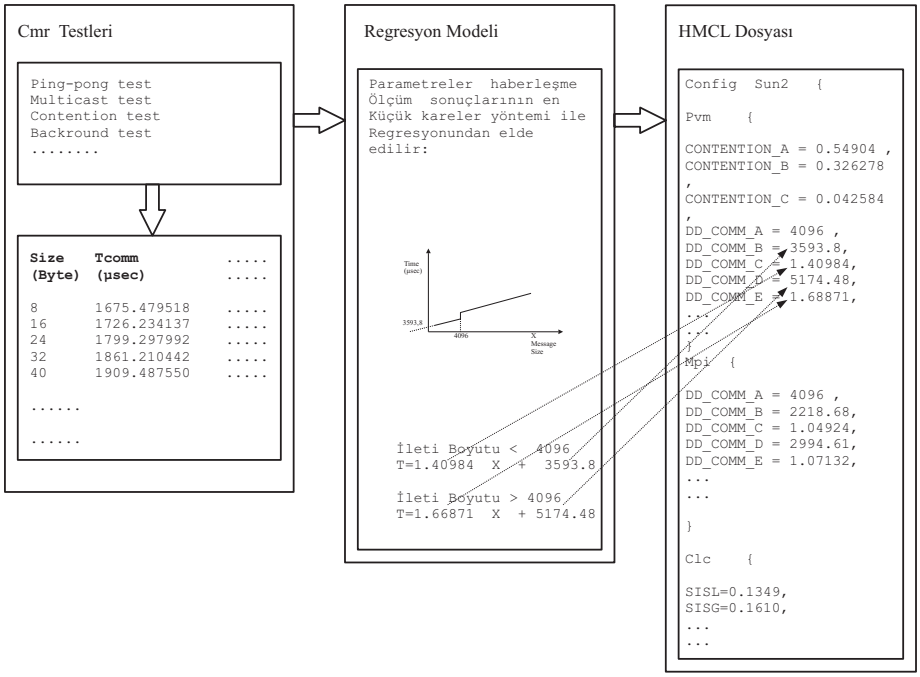
Çizelge 5.1 Örnek verilere ait sayısal değerler

İleti Boyutu (Bayt)	**Süre (µsn)**
16	3616.357
112	3751.702
…….	……
…….	……
4000	9233.16
4096	12091.44
4144	12172.49
…….	……
…….	……
6304	15820.11
6352	15901.17

Çizelge 5.2 Örnek veriler için en küçük kareler yöntemi ile elde edilen denklemler

İleti Boyutu < 4096	T = 1.40984 X + 3593.8
İleti Boyutu > 4096	T = 1.68871 X + 5174.48

Bu testlerden elde edilen sonuçlar çeşitli istatiksel analizler için kullanılabilir. PACE içinde donanım tanımlama dosyaları oluşturulurken bu verilerin nasıl kullanıldığı Şekil 5.10'da gösterilmiştir. Regresyon modellerinin oluşturuluşu ise bir sonraki bölümde açıklanmaktadır.



Şekil 5.10 Donanım tanım dosyalarının oluşturulmasında kullanılan yöntem

5.2 Verilerin Toplanması ve Kullanılan Yöntemler

Bu çalışmada sunulan yapay sinir ağları modellerinin eğitilmesinde kullanılmak üzere iki uygulama seçilmiştir. Warwick Üniversite'sinde yerel Ethernet ağı üzerindeki farklı özellik ve sayıdaki iş istasyonları kullanılarak bu uygulamalar çalıştırılmıştır. Bunun dışında verilerin çeşitliliğini ve miktarını artırabilmek için, halihazırda elimizde bulunmayan iş istasyonları ve ağ teknolojilerinin PACE üzerinde yukarıda açıklanan yöntemlerle oluşturulan modelleri kullanılarak, ikinci bir grup veri daha toplanmıştır. Özellikle, paralel programların gerektirdiği haberleşmenin toplam performans üzerindeki etkilerini görebilmek için, Ethernet ağ teknolojilerine ait verilerin elde edilmesi önemlidir.

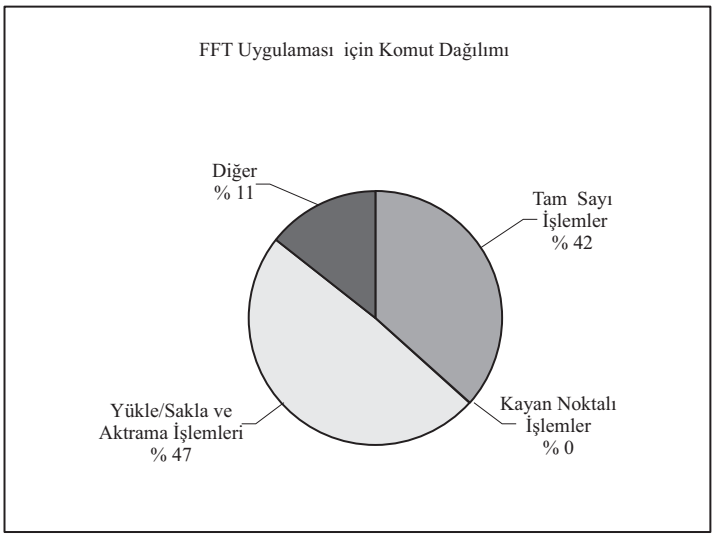
PACE üzerinde oluşturulan modellerin ve bu modeller kullanılarak toplanan verilerin güvenilirliği sorusunu ortadan kaldırmak için, önce varolan sistem üzerinde toplanan gerçek sonuçlar ile, aynı sistemin PACE ile oluşturulan modellerinin verdiği sonuçlar kıyaslanmıştır. Bu karşılaştırmalarda hata oranlarının kabul edilebilir olduğu, hata oranının her zaman %10'un altında kaldığı ve dolayısı ile bu modelleri kullanarak toplanan verilerin de gerçek sistem üzerinde toplanan veriler ile birlikte yapay sinir ağı modellerinin eğitim ve test süreçlerinde kullanılabileceği sonucuna varılmıştır. Yapay sinir ağı modellerinin test

aşamasında kullanılan veri kümeleri, sadece gerçek sistem üzerinde toplanan verileri veya hem gerçek sistem hem modellenmiş sistemler ile elde edilen verileri içerek şekilde gruplanmıştır.

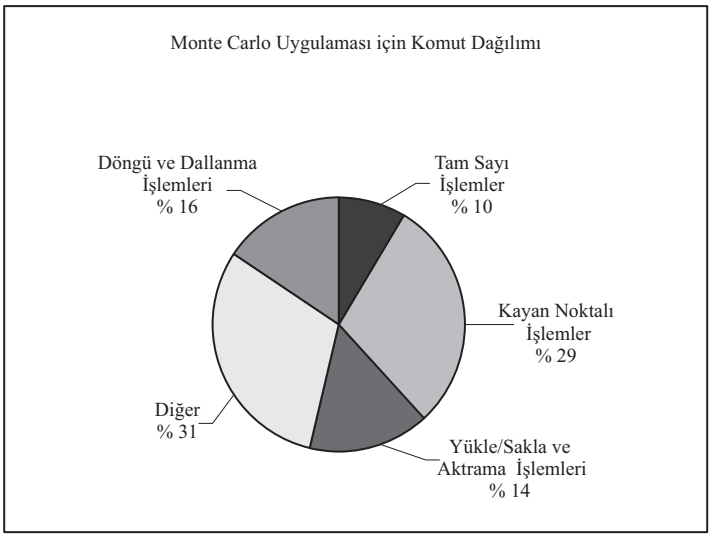
5.2.1 Seçilen Paralel Uygulamalar ve PACE'e Uyarlanması

Bu bölümde, testler için kullanılan paralel uygulamalar tanıtılmakta ve bu uygulamaların PACE ile oluşturulan modeller ile birlikte nasıl kullanılacağı hakkında bilgi verilmektedir.

Bu uygulamalardan iki-boyutlu Hızlı Fourier Dönüşümü (2-D FFT), işlemciler arasında yoğun veri alışverişi gerektirdiğinden özellikle haberleşme performansının ölçümünde kullanılmak üzere seçilmiştir. Bu uygulama, hiç kayan noktalı işlem içermediği ve tüm kodunun yaklaşık %42'si oranında tamsayı işlem içerdiği için, tamsayı işlem performansını ölçmek açısından da uygun bir seçimdir. Seçilen ikinci uygulama ise, diferansiyel denklem çözümleri içeren ve tipik bir yoğun kayan noktalı işlem uygulaması olarak sınıflanabilecek bir Monte Carlo uygulamasıdır (Jackel, 2002). Bu iki uygulamanın ortalama komut dağılımları Şekil 5.11 ve Şekil 5.12'de gösterilmiştir. Bu komut dağılımlarının oranları, C kodlarının analizi ile elde edilmiştir ve problem boyutuna göre küçük değişiklikler gösterebilir. Bu programların içerdiği her bir komutu, işlemci sayısı ve problem boyutunun bir fonksiyonu olarak gösteren formüller Ek 1'de verilmiştir.



Şekil 5.11 FFT Uygulaması için tipik komut dağılımı



Şekil 5.12 Monte Carlo Uygulaması için tipik komut dağılımı

5.2.1.1 İki-boyutlu Hızlı Fourier Dönüşümü (2-D FFT)

Fourier teorisine göre, bir fonksiyonun diğeri ile konvolüsyonunu bulmak için; iki fonksiyonun Fourier dönüşümlerinin çarpımı alınarak, bulunan sonuca ters Fourier dönüşümü uygulanır. Bu çalışmada kullanılan uygulama da iki-boyutlu bir dizi (resim) ile bir filtre dizisi arasında bir konvolüsyon işlemidir (Yavuz, 2005a).

Konvolüsyon teoremi Fourier teorisindeki en önemli ilişkilerden biridir. a ve b gibi iki fonksiyonun Fourier dönüşümlerinin sırasıyla A ve B olduğunu varsayarsak ab çarpımının Fourier dönüşümü A ile B'nin konvolüsyonu olarak adlandırılır ve (5.6) eşitliği ile ifade edilir.

$$C(x) = \sum_y A(y)B(x-y) \tag{5.6}$$

İki fonksiyonun çarpımı, bu iki fonksiyonun her noktadaki değerlerinin birbiri ile çarpımı ile elde edilir.

İki fonksiyonun konvolüsyonu alınırken ise, ilk fonksiyon mümkün olan her noktada diğeri üzerine getirilerek ikinci fonksiyonun bu noktada aldığı değer ile çarpılır. Konvolüsyon, tüm bu çarpımların toplamıdır.

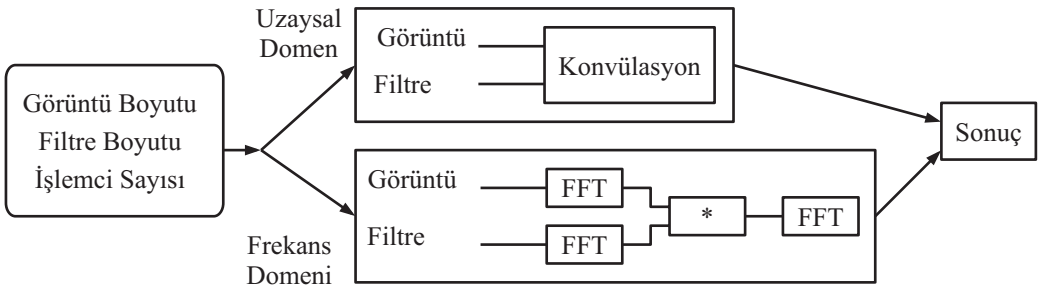
Bu çalışmada kullanılan uygulama ise iki boyutlu bir veri dizisine (görüntü) hızlı Fourier dönüşümü uyguladıktan sonra bu dizi ile bir filtre dizisinin çarpımını alan ve sonuca ters hızlı Fourier dönüşümü uygulayan bir konvolüsyon işlemidir (Şekil 5.13).

I() giriş olarak verilen görüntü, F() çıkış görüntüsü ve W ise konvolüsyon ağırlık fonksiyonu (filtre) olmak üzere, iki-boyutlu konvolüsyon işlemi (5.7) eşitliği ile ifade edilebilir:

$$F(i,j) = W * I(i,j) \tag{5.7}$$

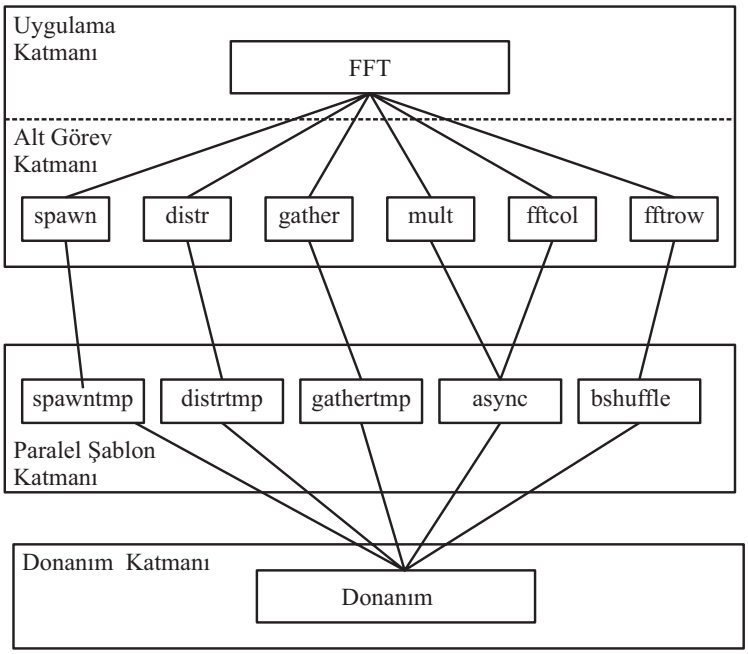
Burada (i,j) indisleri her bir noktayı (pikseli) temsil etmektedir. RxS boyutlu dörtgen bir filtre kullanıldığında konvolüsyon işlemi (5.8) eşitliği ile ifade edilebilir:

$$F(i,j) = \sum_{p=-R}^{R} \sum_{q=-S}^{S} W(p-q) * I(i-p, j-q) \tag{5.8}$$



Şekil 5.13 Uzay veya frekans domeninde konvolüsyon işlemi

Bahsedilen FFT uygulamasının, farklı donanım modelleri ile kullanılmak üzere PACE'e uyarlanmasında kullanılan, hiyerarşik katmanlı yapı diyagramı Şekil 5.14'de ve nesnelerin nasıl tanımlandığı da Şekil 5.15'de gösterilmiştir.



Şekil 5.14 FFT uygulamasına ait hiyerarşik katmanlı yapı diyagramı (HLFD)

```
#define FORWARD  (1)
#define BACKWARD (-1)
application fft {
include hardware;
(* subtasks *)
include spawn;
include distr;
include fftcol;
include fftrow;
include mult;
include gather;
(* Problem & System Size *)
var numeric:
ImageSize = 128,
FilterSize = 17,
Nproc = 8;
(* Globals *)
var numeric:
Mode;
link {
hardware:
Nproc = Nproc;
……
```

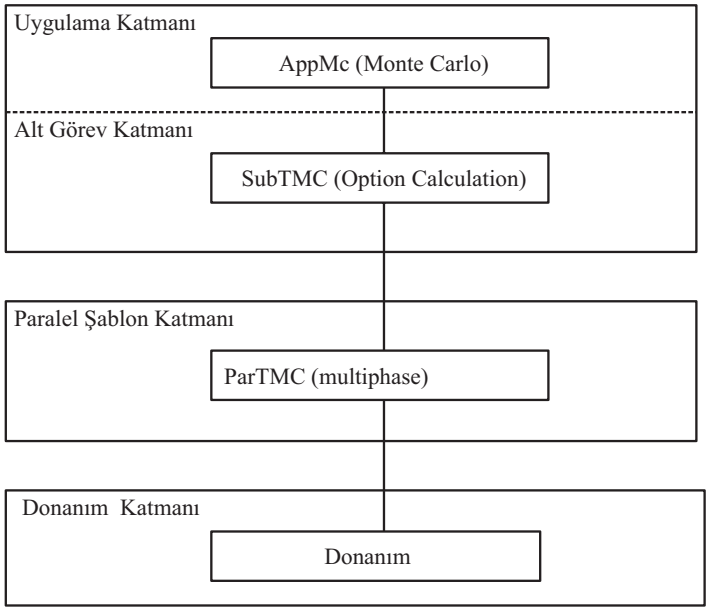
5.2.1.2 Monte Carlo Uygulaması

Seçilen paralel uygulamalardan ikincisi ise Monte Carlo simülasyon teknikleri kullanarak Avrupa opsiyon fiyatlarını hesaplayan bir programdır (Yavuz, 2005b).

Monte Carlo teknikleri, özellikle fiyatın tabandaki değişkenlerin önceki hareketlerine bağlı olduğu ve analitik olarak hesaplanamadığı durumlarda, finansal türevlerin fiyatlandırılmasında çok etkilidir.

Bu çalışmada kullanılan uygulamada, önce her bir işlemci önce varlıkların fiyatlarının yörüngesini hesaplar, ana işlemci bu sonuçları toplayarak bağımlı işlemcilere gönderir, bağımlı işlemciler kendi yerel hatalarını hesaplarlar ve son olarak ana işlemci bu bireysel hataları toplar.

Bu Monte Carlo uygulamasının, PACE'e uyarlanmasında kullanılan, hiyerarşik katmanlı yapı diyagramı Şekil 5.16'da ve nesnelerin nasıl tanımlandığı da Şekil 5.17'de gösterilmiştir.



Şekil 5.16 Monte Carlo uygulamasına ait hiyerarşik katmanlı yapı diyagramı (HLFD)

```
Application AppMC {
include SubTMC;
include hardware;
include Sun2;
var numeric:
NumTrials = 500,
NumSteps = 128,
NumVariates = 2,
Nproc = 1;
link{
SubTMC:
nrproc = CalcNrproc(),
Nsteps = NumSteps,
NumVariates = NumVariates,
NumTrials = NumTrials;
hardware:
Nproc = Nproc;
   }
option {
hrduse = "Sun2";
 }
 ……
```

Şekil 5.17 Monte Carlo uygulaması için uygulama nesnesinin tanımlanması

5.2.2 Deneysel Sonuçlar ve PACE Modellerinin Doğrulanması

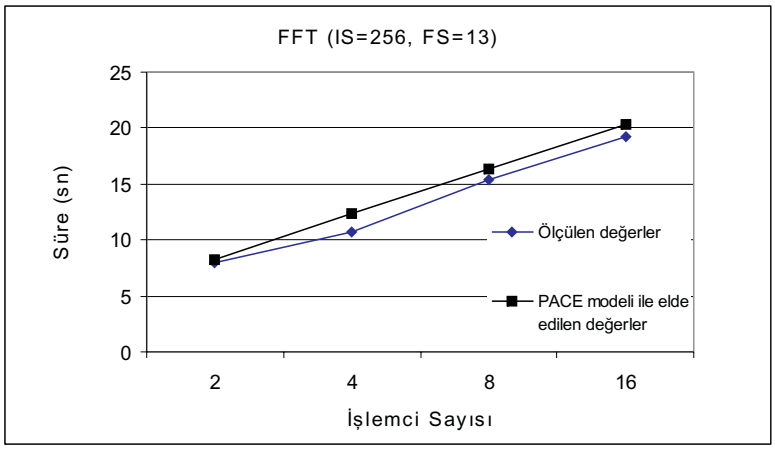
Bu bölümde sunulan deneyler sırasında, Warwick Üniversitesi'nde 10 Mbps Ethernet ağı üzerinde bulunan SunSparcStation5, Ultra1, Ultra5 ve Ultra10 iş istasyonları kullanılmıştır. Bu bölümde sunulan deneylerde, önce FFT ve Monte Carlo uygulamalarına ait paralel C

kodları ağ üzerindeki değişik sayıdaki iş istasyonu kullanılarak ölçümler yapılmıştır. İkinci aşamada ise haberleşme ağının ve iş istasyonlarının PACE için modelleri ve uygulamaların PACE kodları oluşturulmuştur. Gerçek ölçümlerde kullanılan iş istasyonundan oluşan konfigürasyonlar için, bu kez PACE ile çalışma zamanları tahmin edilmiştir.

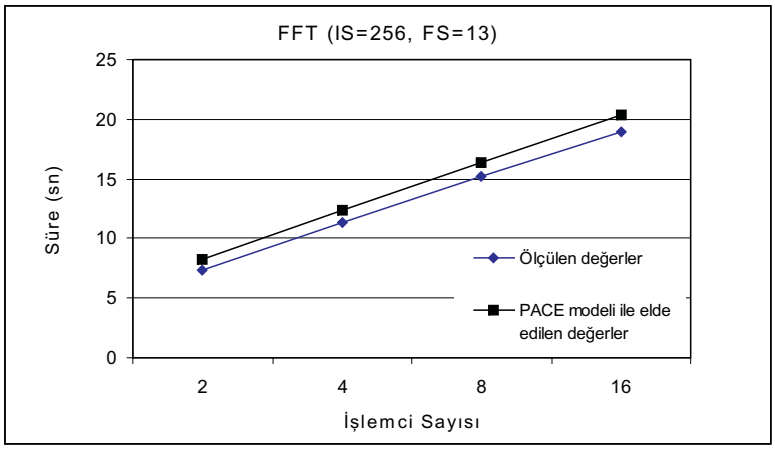
5.2.2.1 FFT Uygulaması ile Elde Edilen Sonuçlar

Bu bölümde, daha önce tanımlanmış olan FFT uygulaması için farklı model ve sayıdaki iş istasyonu kullanılarak ölçülen program yürütüm süreleri ve aynı konfigürasyonlar kullanılarak PACE ile tahmin edilen program yürütüm süreleri karşılaştırılmaktadır. 2 ila 16 işlemci kullanılarak ölçülen ve PACE ile tahmin edilen yürütüm süreleri, Sun-Ultra5 iş istasyonları için Şekil 5.18'de, Sun-Ultra10 iş istasyonları için ise Şekil 5.19'da gösterilmiştir.

Bu sonuçlar için yüzde hata oranları ve hataların standart sapma değerleri de hesaplanmıştır.



Şekil 5.18 FFT uygulamasının Sun-Ultra5 iş istasyonları üzerindeki sonuçlarının doğrulanması



Şekil 5.19 FFT uygulamasının Sun-Ultra10 iş istasyonları üzerindeki sonuçlarının doğrulanması

Çizelge 5.3, bu deneylere ait yüzde hata, ortalama hata ve standart sapma değerlerini göstermektedir. PACE ile tahmin edilen sonuçlar için yüzde hata oranları (5.9) eşitliği ile hesaplanmıştır:

$$Hata(\%) = \frac{|\text{ölçülen değer- tahmin edilen değer}|}{\text{ölçülen değer}} x100 \tag{5.9}$$

$Ort(x)$, veri kümesindeki değerlerin ortalamasını ve x_i, veri kümesindeki i. elemanı göstermek üzere n elemanlı bir veri kümesi için standart sapma (5.10) eşitliği ile hesaplanabilir:

$$Standart\ Sapma = \sqrt{\frac{\sum_{i=1}^{n} (x_i - Ort(x))^2}{n-1}} \tag{5.10}$$

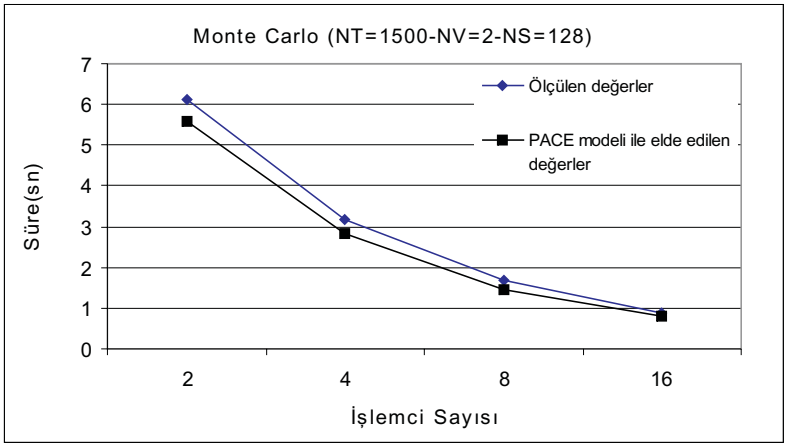
Çizelge 5.3 FFT uygulaması için PACE ile elde edilen sonuçların hata oranları

FFT Uygulaması (IS= 256, FS=13) İçin Elde Edilen Ölçüm ve PACE Tahmin Süreleri						
İşlemci Sayısı	Sun Ultra5 Ölçüm Değerleri (s)	Sun Ultra5 Tahmin Değerleri (s)	**Hata (%)**	Sun Ultra10 Ölçüm Değerleri (s)	Sun Ultra10 Tahmin Değerleri (s)	**Hata (%)**
2	8.014	8.302	3.597	7.329	8.201	11.897
4	10.760	12.420	15.428	11.385	12.339	8.377
8	15.388	16.388	6.499	15.236	16.326	7.154
16	19.164	20.365	6.269	18.935	20.335	7.394
	Ort. Hata(%) ± Std. Sapma		**7.94 ± 5.157**	**Ort. Hata(%) ± Std. Sapma**		**8.07 ± 2.19**

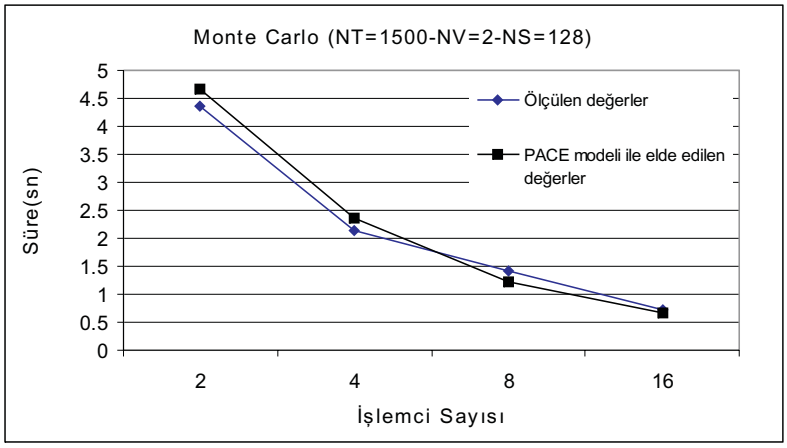
5.2.3 Monte Carlo Uygulaması ile Elde Edilen Sonuçlar

Bu bölümde, Monte Carlo uygulaması için farklı model ve sayıdaki iş istasyonu kullanılarak ölçülen program yürütüm süreleri ve aynı konfigürasyonlar kullanılarak PACE ile tahmin edilen program yürütüm süreleri karşılaştırılmaktadır. 2 ila 16 işlemci kullanılarak ölçülen ve PACE ile tahmin edilen yürütüm süreleri, Sun-Ultra1 iş istasyonları için Şekil 5.20'de, Sun-Ultra10 iş istasyonları için ise Şekil 5.21'de gösterilmiştir.

Çizelge 5.4 ise, bu deneylere ait yüzde hata, ortalama hata ve standart sapma değerlerini göstermektedir.



Şekil 5.20 Monte Carlo uygulamasının Sun-Ultra1 iş istasyonları üzerindeki sonuçlarının doğrulanması



Şekil 5.21 Monte Carlo uygulamasının Sun-Ultra10 iş istasyonları üzerindeki sonuçlarının doğrulanması

Çizelge 5.4 Monte Carlo uygulaması için PACE ile elde edilen sonuçların hata oranları

Monte Carlo Uygulaması İçin Elde Edilen Ölçüm ve PACE Tahmin Süreleri (NT= 1500, NV=2, NS=128)						
İşlemci Sayısı	Sun Ultra1 Ölçüm Değerleri (s)	Sun Ultra1 Tahmin Değerleri (s)	**Hata (%)**	Sun Ultra10 Ölçüm Değerleri (s)	Sun Ultra10 Tahmin Değerleri (s)	**Hata (%)**
2	6.124	5.588	8.750	4.368	4.657	6.610
4	3.181	2.815	11.516	2.132	2.349	10.177
8	1.684	1.443	14.325	1.412	1.210	14.313
16	0.892	0.792	11.217	0.723	0.675	6.700
	Ort. Hata(%) ± Std. Sapma		**11.45 ± 2.28**	**Ort. Hata(%) ± Std. Sapma**		**9.45 ± 3.64**

FFT ve Monte Carlo uygulamaları kullanılarak yapılan testler sonucunda, PACE ile elde edilen sonuçların, hata oranının ortalama %10 civarında olduğu gözlenmiştir. Geliştirilen yapay sinir ağı modellerinin eğitim ve testinde, ölçüm yolu ile elde edilen verilerin yanı sıra, PACE ile elde edilecek ek verilerin de kullanılabileceği sonucuna varılmıştır. Bu amaçla oluşturulan donanım ve haberleşme ağı modellerinin özellikleri ve bu modelleri kullanarak elde edilen sonuçlar bir sonraki bölümde açıklanmaktadır.

5.2.4 PACE Modelleri ile Elde Edilen Verilerin İncelenmesi

5.2.4.1 Oluşturulan Modellerin Özellikleri

Bu bölümde, Donanım ve Model Düzenleme Dili (HMCL) kullanılarak, bir grup yeni nesil ürüne ait modeller oluşturulmuştur. Bu modellerin oluşturulması sırasında, ürünlere ait veri föylerinin yanı sıra, SpecCpu2000 karşılaştırmalı değerlendirme sonuçlarında belirtilen değerlerden de yararlanılmıştır [10]. Ticari markaları da verilen makinelere ait modellerin oluşturulması aşamasında bellek boyutu ve işlemci hızı gibi değişken olabilen değerlerden sadece bir tanesi seçilmiştir. Çizelge 5.5 bu makinelerin temel özelliklerini - modeller oluşturulurken kullanılan değerleri - ile göstermektedir. Bu makinelerin diğer özellikleri, mimarilerine göre sınıflanmış olarak 2. bölümde verilmiştir.

Çizelge 5.5 Modellenen makinelerin temel özelliklerini

	Dell Itanium 800 MHz	**HP Itanium 733 MHz**	**Sun Blade 1000 Model 1600**	**Sun Blade 1000 Model 1750**	**Sun Blade 1000 Model 1900**
Merkezi İşlemci Hızı	800 MHz	733 MHz	600 MHz	750 MHz	900 MHz
Ana Bellek	8 Gb	8 Gb	1 Gb	1 Gb	1 Gb
Cep Bellek	L1: 16Kb Veri, 16Kb Komut L2: 96Kb L3: 4Mb	L1: 16Kb Veri, 16Kb Komut L2: 96Kb L3: 4Mb	L1: 34Kb Veri, 64Kb Komut L2: 8Mb	L1: 34Kb Veri, 64Kb Komut L2: 8Mb	L1: 34Kb Veri, 64Kb Komut L2: 8Mb
FPU (Kayan Noktalı İşlem Birimi) Sayısı	4 FMAC	4 FMAC	3	3	3
IEU (Tamsayı İşlem Birimi) Sayısı	4	4	4	4	4

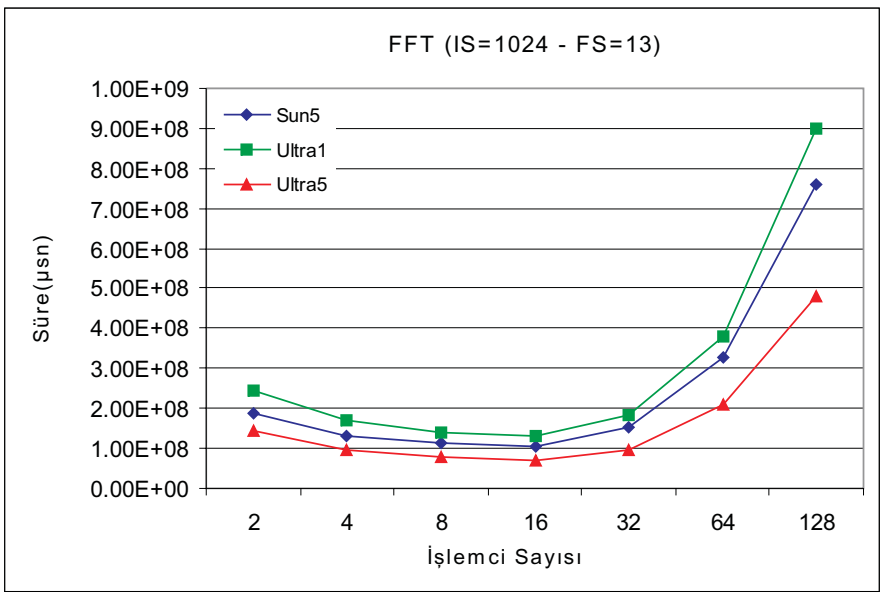
5.2.4.2 PACE Ortamında Modeller ile Yapılan Deneyler ve Elde Edilen Veriler

Belirlenen makinelere ait modellerin oluşturulmasından sonra, FFT ve Monte Carlo uygulamaların PACE kodları bu modeller kullanılarak çalıştırılmıştır. Varolan makineler kullanılarak toplanan veriler ise, 128 ve 256 işlemci kullanılması ve farklı ağ teknolojilerinin

kullanılması durumundaki sonuçları da içerecek şekilde artırılmıştır. Bu bölümde elde edilen verilerden bir kısmı seçilerek sunulmuştur. Çizelge 5.6 ve Şekil 5.22 aynı boyuttaki FFT probleminin, farklı iş istasyonları kullanılarak elde edilen sonuçlarını göstermektedir. Çizelge 5.7 ve Şekil 5.23 ise değişik boyutlardaki FFT probleminin, Sun Ultra 5 iş istasyonları kullanılarak elde edilen sonuçlarını göstermektedir. Bu iki gruptaki ölçümler için 10 Mbps Ethernet haberleşme ağı kullanılmıştır ve grafiklerden izlenebildiği gibi işlemci sayısının artışı ile haberleşme yükü çok arttığı için programın paralel çalışması etkin olmaktan çıkmaktadır.

Çizelge 5.6 FFT uygulaması ile farklı iş istasyonları üzerinde elde edilen sonuçlar (IS=1024,FS=13)

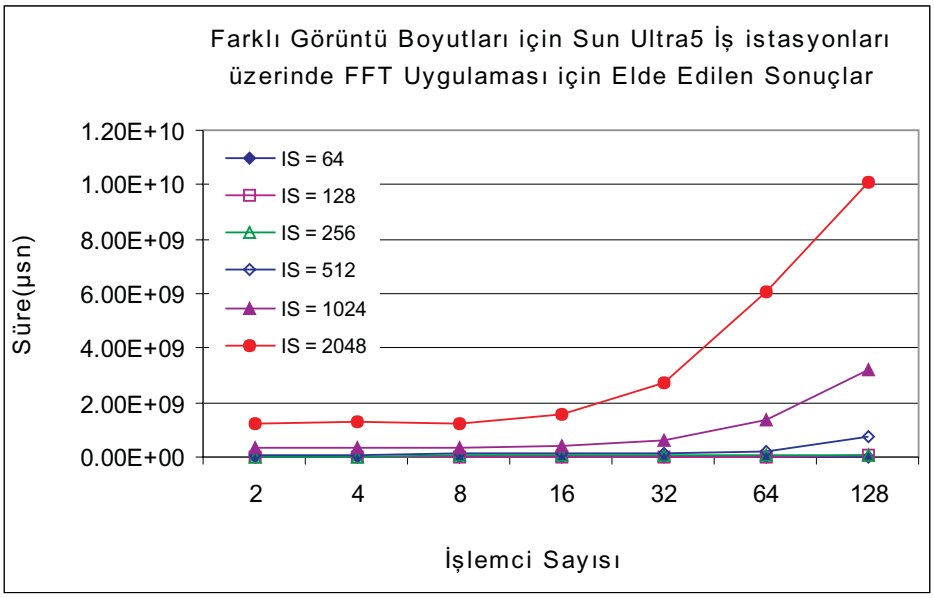
		FFT Görüntü Boyutu = 1024, Filtre Boyutu = 13		
		Yürütüm Süresi (µsn)		
		Sun5(170 MHz)	**Ultra1**(143 MHz)	**Ultra5** (270 MHz)
İşlemci Sayısı	**2**	1.87E+08	2.46E+08	1.43E+08
	4	1.32E+08	1.70E+08	9.56E+07
	8	1.11E+08	1.41E+08	7.74E+07
	16	1.06E+08	1.31E+08	7.00E+07
	32	1.52E+08	1.82E+08	9.74E+07
	64	3.26E+08	3.80E+08	2.11E+08
	128	7.58E+08	8.99E+08	4.78E+08



Şekil 5.22 FFT uygulaması ile farklı iş istasyonları üzerinde elde edilen sonuçlar (IS=1024,FS=13)

Çizelge 5.7 FFT uygulamasın farklı görüntü boyutları için Sun Ultra 5 iş istasyonları kullanılarak elde edilen sonuçları (FS=55)

		Sun Ultra 5 iş istasyonları, Filtre Boyutu=55					
		Yürütüm Süresi (µsn)					
Görüntü Boyutu		64	128	256	512	1024	2048
İşlemci Sayısı	2	3.44E+06	2.62E+07	2.43E+07	7.93E+07	3.17E+08	1.21E+09
	4	5.63E+06	1.24E+07	3.27E+07	9.52E+07	3.16E+08	1.28E+09
	8	7.67E+06	1.64E+07	3.96E+07	1.08E+08	3.51E+08	1.22E+09
	16	9.96E+06	2.04E+07	4.77E+07	1.23E+08	3.78E+08	1.54E+09
	32	1.28E+07	2.56E+07	5.89E+07	1.54E+08	6.03E+08	2.75E+09
	64	1.67E+07	3.24E+07	7.49E+07	2.07E+08	1.39E+09	6.10E+09
	128	1.97E+07	4.21E+07	9.94E+07	7.19E+08	3.22E+09	1.01E+10

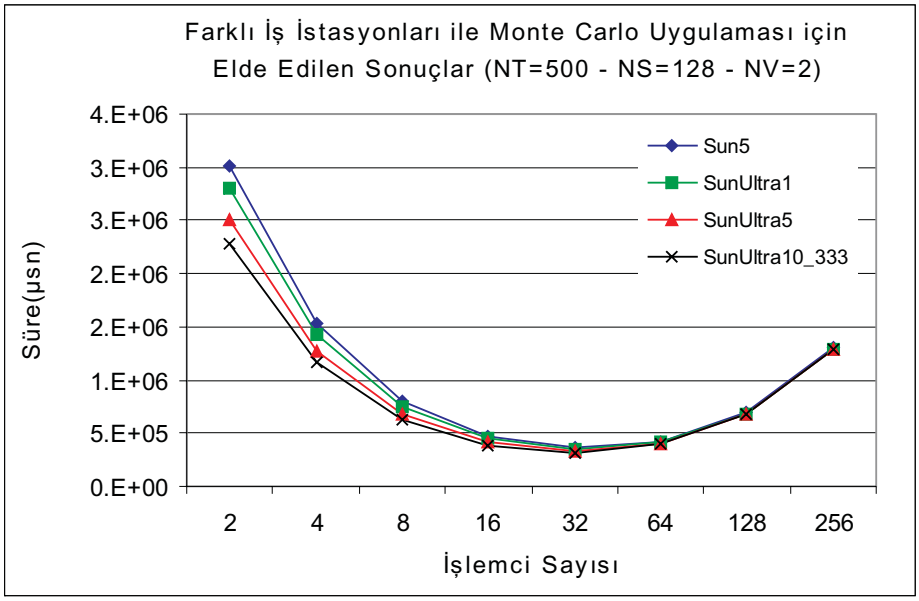


Şekil 5.23 FFT uygulamasın farklı görüntü boyutları için Sun Ultra 5 iş istasyonları kullanılarak elde edilen sonuçları (FS=55)

Paralel Monte Carlo uygulaması ise, FFT uygulamasının tersine, fazla haberleşme gerektirmeyen aritmetik işlem yoğunluklu bir uygulama olduğu için, yavaş Ethernet haberleşme ağına rağmen daha başarılı sonuçlar vermiştir. Ancak yine de paralellik 32 işlemciden sonra etkinliğini kaybetmektedir. Çizelge 5.8 ve Şekil 5.24 aynı boyuttaki Monte Carlo probleminin, farklı iş istasyonları kullanılarak elde edilen sonuçlarını göstermektedir.

Çizelge 5.8 Monte Carlo uygulaması için farklı iş istasyonları ile elde edilen sonuçlar (NT=500,NS=128,NV=2)

		Monte Carlo (NT=500,NS=128,NV=2)			
		Yürütüm Süresi (µsn)			
		Sun5 (170 MHz)	**Ultra1** (143 MHz)	**Ultra5** (270 MHz)	**Ultra10** (333 MHz)
İşlemci Sayısı	**2**	3.E+06	3.E+06	3.E+06	2.E+06
	4	2.E+06	1.E+06	1.E+06	1.E+06
	8	8.E+05	8.E+05	7.E+05	6.E+05
	16	5.E+05	4.E+05	4.E+05	4.E+05
	32	4.E+05	3.E+05	3.E+05	3.E+05
	64	4.E+05	4.E+05	4.E+05	4.E+05
	128	7.E+05	7.E+05	7.E+05	7.E+05
	256	1.E+06	1.E+06	1.E+06	1.E+06

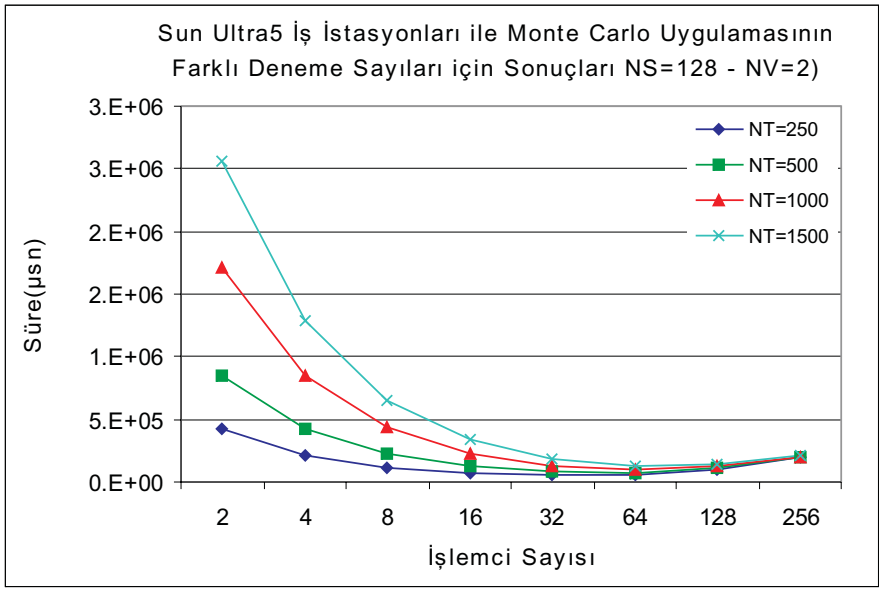


Şekil 5.24 Monte Carlo uygulaması için farklı iş istasyonları ile elde edilen sonuçlar (NT=500,NS=128,NV=2)

Çizelge 5.9 ve Şekil 5.25 ise problem boyutunun performans üzerindeki etkilerini yakalamak için, değişik boyutlardaki Monte Carlo problemi için, Sun Ultra 5 iş istasyonları kullanılarak yapılan ölçümlerin sonuçlarını göstermektedir.

Çizelge 5.9 Sun Ultra 5 iş istasyonları üzerinde Monte Carlo uygulamasının farklı deneme sayıları için sonuçları

		Sun Ultra 5 iş istasyonları - Monte Carlo (NT=500,NS=128,NV=2)			
		Yürütüm Süresi (µsn)			
Deneme Sayısı		250	500	1000	1500
İşlemci Sayısı	2	3.E+06	3.E+06	3.E+06	2.E+06
	4	2.E+06	1.E+06	1.E+06	1.E+06
	8	8.E+05	8.E+05	7.E+05	6.E+05
	16	5.E+05	4.E+05	4.E+05	4.E+05
	32	4.E+05	3.E+05	3.E+05	3.E+05
	64	4.E+05	4.E+05	4.E+05	4.E+05
	128	7.E+05	7.E+05	7.E+05	7.E+05
	256	1.E+06	1.E+06	1.E+06	1.E+06



Şekil 5.25 Monte Carlo uygulamasın farklı deneme sayıları için Sun Ultra 5 iş istasyonları kullanılarak elde edilen sonuçları

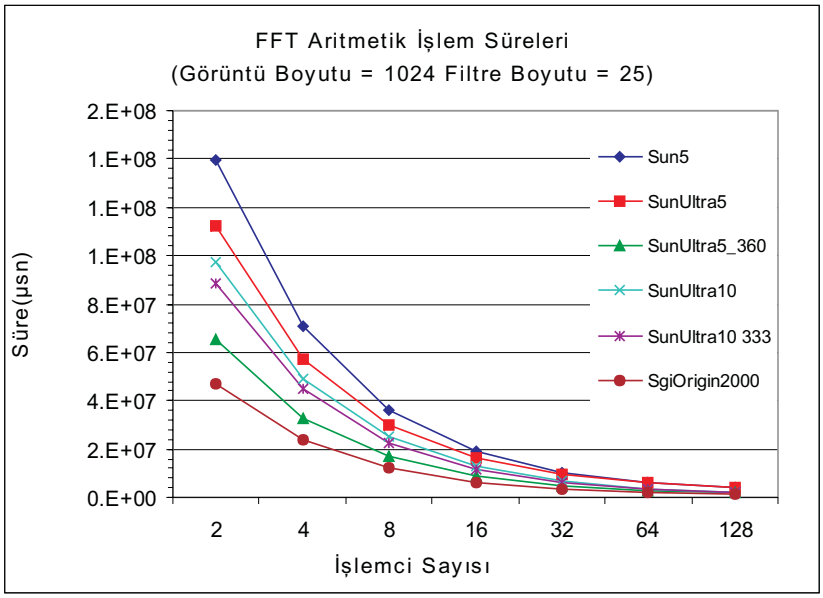
5.2.5 Aritmetik İşlem Performanslarının Ölçülmesi

Önceki bölümde sunulan sonuçlardan da görüldüğü gibi paralel sistemlerin performansının değerlendirilmesinde işlem ve haberleşme performanslarının bağımsız olarak değerlendirilmesi faydalı olacaktır. Varolan sistemler ve PACE için oluşturulan modeller kullanılarak yapılan, çok sayıda ölçüm ile seçilen uygulamalar için toplam yürütüm süreleri belirlendikten sonra, programların işlem ve haberleşme için harcadıkları süreler birbirinden ayrılmıştır.

Programlara ait kodlardan farklı işlemler için harcanan süreler, işlemci sayısı ve problem boyutunun fonksiyonu olarak formüle edilmiştir (Ek 1).

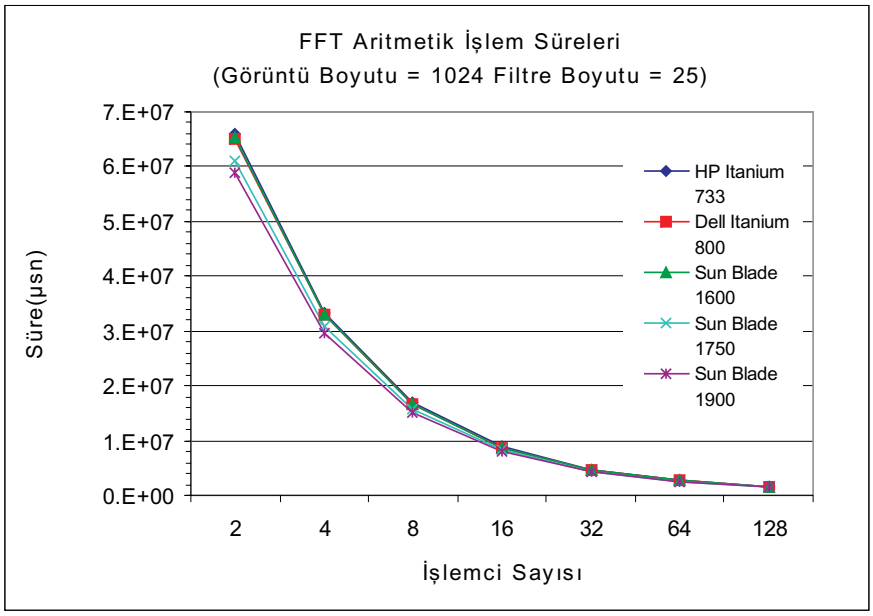
Bu bölümde, ilk olarak değişik boyutlardaki FFT uygulamasının farklı iş istasyonları üzerindeki aritmetik işlem performansları incelenecektir. FFT uygulaması, yoğun miktarda tamsayı işlem içermekle birlikte hiç kayan noktalı işlem içermediği için, işlemcilerin tamsayı ve kayan noktalı işlem performanslarını karşılaştırmak açısından da önemlidir.

Şekil 5.26'da verilen sonuçlar incelendiğinde Sun Ultra 5 modelinin 360 MHz işlemcisi ile aynı mimariye ancak 333 MHz işlemciye sahip Ultra 10 modelinden daha iyi performans gösterdiği görülmektedir. Daha önemlisi diğerlerinden farklı olarak paylaşımlı bellek mimarisine ve 350 MHz işlemciye sahip SGI Origin 2000 modeli en iyi işlem performansı göstermektedir. Bu durum, FFT programının aynı zamanda yoğun veri alışverişi gerektirmesi ile açıklanabilir.



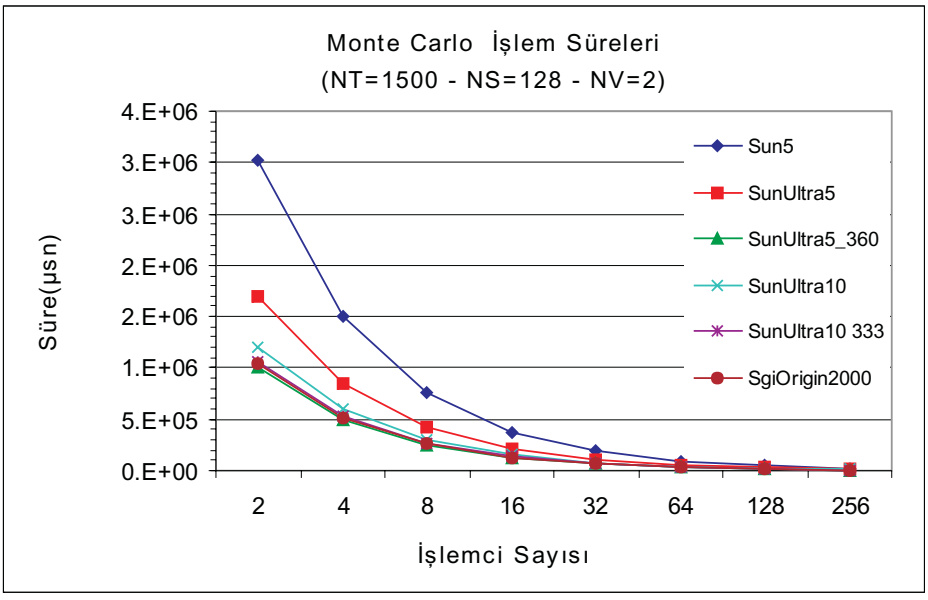
Şekil 5.26 FFT uygulaması için aritmetik işlem süreleri (IS=1024, FS=25)

Şekil 5.27 SunBlade 1000 iş istasyonlarının farklı modelleri ile iki farklı Itanium sunucusuna ait sonuçları göstermektedir. SunBlade 1600 model iş istasyonu 600 MHz işlemcisine rağmen, 733 ve 800 MHz hızında işlemcilere sahip Itanium sunucularına yakın bir performans göstermektedir. Bu sonuçlar, Sun Blade mimarisinin, tamsayı aritmetik işlem performansı açısından, biraz daha başarılı olduğu şeklinde değerlendirilebilir. Bu grupta test edilen makineler arasında, 900 MHz merkezi işlemci birimi avantajına da sahip olan, SunBlade 1900 en yüksek performansı göstermiştir.



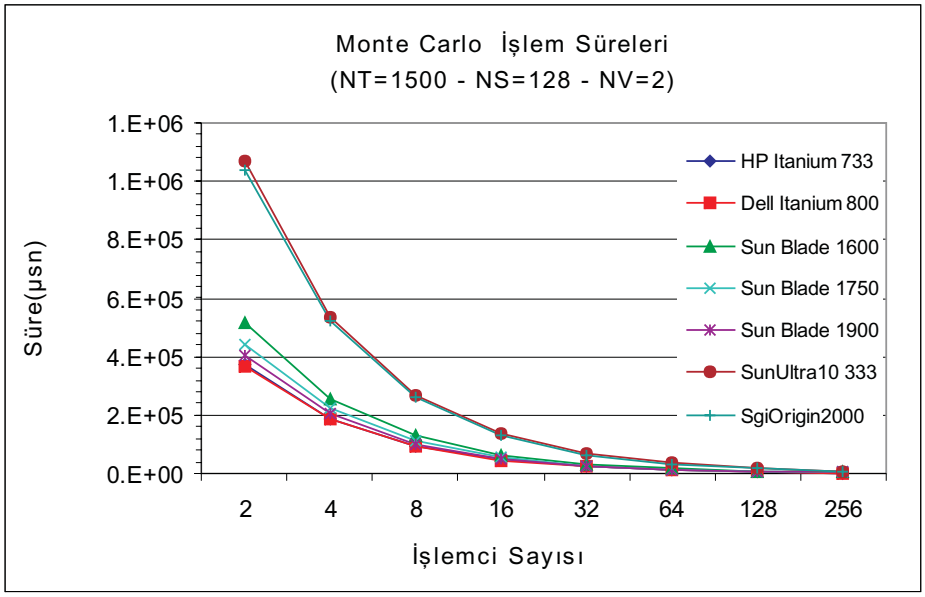
Şekil 5.27 FFT uygulaması için aritmetik işlem süreleri (IS=1024, FS=25)

Monte Carlo uygulaması için aritmetik işlem performansları incelendiğinde (Şekil 5.28), FFT uygulaması için elde edilenden farklı bir tablo gözlenmektedir. Merkezi işlem biriminin hızı veya paylaşımlı bellek mimarisi, bu uygulama için önemli bir fark yaratmamaktadır.



Şekil 5.28 Monte Carlo uygulaması için, Sun iş istasyonları üzerinde, elde edilen aritmetik işlem süreleri (NT=1500, NS=128, NV=2)

Şekil 5.29, Sun Blade iş istasyonları ile Itanium sunucuların aritmetik işlem performanslarını karşılaştırmaktadır. Itanium sunucular merkezi işlem birimlerinin hızları farklı olmasına rağmen birbirine yakın ve Sun Blade iş istasyonlarından daha iyi performans göstermiştir. SunBlade 1900 modeli işlem performansı olarak Itanium sunucuların fazla gerisinde olmamakla birlikte, sunulan gruptaki en hızlı merkezi işlem birimine sahip olduğu göz önüne alındığında, Itanium sunucuların kayan noktalı işlem performansının daha iyi olduğunu söylemek yanlış olmayacaktır.



Şekil 5.29 Monte Carlo uygulaması için, Itanium server ve Sun iş istasyonları üzerinde, elde edilen aritmetik işlem süreleri (NT=1500, NS=128, NV=2)

İki farklı uygulama için elde edilen sonuçlar, performans değerlendirmenin pek çok uygulama ve mimari parametreye bağlı, karmaşık bir iş olduğunu fikrini destekler yöndedir ve hedef uygulama için en doğru platformun seçilmesinin, performans açısından, önemini vurgulamaktadır.

5.2.6 Haberleşme Performanslarının Değerlendirilmesi

Bu bölümde, seçilen uygulamaların haberleşme performanslarının analizinde kullanılmak üzere, Ethernet dışındaki daha hızlı ağ teknolojilerine ait modeller oluşturulmuştur. Paralel uygulamalar söz konusu olduğunda, uygulamanın özelliklerine de bağlı olarak, kullanılan ağ teknolojisi performans üzerinde son derece etkili olabilir. Bölüm 4.2'de sunulan sonuçlarda da gözlendiği gibi, özellikle FFT gibi işlemciler arası veri alışverişinin yoğun olduğu uygulamalarda, kullanılan haberleşme ağı büyük bir darboğaz yaratabilmekte ve hatta paralellik fayda sağlamaktan ziyade programın yürütülmesini yavaşlatabilmektedir.

Bu bölümde sunulan sonuçların elde edilmesinde kullanılan ağ teknolojilerinin teorik ve deneysel bant genişlikleri Çizelge 5.11'de gösterilmiştir. Bu ağlara ait PACE modelleri yaratılırken, hem PVM hem de MPI ileti geçirme kütüphanelerini kullanan modeller oluşturulmuştur.

Çizelge 5.10 Modellenen Ağ Teknolojileri için Bant Genişlikleri

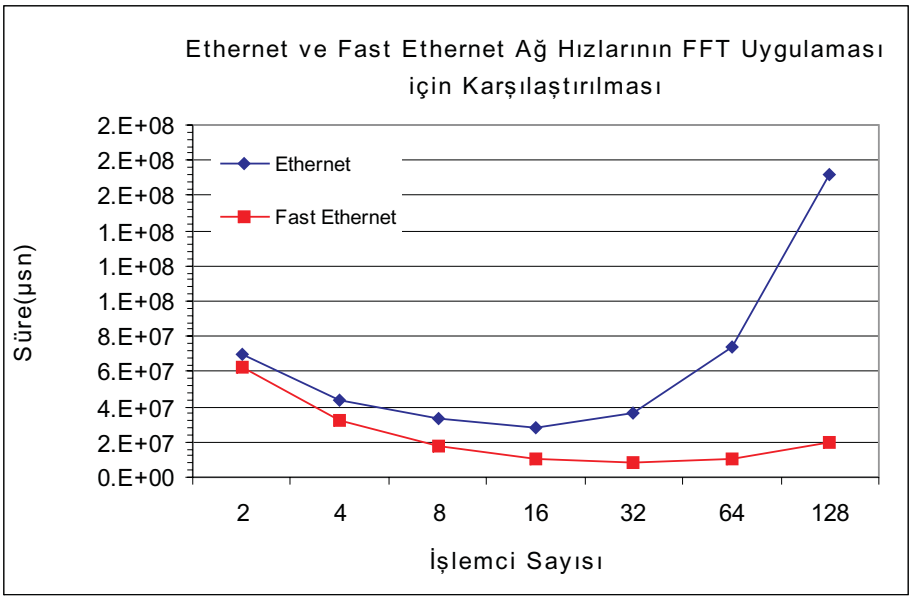
	Modellenen Haberleşme Ağları							
	Ethernet	**Fast Ethernet**	**FDDI**	**ATM 155**	**ATM 622**	**Gigabit Ethernet**	**HiPPI**	**Myrinet**
Teorik Bant Genişliği veya Veri Aktarma Oranı	10 Mbps	100 Mbps	100 Mbps	155 Mbps	622 Mbps	1000 Mbps	6.4 Gbps	1.2Gbps
Ulaşılabilen En Yüksek Bant Genişliği (Netperf Veritabanından alınmıştır)	9.21 Mbps	93.12 Mbps	95.8 Mbps	120.47 Mbps	535.42 Mbps	755 Mbps	4470.82 Mbps	735.89 Mbps

Deneylerde kullanılan ağ modellerinin oluşturulmasında, bu ağ teknolojilerinin elimizde bulunmayışı nedeni ile, bölüm 5.1.3'de bant genişliği ve ileti boyutunun fonksiyonu olarak verilen analitik denklemlerden yararlanılmıştır. Öncelikle belirli ileti boyutları için, haberleşme süreleri hesaplanmış ve daha sonra en küçük kareler yöntemi kullanılarak oluşturulan denklemler modellere ait HMCL dosyalarına aktarılmıştır.

Modeller oluşturulurken, sistemlerin teorik olarak erişebilecekleri en yüksek bant genişlikleri yerine, gerçeğe daha yakın olmaları için Netperf [11] karşılaştırmalı değerlendirilmelerinde gözlenen bant genişliği değerleri alınmıştır.

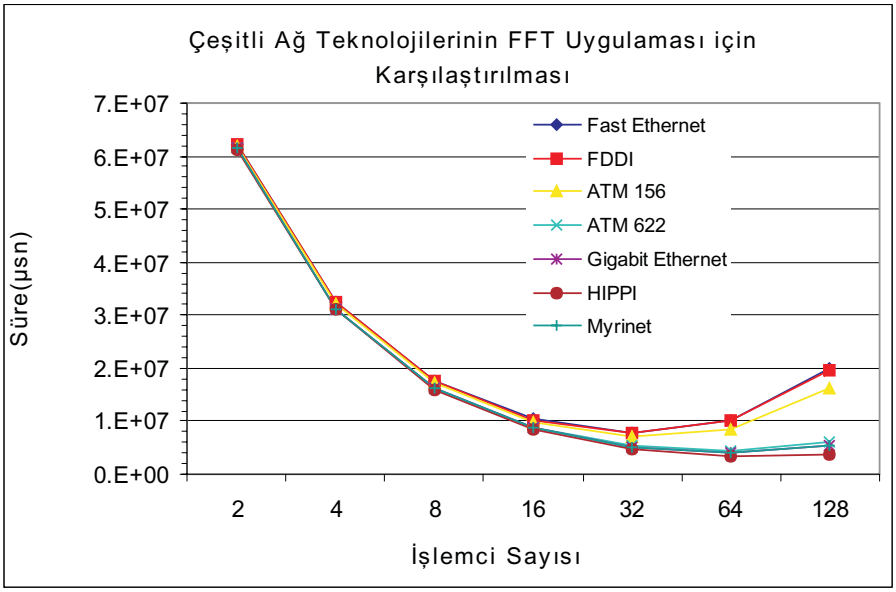
Çeşitli ağ teknolojilerine ait modeller oluşturulduktan sonra, yine FFT ve Monte Carlo uygulamaları kullanılarak, PACE üzerinde farklı konfigürasyonlar için ölçümler yapılmıştır.

Şekil 5.30, FFT uygulaması ile 10Mbps Ethernet ve 100Mbps Hızlı Ethernet ağları üzerinde elde dilen sonuçları karşılaştırmaktadır. Hızlı Ethernet kullanılması durumunda paralel FFT uygulamasının performansında önemli bir artış görülmüş ve 32 işlemciye kadar paralelleştirme beklenen şekilde uygulamanın çalışma hızını artırmıştır.



Şekil 5.30 Ethernet ve Fast Ethernet ağlar üzerinde Sun Blade 1750 iş istasyonları ile, FFT uygulamasına ait yürütüm süreleri

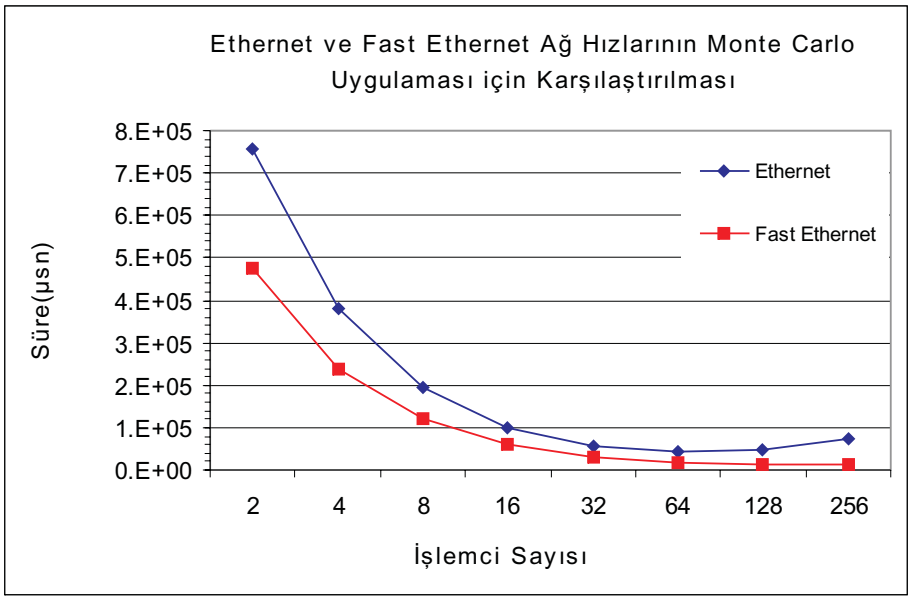
Şekil 5.31, diğer yüksek hızlı haberleşme ağları ile elde edilen sonuçları da içermektedir. Gigabit seviyesinde bant genişliği olan ağlar, çok başarılı sonuçlar vermekle birlikte, bunların kullanımı da ancak 16 veya 32 işlemciden sonra etkin olmaktadır.



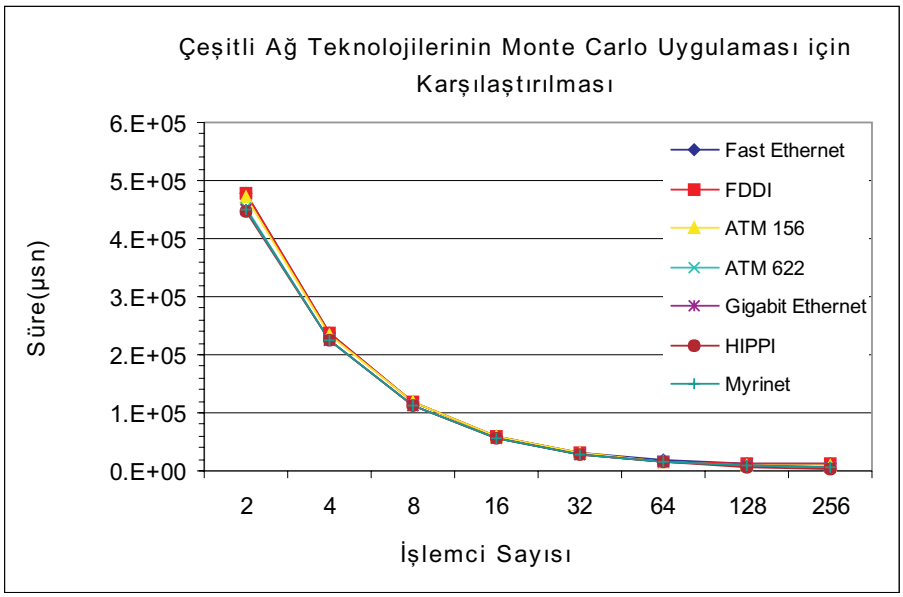
Şekil 5.31 Farklı haberleşme ağları üzerinde Sun Blade 1750 iş istasyonları ile, FFT uygulamasına ait yürütüm süreleri

Şekil 5.32 ve 5.33, Monte Carlo uygulamasının daha hızlı haberleşme ağları üzerindeki sonuçlarını göstermektedir. Bu sonuçlar, FFT uygulaması ile elde edilenlerden oldukça

farklıdır. Örneğin 10Mbps Ethernet yerine 100Mbps Hızlı Ethernet kullanılması, performansı bir miktar artırsa da, bütün olarak bakıldığında bu uygulama için karlı bir yatırım değildir. Monte Carlo uygulaması kayan noktalı işlem yoğunluğu olan ama sınırlı haberleşme gerektiren bir uygulama olduğu için, haberleşmeyi hızlandırarak elde edilebilecek performans iyileştirmesi sınırlıdır.



Şekil 5.32 Ethernet ve Fast Ethernet ağlar üzerinde Sun Blade 1750 iş istasyonları ile, Monte Carlo uygulamasına ait yürütüm süreleri



Şekil 5.33 Farklı haberleşme ağları üzerinde Sun Blade 1750 iş istasyonları ile, Monte Carlo uygulamasına ait yürütüm süreleri

6. PERFORMANS TAHMİNİ İÇİN ÖNERİLEN YSA MODELLERİ

Bu çalışmada, mimari ve program parametrelerinin performansa bireysel etkilerini tespit etmeye de olanak verecek şekilde, paralel sistemlerin performansının tahmini değerlendirilmesi için yapay sinir ağlarının nasıl kullanılabileceği incelenmiştir.

Paralel sistemlerin performans tahmini için, ileri beslemeli ve kısmen geri dönüşümlü topolojiye sahip iki ayrı model sunulmuştur.

Aritmetik işlem ve haberleşme performansları, aynı YSA mimarilerini farklı girdi parametreleri ile kullanılarak, birbirlerinden bağımsız olarak değerlendirilmiştir. Seçilen girdi parametreleri, kullanılan sistemlerin veri föylerinden ve uygulamaya ait kodlardan elde edilen temel özelliklerdir. Ancak, sunulan modeller esnek ve istenen derinlikte analiz olanağı sağlayan modellerdir. Modeller için farklı girdi parametreleri belirlenip, tekrar eğitilerek, kullanılabilirler. Her seferinde belirli bir girdi parametresi eklenerek, özellikle bu parametrenin temsil ettiği özelliğin, performans üzerine etkisi analiz edilebilir.

Aşağıda, kullanılan YSA modelleri, girdi ve çıktı olarak seçilen değişkenler ve kullanılan algoritmalar daha detaylı olarak tanımlanmıştır. Yapay sinir ağı modellerinin oluşturulması, eğitilmesi ve başarısının incelenmesi için MATLAB (Demuth ve Beale, 2001) kullanılmıştır.

6.1 İleri Beslemeli Çok Katmanlı Algılayıcı Modeli

Çok katmanlı ileri beslemeli yapay sinir ağları genellikle geriye yayma algoritması ile birlikte kullanılır. İleri beslemeli ağlar, sigmoid hücrelerden oluşan bir veya daha fazla gizli katman ve doğrusal hücrelerden oluşan bir çıkış katmanı içerirler. Doğrusal olmayan aktivasyon fonksiyonlarına yer veren çok katmanlı yapı, ağın, girdileri ile çıktıları arasındaki, doğrusal ve doğrusal olmayan ilişkileri öğrenmesine olanak verir.

Modelin son hali belirlenmeden önce, farklı sayıda nöron içeren bir ve birkaç gizli katmanlı modeller ile farklı aktivasyon fonksiyonlarının ve eğitim algoritmalarının kullanıldığı testler de yapılmıştır. Burada sunulan model, eldeki deney verileri için en başarılı sonuçların elde edildiği modeldir. Modeller oluşturulurken, yapay sinir ağının öğrenmesine olanak verecek miktarda nöron kullanılmaya ve farklı test verileri için aynı yüksek başarıyı elde etmeye özen gösterilmiştir.

Birden fazla gizli katman kullanılması sonuçlarda bir iyileştirme sağlamamıştır. Gizli katmanda çok fazla nöron kullanılması ise ezberlemeye neden olup, test verilerinin eğitim verilerinden farklı olması durumunda başarısız sonuç vermiştir. Benzer şekilde nöron sayısının çok az olması durumunda, tam olarak öğrenme gerçekleşmediği için, elde edilen

sonuçlar tutarsız ve başarısız olmuştur.

Bu çalışmada sunulan ilk yapay sinir ağı modeli ileri beslemeli, tek gizli katmanlı modeldir (Şekil 6.1). Hem aritmetik işlem hem de haberleşme performansı tahmini için ağın beş girdisi vardır ve gizli katmanda beş nöron bulunmaktadır.

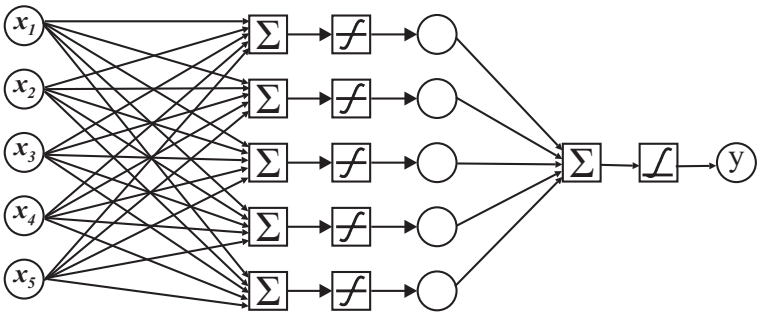
Gizli katmandaki nöronların aktivasyon fonksiyonları (6.1) ile tanımlanan hiperbolik tanjant (tansig) fonksiyonudur.

$$f(x) = \frac{2}{1-e^{-2x}} - 1 = \frac{e^{x} + e^{-x}}{e^{x} - e^{-x}} \tag{6.1}$$

Çıkıştaki aktivasyon fonksiyonu ise sigmoid olarak seçilmiştir (6.2).

$$f(x) = \frac{1}{1+e^{-x}} \tag{6.2}$$

Yapay sinir ağının eğitilmesinde, yaygın kullanılan eğri uydurma yöntemlerinden biri olan, Levenberg-Marquardt algoritması (Hagan ve Menhaj, 1994) kullanılmıştır.



Şekil 6. 1 Paralel sistemlerin aritmetik işlem ve haberleşme performanslarının tahmininde kullanılan ileri beslemeli YSA modeli

6.2 Geri Dönüşümlü Yapay Sinir Ağı Modeli

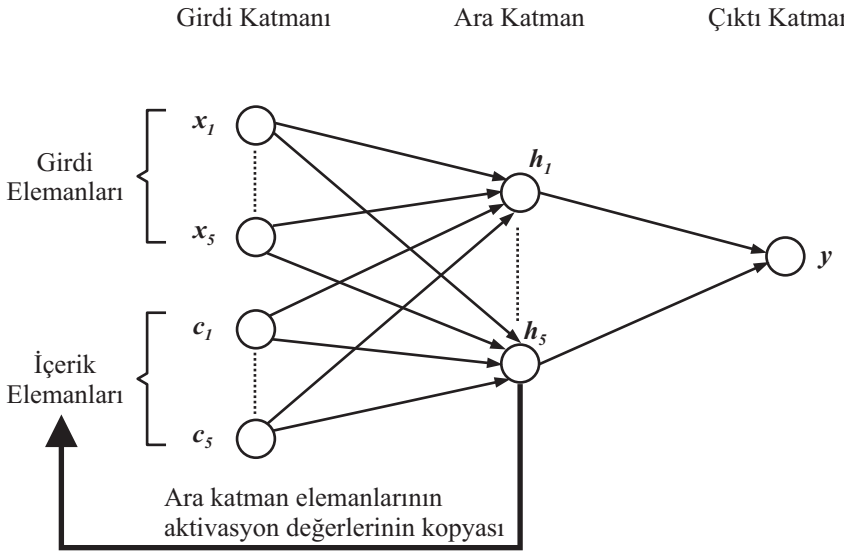
Çok katmanlı ileri beslemeli yapay sinir ağlarının aksine, geri dönüşümlü ağlarda, işlem elemanlarının çıktıları ağa belirli bir şekilde geri gönderilerek girdi olarak kullanılır.

Bu çalışmada sunulan ikinci yapay sinir ağı, basit geri dönüşümlü olarak da adlandırılan Elman ağıdır. Elman ağında, girdi, ara katman ve çıktı elemanlarının yanı sıra bir de içerik elemanları vardır. İçerik elemanları, ara katman elemanlarının bir önceki aktivasyon değerlerini hatırlamak için kullanılırlar. Ağın bir t zamanındaki durumu, hem o andaki girdilere, hem de (t-1) zamanındaki ara katman elemanlarının aktivasyon değerlerine bağlıdır. İleri doğru hesaplama yapıldıktan sonra oluşan ara katman elemanlarının aktivasyon değerleri, geriye doğru içerik elemanlarına gönderilir ve bir sonraki iterasyonda kullanılmak üzere

saklanır.

Bu çalışmada kullanılan Elman ağında, beş ara katman elemanı ve beş adet içerik elemanı bulunmaktadır. Sunulan ilk yapay sinir ağı modeli, ileri beslemeli, tek gizli katmanlı modeldir (Şekil 6.2).

Gizli katmandaki nöronların aktivasyon fonksiyonları hiperbolik tanjant (tansig) fonksiyonlarıdır. Çıktı katmanında kullanılan aktivasyon fonksiyonları ise doğrusaldır, sadece kendilerine gelen bilgileri toplarlar. Geri dönüşümlerin, içerik elemanlarının, bağlantı ağırlıkları sabittir ve değiştirilemez. Elman ağının kısmi geri dönüşümlü olarak adlandırılması da bu nedenledir (Öztemel, 2003).



Şekil 6. 2 Paralel sistemlerin aritmetik işlem ve haberleşme performanslarının tahmininde kullanılan geri dönüşümlü YSA modeli

Elman ağının eğitilmesinde, quasi-Newton algoritmasının BFGS (Broyden-Fletcher-Goldfarb-Shanno) sürümü kullanılmıştır (Dennis ve Schnabel, 1983).

Elman ağının eğitilmesinde, çok katmanlı ağın tersine, içerik elemanlarından yararlanabilmek için, eğitim verilerinin aralarındaki ilişkiyi temsil edecek şekilde sıralı olarak verilmesi anlamlı olacaktır.

6.3 YSA Modellerinin Girişleri

İyi bir model oluşturmanın en zor yanlarından biri, beklenen çıkışın oluşmasında rol alan faktörleri içeren girdi kümesinin seçilmesidir.

Paralel sistemlerin performansı birbirine içten bağımlı çok çeşitli donanım ve yazılım parametrelerinin bir fonksiyonudur.

Önerilen modeller için bir girdi kümesi oluşturulurken, modellerin evrensel olarak kullanılmasına olanak verecek, önemli temel donanımsal ölçütlerin yanı sıra, kullanıcının uygulamasını da temsil edecek parametrelerin bulunmasına dikkat edilmiştir. Bu modeller, amaca uygun olarak, daha genel girdi kümeleri kullanılarak veya daha detaylı analiz gereken durumlarda, sisteme özel girdiler eklenerek, yeniden eğitilerek kullanılabilirler.

Yukarıda anlatılan, ileri beslemeli ve geri dönüşümlü, iki topoloji için de aynı girdi kümeleri kullanılmıştır, elde edilen sonuçlar 7. Bölüm'de verilmiştir.

Kayan noktalı ve tamsayı işlemler ayrıntılı olarak sınıflanmış ve uygulamaların C kodları analiz edilerek kullanılan işlemci sayısının ve problem boyutunun fonksiyonu olarak her bir işlemin adedi belirlenmiştir. Bu analizlerde, GNU araçlarından, kapsam ve profil veri göstericileri gcov [8] ve gprof [9] kullanılmıştır. Bu analizle elde edilen formülasyonlar Ek1'de verilmiştir.

Bu çalışmada hesaplama performansının tahmininde kullanılan, yapay sinir ağı modellerinin girişleri şunlardır:

- İşlemci sayısı: Ağ üzerinde hedef uygulamayı çalıştıran işlemci -makine- sayısı
- Merkezi işlem ünitesi (CPU) hızı: Uygulamayı çalıştırmakta olan işlemcilerin hızı
- Problem boyutu: FFT uygulaması için problem boyutu resim ve bu resme uygulanan filtrenin boyutu olarak; Monte Carlo uygulaması için ise olasılıksal değişkenlerin ve denemelerin sayısı olarak verilmelidir.
- Tam sayı (integer) işlemlerin miktarı: Uygulamanın C kodundan problem boyutu ve işlemci sayısının fonksiyonu olarak ifade edilecek şekilde belirlenen, uygulamada yer alan tam sayı işlemlerinin miktarıdır.
- Kayan noktalı (floating point) işlemlerin miktarı: Uygulamanın C kodundan problem boyutu ve işlemci sayısının fonksiyonu olarak ifade edilecek şekilde belirlenen, uygulamada yer alan kayan noktalı sayılarla yapılan işlemlerinin miktarıdır.

Haberleşme performansının tahmininde kullanılan yapay sinir ağı modellerinin girişleri ise şunlardır:

- İşlemci sayısı
- Merkezi işlem ünitesi (CPU) hızı
- Problem boyutu
- MPI haberleşme çağrıları miktarı: Girişlerin bir fonksiyonu olarak ifade edilen ve dinamik olarak belirlenen MPI noktadan noktaya haberleşme çağrılarının miktarıdır.

- Ağ veri hızı: Mesaj ağa girdikten sonra bilginin yayılabileceği en yüksek hızdır.

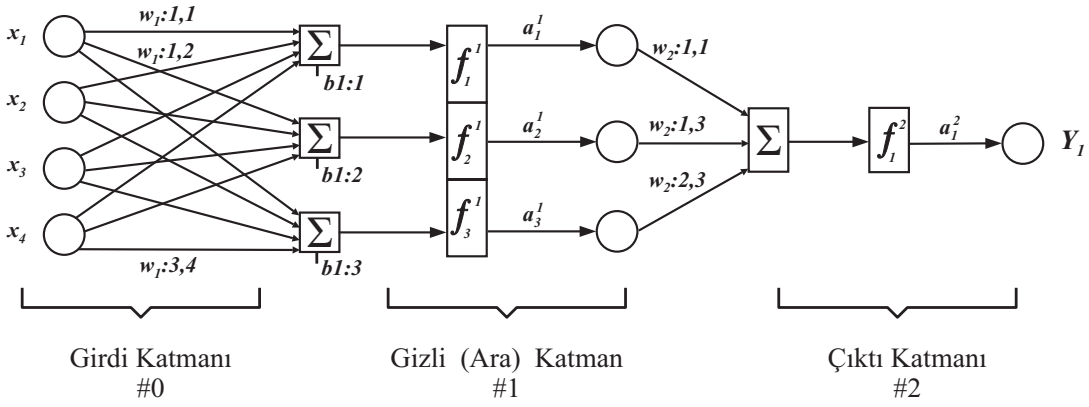
6.4 YSA Elemanları için Kullanılan Gösterimler

Yapay sinir ağlarında katmanların farklı şekilde numaralanması ve farklı gösterimler kullanılması söz konusudur (Fine, 1999). Bu alt bölümde, eğitim algoritmalarının tanımlanmasından önce kullanılan gösterimler kısaca tanıtılmıştır.

Şekil 6.3'de, x_1, x_2, x_3, x_4 olmak üzere dört gidisi bulunan, iki katmanlı ileri beslemeli bir yapay sinir ağı modeli gösterilmiştir. İlk katman $F_{1:1}, F_{1:2}, ve\, F_{1:3}$ ile gösterilen dört işlem elemanı içermektedir.

Her bir $\{F_{1:i}\}$ düğümünün çıktısı (6.3) eşitliği ile ifade edilir:

$$a_{1:i} = F_{1:i}\left(\sum_{j=1}^{4} w_{1:i,j} x_j - \tau_{1:i}\right) \ , \ (b_{1:i} = -\tau_{1:i}) \tag{6.3}$$



Şekil 6.3 İki katmanlı ileri beslemeli YSA mimarisi

Yapay sinir ağlarını oluşturan temel elemanlar için bu çalışmada kullanılan gösterimler aşağıda açıklanmıştır.

Ağın girdileri $a_{0:i}, i = 1:d$ veya $x_i, i = 1:d$ şeklinde gösterilir.

Karmaşıklık olabilecek durumlarda, ağın girdileri *d* x *n* boyutunda bir matris olarak da gösterilebilir, bu durumda aşağıdaki gösterimler kullanılmıştır:

$$\{\underset{-m}{x}, m = 1:n\}, \quad a^m_{0:i}, x^m_i, a^m_{i:j}, c^m_{i:j}$$

Ağın girdileri *d* x *n* boyutunda bir *S* matrisi olarak alındığında, bu matrisin *m*. sütununda yer alan girdi $\underset{-m}{x} = \{x_i^m\}$ şeklinde gösterilir.

Burada, girdiler 0 numaralı katmanda ve çıktılar *L* numaralı son katmanda yer almak üzere, ***i*** katman numarasını gösterir.

Katman numarası ilk alt simge olarak gösterilmiştir ve diğer alt simgelerden iki nokta üst üste (:) işareti ile ayrılmıştır.

i. katmandaki düğüm sayısı s_i şeklinde gösterilmiştir.

$F_{i,j}$, i numaralı katmandaki *j*. düğüme ait fonksiyondur. $\sigma_{i:j}$ veya σ şeklinde de gösterilebilir.

$\{F_{1:i}\}$ düğümünün çıktısı $a_{1:i}$ olup, (6.4) eşitliği ile ifade edilir:

$$a_{1:i} = F_{1:i}(\sum_{j=1}^{4} w_{1:i,j} x_j - \tau_{1:i}) \tag{6.4}$$

Yakınlık (bias) değeri *b* ile eşik değeri τ ile gösterilmiştir ve yakınlık değeri eşik değerinin negatifidir: $b_{i:j} = -\tau_{i:j}$.

Bağlantı ağırlıklarının gösteriminde, i nesnenin seviyesini ve *j,k* i-1 numaralı katmandaki k düğümü ile i numaralı katmandaki j düğümü arasındaki bağlantının ağırlığının kastedildiğini belirtmek üzere *i:j,k* şeklinde üç alt simgeli gösterim kullanılmıştır.

Analiz sırasında kolaylık sağlamak amacı ile tüm bu parametreleri temsil eden tek sütunlu bir $\underset{-}{w}$ ağırlıklar vektörü de kullanılmaktadır, bu durumda komşu katmanlar arasındaki tüm bağlantıların var olduğu ve bağlantıların olmaması halinde ise ağırlık değerinin sıfır olarak verildiği kabul edilmiştir.

6.5 Eğitim Algoritmaları

Eğitim algoritmaları, yapay sinir ağındaki bağlantıların ağırlıklarının ayarlanmasında kullanılan mekanizmalardır. Literatürde farklı yapay sinir ağı modelleri için önerilmiş olan çok sayıda algoritma vardır. Bu bölümde, sadece bu çalışmada kullanılan ve hata fonksiyonunun eğimini (gradyan) kullanan algoritmalar açıklanmıştır. Eğim, ağ üzerinde geriye doğru hesaplamalar yapan ve geri yayılım olarak adlandırılan teknik kullanılarak tespit edilmektedir (Hagan vd., 1996). Bu yöntemde, eğimin belirlenmesinden sonra, bağlantı ağırlıkları negatif eğim yönünde ayarlanır.

6.5.1 Hata Yüzeyi (Fonksiyonu) ve Newton Yöntemi

Yapay sinir ağının girişleri $\underset{-i}{x}$ ve arzu edilen çıkışları da $\underset{-i}{t}$ olmak üzere, bir

$T = \{(\underline{x}_i, \underline{t}_i), i = 1:n\}$ eğitim kümesi verildiğinde, ağın vereceği $\underline{y}_i = \eta(\underline{x}_i, \underline{w}_i)$ çıktısının arzu edilen $\underline{t}_i$ çıktısına en yakın olmasını sağlayacak η seçilmek istenir.

T eğitim kümesinin elde edilen çıktılara yakınlığı (6.5)'deki gibi bir hata fonksiyonu ile ifade edilir:

$$\varepsilon_T = \frac{1}{2} \sum_{i=1}^{n} ||\underline{y}_i - \underline{t}_i||^2 \tag{6.5}$$

Bu ε_T hata fonksiyonu, $\underline{w}$ ağırlıklarının bir fonksiyonudur. Bu fonksiyonun, $\underline{w}$ 'ye göre, birinci türevi alınarak elde edilen eğim vektörü ise (6.6) eşitliği ile ifade edilir:

$$\underline{g}(\underline{w}) = \nabla^2 \varepsilon_T \,|_{\underline{w}} = [\, \frac{\partial \varepsilon_T}{\partial \underline{w}_i} \,] \,|_{\underline{w}} \tag{6.6}$$

İkinci türev alınarak simetrik Hessian matrisi (6.7) ve (6.8)'deki gibi elde edilir:

$$H(\underline{w}) = [\, H_{ij}(\underline{w}) \,] = \nabla^2 \varepsilon_T(\underline{w}) \tag{6.7}$$

$$H_{ij} = \frac{\partial \varepsilon_T(w)}{\partial w_i \partial w_{ij}} = H_{ji} \tag{6.8}$$

$o(\delta)$, $\underset{\delta \to 0}{Lim} \frac{o(\delta)}{\delta} = 0$ koşulunu sağlayan, sıfır mertebesinde bir terimi belirtmek üzere, ε_T hatasının muhtemel yerel minimuma ulaştığı bir noktada kesilen Taylor serisi açılımı (6.9)'daki gibi yazılabilir:

$$\varepsilon_T(\underline{w}) = \varepsilon_T(\underline{w}^0) + \underline{g}(\underline{w}^0)^T(\underline{w} - \underline{w}^0) + \frac{1}{2}(\underline{w} - \underline{w}^0)^T H(\underline{w}^0)(\underline{w} - \underline{w}^0) + o\,\|\underline{w} - \underline{w}^0\|^2 \tag{6.9}$$

Burada açıklanan algoritmalar aşağıda (6.10) ile ifade edilen yaklaşık karesel modele dayanır.

$$m(\underline{w}) = \varepsilon_T(\underline{w}^0) + \underline{g}(\underline{w}^0)^T(\underline{w} - \underline{w}^0)\, \frac{1}{2}(\underline{w} - \underline{w}^0)^T \tag{6.10}$$

(6.10) ile ifade edilen model, ancak ve ancak *H* Hessian matrisi kesin pozitif bir matris ise, tek bir minimuma sahiptir. Bu ifadenin ispatı (Reed ve Marks, 1999) tarafından verilmiştir.

(6.10) ile verilen karesel modelin eğimi (6.11)'deki gibi yazılabilir.

$$\nabla m = \underline{g}(\underline{w}^0) + H(\underline{w} - \underline{w}^0) \tag{6.11}$$

Bu eğimi sıfıra eşitleyip, en küçük hale getiren $\underline{w}^*$ değeri için çözersek (6.12) eşitliği elde edilir.

$$\underline{w}^* = \underline{w}^0 - H^{-1}\underline{g} \tag{6.12}$$

Şimdi (6.10) ile verilen karesel model, bu minimum $\underline{w}^*$ cinsinden (6.13)'de ki gibi yeniden yazılabilir.

$$m(\underline{w}) = m(\underline{w}^*) + \frac{1}{2}(\underline{w} - \underline{w}^*)^T H(\underline{w}^*)(\underline{w} - \underline{w}^*) \tag{6.13}$$

Karesel modelde, ağırlık vektörünün herhangi bir başlangıç değerinden başlayarak, eğer mevcut ise en küçük hale getiren $\underline{w}^*$ değerine ulaşılabilir. Bu yöntem Newton metodu olarak bilinir. Newton metodu, *H* Hessian matrisi kesin pozitif bir matris olmak şartı ile karesel olmayan durumlarda da kullanılabilir.

6.5.2 Özyineli (iteratif) Düşüm Algoritmaları

Geriye yayma algoritması, en genel anlamı ile, makul bir *T* çalışma süresinde iteratif olarak $\{\underline{w}_k, k = 1:T\}$ parametre vektörlerini seçen, bir düşüm algoritmasıdır. Algoritmanın amacı, genellikle bulunması zor olan evrensel minimuma ulaşmaktan ziyade, iyi bir yerel minimum komşuluğuna yakınsamaktır[*].

Temelde, tekrarlayıcı özyineleme işlemi, o anki $\underline{w}_k$ vektörü cinsinden yeni bir $\underline{w}_{k+1}$ parametre vektörü belirler. Bunun için öncelikle bir $\underline{d}_k$ arama yönü ve bir α_k öğrenme katsayısı seçilmelidir (6.14).

$$\underline{w}_{k+1} = \underline{w}_k + \alpha_k \underline{d}_k \tag{6.14}$$

p-boyutlu bir uzayda, minimum aranırken, sırayla her bir $\{\underline{d}_k\}$ yönü taranır. Bir $\underline{d}_k$ yönünde arama yapılırken, gidilecek mesafe ise öğrenme katsayısı α_k ile belirlenir. Düşüm algoritmaları Markov özelliğine sahiptir; bir sonraki durum o duruma varana kadarki olan tüm durumlarına değil, sadece bir önceki durumuna bağlı olarak tanımlanabilir. Yani *k*.

---

[*] Tüm bağlantı ağırlıkları ve yanlılık değerleri, $\underline{w}$ vektörü ile gösterilmiştir.

özyinelemeden sonraki durum sadece $\underset{-k}{w}$, $\underset{-k}{d}$ ve $\underset{-k}{g}$ değerlerini içerebilir.

Temel en dik düşüm algoritmasında, *k*. özyinelemeden sonraki durum sadece $\underset{-k}{w}$ parametresine ve $\underset{-k}{g}$ eğimine bağlıdır. Moment kullanılan en dik düşüm algoritmasında, durum o andaki parametre değeri ve eğim ile bir önceki parametre değerine bağlıdır.

Bu çalışmada kullanılan algoritmalar, hata yüzeyine yerel olarak bakarak, bir sonraki arama noktasını tespit eder. Arama yönünü belirlerken, hata fonksiyonunun ardı ardına gelen değerleri için (6.15) ile verilen birinci dereceden yaklaşık değer dikkate alınır:

$$\mathcal{E}(k+1)-\mathcal{E}(k) \approx \underset{-}{g}(\underset{-k}{w})^T(\underset{-k+1}{w}-\underset{-k}{w}) \tag{6.15}$$

Özyineli algoritmanın istikralı bir düşüme ulaşabilmesi için, her aşamada hatanın azaltılması gerekir. Çok büyük olmayan $(\underset{-k+1}{w}-\underset{-k}{w})$ artımları için, aşağıda (6.16) ile verilen düşüm koşulu sağlanmalıdır.

$$\underset{-}{g}(\underset{-k}{w})^T(\underset{-k+1}{w}-\underset{-k}{w}) = \underset{-}{g}(\underset{-k}{w})^T(\alpha_k \underset{-k}{d}) = \alpha_k \underset{-k}{g}^T \underset{-k}{d} < 0 \tag{6.16}$$

Bu düşüm koşulu en basit şekilde (6.17) ile sağlanır.

$$\alpha_k > 0 \ , \ \underset{-k}{d} = -\underset{-k}{g} \tag{6.17}$$

Bu şekilde belirli bir düşüm yönü seçilmesi en dik düşüm algoritmalarının temelini oluşturur. Burada, arama yönü, hata yüzeyinin en dik düşümü yönünde seçilmiştir.

Daha genel olarak ifade edildiğinde, eğer $\{\mathrm{M}_k\}$ kesin pozitif matrisler zinciri ise $\underset{-k}{d} = -M_k \underset{-k}{g}$ düşüm koşulunu sağlar denilebilir.

Newton yöntemi M_k için, $M_k = H^{-1}(\underset{-k}{w})$ seçimini yapar. Quasi-Newton ve Levenberg-Marquardt yöntemleri ise M_k seçimini yaparken Hessian *H* matrisi için kesin pozitif yaklaşıklar kullanırlar.

6.5.3 Eğimin Belirlenmesi - Geriye Yayma Algoritması

Quasi-Newton ve Levenberg-Marquardt algoritmalarının gerçeklenebilmesi için, eğim vektörünün belirlenmesi gereklidir. Geriye yayma algoritması, eğim vektörünün belirlenmesinde, etkin bir yöntem sağlar.

Geriye yayma algoritmasının uygulanması iki aşamadan oluşur. İlk aşamada, girdi ağa

gösterilir ve her birimin çıkışını hesaplamak üzere ağ boyunca ileriye yayılır, ardından bu çıkış değerleri, hedeflenen değerler ile kıyaslanarak bir δ hata değeri belirlenir. İkinci aşamada, δ hata değeri geriye doğru ağdaki tüm birimlere yayılır ve uygun ağırlık değişimleri yapılır. Geriye yayım, δ hata değerinin özyineli olarak hesaplanmasına olanak verir (Rumelhart ve Mcclelland, 1986; Shavlik ve Dietterich, 1990).

Geriye yayma algoritması kullanılarak, eğim vektörünün belirlenmesi aşağıdaki gibi özetlenebilir:

- Eğitim verileri, düğümlerin $a_{i:j}^m$ çıkışlarını ve $c_{i:j}^m$ girişlerini belirlemek üzere ağ boyunca ileriye yayılır. Temel yapay sinir ağı denklemlerinden yola çıkarak (6.18), (6.19) ve (6.20) eşitlikleri yazılabilir:

$$a_{0:j}^m = (\underset{\_}{x}_m)_j = x_j^m \tag{6.18}$$

$$c_{i:j}^m = \sum_k^{S_{i-1}} w_{i:j,k} a_{i-1:k} + b_{i:j} \quad , \quad i > 0 \tag{6.19}$$

$$a_{i:j}^m = F_{i:j}(c_{i:j}^m) \quad , \quad i > 0 \tag{6.20}$$

- Ağ üzerinde geriye yayma yöntemi ile, (6.21) ve (6.22) eşitlikleri kullanılarak, $\delta_{i:j}^m$ değerleri belirlenir.

$$\delta_{i:j}^m = f_{i:j}(c_{i:j}^m) \sum_{k=1}^{S_{i+1}} \delta_{i+1:k}^m w_{i+1:k,j}^m \tag{6.21}$$

$$\delta_{L:1}^m = f_{L:1}(c_{L:1}^m)(a_{L:1}^m - t_m) \tag{6.22}$$

- Elde edilen sonuçlar, (6.23) ve (6.24) eşitlikleri kullanılarak, birleştirilir ve eğim belirlenir.

$$\frac{\partial \mathcal{E}_m}{\partial w_{i:j,k}} = \delta_{i:j}^m a_{i-1:k}^m \tag{6.23}$$

$$\frac{\partial \mathcal{E}_m}{\partial b_{i:j}} = \delta_{i:j}^m \tag{6.24}$$

6.5.4 Quasi-Newton Algoritmaları

Eğer hata fonksiyonu tam olarak karesel ise, hatayı en küçük hale getiren $\underset{\_}{w}^*$ değeri, Newton metodu kullanılarak, (6.25) eşitliği ile tek adımda çözülebilir:

$$\underset{\_}{w}^* = \underset{\_}{w}^0 - H^{-1} \underset{\_}{g} \tag{6.25}$$

Ancak bu çözüm Hessian matrisinin bilinmesini ve kesin pozitif olmasını gerektirir.

quasi-Newton yöntemleri, özyineli algoritmayı öncelikle (6.26)'da verilen şekilde genelleştirir.

$$\underset{-k+1}{w} = \underset{-k}{w} - \alpha_k M_k \underset{-k}{g} \tag{6.26}$$

Arama yönü $d_k = M_k \underset{-k}{g}$ ile kullanılacak, öğrenme katsayısı (adım miktarı) α_k olan, yaklaşık bir yön araması ile belirlenir.

$\underset{-k}{w}$, $\mathcal{E}$ hatasının $\underset{-}{w}^*$ yerel minimumuna yakın ise, (6.27)'ye göre,

$$\alpha_k \approx 1, M_k \approx H^{-1}(\underset{-}{w}^*) \tag{6.27}$$

olması beklenir. Eğer M_k kesin pozitif bir matris olacak şekilde seçilirse düşüm koşulu sağlanmış olur. Düşüm koşulunu sağlayan matrislere örnek olarak M_k şu şekilde seçilebilir:

$$M_k = \varepsilon_k I + H(\underset{-k}{w}) \tag{6.28}$$

Eğer $\varepsilon_k = 0$ ise (6.28) eşitliği Newton formülüne dönüşür, eğer ε_k daha büyük ise bu yaklaşık en dik düşümdür (Fine, 1999).

H_k matrisinin kesin pozitif olmaması durumunda, Hessian matrisinin (λ_i) özdeğerleri için $\varepsilon_k > \max_i(-\lambda_i)$ olacak şekilde bir ε_k değeri seçilerek M_k matrisinin kesin pozitif olması sağlanabilir. Bu durumda kesin pozitif olmayan Hessian matrisi kullanılabilir, ama yine de Hessian matrisinin belirlenmesi gerekmektedir.

Quasi-Newton yöntemleri Hessian matrisinin tersinin, direk hesaplama yapmaya gerek kalmadan, özyineli olarak belirlenmesine olanak verir.

Eğer $q_k = \underset{-k+1}{g} - \underset{-k}{g}$ ise, eğim şu şekilde ifade edilebilir:

$$q_k = H_k(\underset{-k+1}{w} - \underset{-k}{w}) = H_k p_k \tag{6.29}$$

Eğer $\underset{-0}{p}, ..., \underset{-p-1}{p}$ gibi p adet doğrusal bağımsız artım için eğim farkları belirlenebilirse Hessian matrisi için çözüm yapılabilir.

$P = \begin{bmatrix} p_0 \\ p_1 \\ : \\ p_{i-1} \end{bmatrix}$ ve $Q = \begin{bmatrix} q_0 \\ q_1 \\ : \\ q_{i-1} \end{bmatrix}$ şeklinde iki matris oluşturulursa, $Q = HP$ matris denklemi yazılabilir ve Hessian matrisi $H = QP^{-1}$ şeklinde çözülebilir.

Hessian matrisinin tersi olan M matrisi de $\underset{-k}{q}$ ve $\underset{-k}{p}$ yer değiştirilerek $M = PQ^{-1}$ şeklinde ifade edilebilir.

Quasi-Newton algoritmasının çalışması aşağıdaki gibi özetlenebilir:

- Hessian matrisinin $\underset{-k}{w}$ için, yaklaşık tersi olan kesin pozitif M_k matrisi oluşturulur,
- $d_k = M_k \underset{-k}{g}$ şeklinde bir arama yönü seçilir,
- $\underset{-k+1}{w} = \underset{-k}{w} - \alpha_k d_k$ özyinelemesinde, α_k öğrenme katsayısını belirlemek için yaklaşık yön arama işlemi gerçekleştirilir,
- $\underset{-k+1}{w}$ için yaklaşık ters Hessian matrisi M_{k+1} tekrar hesaplanır ve yukarıdaki işlemler tekrarlanır.

6.5.5 Levenberg-Marquardt Algoritması

Levenberg-Marquart gerçekleştirimi pek çok doğrusal olmayan probleme başarı ile uygulanmıştır ve Gauss-Newton yönteminden daha başarılı olduğu ispatlanmıştır. Bu algoritma, iteratif olarak da kısıtlanmamış metotlardan daha etkindir.

Levenberg-Marquardt Algoritması, hata fonksiyonunun (6.30) eşitliğinde verildiği gibi kareler toplamı olması durumunu kullanır:

$$\varepsilon_T = \frac{1}{2}\sum_{i=1}^{n}(\underset{-i}{y} - \underset{-i}{t})^2 = \frac{1}{2}\sum_{i=1}^{n} e_i^2 \tag{6.30}$$

Hata vektörü $\underset{-}{e} = [e_i]$ şeklinde gösterilirse, hata vektörünün, ağ parametreleri $\underset{-}{w}$'ye göre, $J = [J_{ij}]$ Jacobian matrisi (6.31) eşitliğindeki gibi ifade edilir:

$$J_{ij} = \frac{\partial e_j}{\partial w_i}\ , \quad \begin{cases} i = 1:p \\ j = 1:n \end{cases} \tag{6.31}$$

J Jacobian matrisi p x n boyutunda büyük bir matristir ve bu matrisin tüm elemanları geriye yayma yöntemi ile hesaplanır.

Karesel hata fonksiyonu için, p-boyutlu $\underline{g}$ eğim vektörü (6.32) eşitliği ile ifade edilir.

$$\underline{g}(\underline{w}) = \sum_{i=1}^{n} e_i \nabla e_i(\underline{w}) = J\,\underline{e} \tag{6.32}$$

$H = [\,H_{ij}\,]$ Hessian matrisi ise (6.33) eşitliği ile bulunabilir:

$$H_{ij} = \frac{\partial^2 \boldsymbol{\mathcal{E}}_T}{\partial w_i \partial w_j} = \frac{1}{2}\sum_{k=1}^{n} \frac{\partial^2 e_k^2}{\partial w_i \partial w_j} = \sum_{k=1}^{n}\left[e_k \frac{\partial^2 e_k}{\partial w_i \partial w_j} + \frac{\partial e_k}{\partial w_i}\frac{\partial e_k}{\partial w_j} \right] = \sum_{k=1}^{n}\left[e_k \frac{\partial^2 e_k}{\partial w_i \partial w_j} + J_{ik} J_{jk} \right] \tag{6.33}$$

Eğer $D = \sum_{i=1}^{n} e_i \nabla^2 e_i$ şeklinde bir tanımlama yapılırsa, Hessian matrisi $H(\underline{w}) = JJ^T + D$ şeklinde yazılabilir.

Hessian matrisi için yazılan bu ifade, ikinci türevlerden oluşan D matrisinin yerine pozitif olarak ölçeklenmiş bir εI birim matrisi kullanılarak basitleştirilirse Levenberg-Marquard algoritmasının kullandığı (6.34) ve (6.35) eşitlikleri elde edilir.

$$M_k = [\,JJ^T + \varepsilon I\,]^{-1} \tag{6.34}$$

$$\underline{w}_{k+1} = \underline{w}_k - \alpha_k M_k\, \underline{g}(\underline{w}_k) \tag{6.35}$$

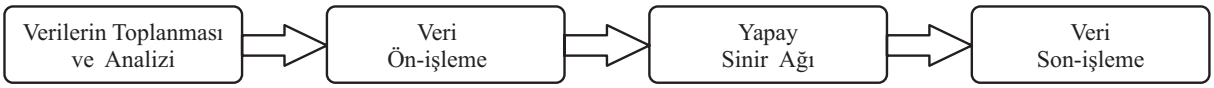
Bu algoritmanın başarılı kullanımı için, yaklaşık yön bulma yöntemi ile α_k öğrenme katsayısının tespit edilmesi gereklidir.

6.6 Verilerin Analizi ve İşlenmesi

Bu bölümde, elde edilen verilerden önerilen yapay sinir ağı modellerinin eğitilmesi, doğrulanması ve testi için kullanılan veri kümelerinin nasıl oluşturulduğu açıklanmıştır.

Tahmin edilecek verilerin kalitesi, yapay sinir ağının doğru bir şekilde eğitilebilmesi ve tahmin işinin başarısı açısından önemlidir. Veriler önemli bilgileri doğru yansıtacak şekilde işlenmezse iyi oluşturulmuş bir yapay sinir ağı modeli dahi başarılı olamayabilir.

Verilerin işlenmesi süreci verilerin toplanması ve analizi ile başlar, ardından veriler, gerekiyorsa, bir ön-işleme tabi tutulur ve sinir ağına gösterilir. Son olarak da verileri istenen çıktılara dönüştürmek için son-işleme aşaması uygulanır (Şekil 6.4).



Şekil 6.4 Veri işleme süreci

6.6.1 Ön Veri Analizi

Çizelge 6.1 ve 6.2 önceki bölümlerde açıklandığı gibi deneysel yollarla ve PACE üzerinde oluşturulan modeller yardımı ile elde edilen ham verilerden bir kesit göstermektedir. Deneysel yollarla ve PACE üzerinde oluşturulan modeller yardımı ile, elde edilen toplam veri adedi 5000'in üzerindedir.

Bu veriler, yapay sinir ağlarına gösterilmeden önce alt kümelere ayrılmış, bu alt kümelerdeki veriler de kendi aralarında eğitim, doğrulama ve test verileri olarak gruplanmıştır. Özellikle eğitim verisi olarak kullanılacak verilerin problem uzayını tam olarak temsil edebilecek nitelikte olması modellerin başarısı için gereklidir.

Verilere yapay sinir ağı modelleri ile kullanılmadan önce hangi ön işlemler uygulandığı ve yapay sinir ağının çıktılarına uygulanan ardıl işlemler aşağıda açıklanmıştır.

Çizelge 6.1 Haberleşme performansının tahmini için kullanılacak olan ham verilerden bir kesit

		Problem Boyutu				
İşlemci Sayısı	CPU Hızı (MHz)	Filtre Boyutu	Görüntü Boyutu	MPI Çağrı Sayısı	Haberleşme Ağı Bant Genişliği (Mbps)	Haberleşme için Harcanan Süre (µsn)
4	170	13	64	41	9.21	5632197.889
8	170	13	64	93	9.21	7689303.679
16	170	13	64	197	9.21	9974381.575
256	170	13	64	3317	93.12	4618407..35
---	---	---	---	---	---	---
---	---	---	---	---	---	---
64	270	55	1024	821	9.21	1386924628
128	270	55	1024	1653	9.21	3185247491
2	143	25	64	15	93.12	646500.921
4	143	25	64	41	93.12	8561153.91
---	---	---	---	---	---	---
---	---	---	---	---	---	---

Çizelge 6.2 Aritmetik işlem performansının tahmini için kullanılacak olan ham verilerden bir kesit

		Problem Boyutu				
İşlemci Sayısı	CPU Hızı (MHz)	Filtre Boyutu	Görüntü Boyutu	Tamsayı işlem komutu miktarı	Kayan Noktalı işlem komutu miktarı	Aritmetik İşlemler için Harcanan Süre (µsn)
16	270	13	64	184,550	2,938	9.67E+03
32	270	13	64	109,808	1,475	6.02E+03
64	270	13	64	72,437	744	4.19E+03
---	---	---	---	---	---	---
---	---	---	---	---	---	---
64	143	25	64	75,209	744	5.24E+03
128	143	25	64	56,524	378	3.60E+03
256	270	13	64	44,409	195	2.82E+03
2	300	13	64	123,938	23,420	52268.7
4	300	13	64	633,002	11,716	26588.68
8	300	13	64	334,034	5,864	13748.67
16	300	13	64	184,550	2,938	7328.665
32	300	13	64	109,808	1,475	4118.662
---	---	---	---	---	---	---
---	---	---	---	---	---	---

6.6.2 Veri Kalitesi

Tahmin işleminin mümkün olduğundan ve yapay sinir ağının gerektiği gibi eğitilebileceğinden emin olmak için öncelikle kullanılacak verilerin kalitesinin kontrol edilmesi gerekir. Verilerin kalitesi ile kastedilen aykırı değerler içermemesi ve rasgele olmamasıdır.

Verilerin kalan kısmından son derece farklı veya uyumsuz olan örnekler "*aykırı değerler"* olarak adlandırılır (Han ve Kamber, 2001). Aykırı değerleri tespit etmek için veri dağılımının 75. yüzdelik ve 25. yüzdelik değerleri kullanılır.

Veri kümesini, her alt küme verilerin %25 ve %75'ini içerecek şekilde, ikiye ayıran değerler

Q_1 (alt çeyrek) = 25. yüzdelik,

Q_2 (medyan) = 50. yüzdelik,

Q_3 (üst çeyrek) = 75. yüzdelik olarak adlandırılır.

Veri kümesinde $Q_3 + 3(Q_3 - Q_1)$ değerinden büyük veya $Q_1 - 3(Q_3 - Q_1)$ değerinden küçük olan değerler "*uç aykırı değerler"* olarak adlandırılır. Bu çalışmada kullanılan veriler herhangi bir aykırı değer içermemeleri için kontrol edilmiştir.

Rasgelelik geçmiş örneklere bakarak, gelecek olayların tahmininin mümkün olmaması anlamına gelir. Literatürde bir dizinin rasgele olup olmadığını kontrol etmek için kullanılan çeşitli testler vardır. Bu testler deneysel ve teorik olmak üzere iki ana gruba ayrılır. Bu çalışmada veri kümelerinin rasgelelik kontrolünde 'Run' testi kullanılmıştır (Crawford, 2000; Knuth, 1981).

Yapay sinir ağlarında kullanılacak veriler üzerinde bu testin uygulanması için önce veri kümesinin ortanca (medyan) değeri hesaplanır ve ardından bu medyan değerinin üzerinde kalan örnekler '+', altında kalanlar ise '-' sembolleri atanarak belirlenir. Bu işlem sonucunda '+' ve '-' sembollerinden oluşan yeni bir S serisi elde edilir. Eğer seri başlangıçta tek sayıda eleman içeriyorsa, yeni S dizisi oluşturulurken medyan değeri atılır. Bu seri her dönüm noktasında dikey bir çizgi ile ayrılırsa '+' ve '-' sembollerinden oluşan gruplar elde edilir.

Örneğin, + + + - + + - - - - + + - - şeklindeki bir seride tekrarlayan kısımlar gruplanarak

her dönüm noktasında dikey bir çizgi ile ayrılır ; + + + | - | + + | - - - - | + + | - -. Bu durumda, '+' sembollerini içeren üç grup ve '-' sembollerini içeren üç grup elde edilmiş olur.

S dizisi m adet '+' ve m adet '-' sembolünden oluşur ve bu dizinin boyutu 2m olacaktır. Ayrıca, '+' sembollerini içeren grupların sayısı r_+ ve '-' sembollerini içeren grupların sayısı r_- olarak tanımlanırsa, toplam grup sayısı $r = r_+ + r_-$ olacaktır. Eğer, grup sayısı *r* tek sayı ise, $r_+ = r_- + 1$ veya $r_- = r_+ + 1$ olmalıdır. Genellikle *r* değerinin çok küçük veya çok büyük olması dizinin rasgele olduğuna işaret eder. Rasgele bir S dizisi için yaklaşık normal bir dağılıma sahip *r* için ortalama değerinin (6.36),

$$E(r) = m + 1 \tag{6.36}$$

ve varyans değerinin (6.37)'deki gibi,

$$\text{var}(r) = \frac{m(m+1)}{2m-1} \tag{6.37}$$

olduğu kanıtlanmıştır.

Bu çalışmada sunulan modellerin eğitiminde kullanılan veri kümelerinin rasgele olmadıkları doğrulanmıştır.

6.6.3 Ön-İşleme

Teorik olarak yapay sinir ağları ham giriş verisini herhangi bir çıkış verisine eşleyebilir. Ancak uygulamada verilere bazı ön işlemler uygulanması faydalı ve çoğu zaman da

gereklidir. Ön işlem olarak nitelendirilebilecek işlemler arasında zaman serilerine uygulanan basit filtreleme işleminden görüntü verilerinden özellik çıkarma gibi daha karmaşık işlemlere kadar pek çok işlem sayılabilir. Uygulanacak ön işlemler uygulamanın ve verilerin özelliğine göre farklılık ve çeşitlilik gösterir.

Verilerin içerdiği bilgi ile ilgili verilerin seçilmesi ve gürültü niteliğindeki verilerin ayıklanması da önemlidir. Verileri yapay sinir ağının girişlerine ve kullanılan aktivasyon fonksiyonlarına uygun hale dönüştürmek ağın çalışmasını hızlandıracaktır. Bu dönüşümler verilere bir fonksiyon uygulama veya ölçekleme yolu ile yapılabilir.

Bu çalışmada veriler, çıktı katmanında kullanılan sigmoid fonksiyonu ve verilerin sayısal değerlerinin birbirlerinden çok farklı aralıklarda yer aldıkları için ondalık ölçekleme yöntemi ile [0,1] aralığına sıkıştırılmıştır.

Ondalık ölçekleme yöntemi kullanıldığında, k elemankı bir seride yer alan s elemanının, ölçeklenmiş s' değeri şu şekilde ifade edilir:

$$s' = \frac{s}{10^j}, \quad j = \log(\max(s_k)) \tag{6.38}$$

6.6.4 Ardıl İşlemler

Ardıl işlemler, ağın çıktılarına uygulanan tüm işlemleri kapsar. Ön işlemede olduğu gibi, ardıl işlemler de tümüyle uygulamaya bağlıdır. Ardıl işlemlere örnek olarak, bir parametrenin kabul edilebilir sınırları aştığının tespiti veya ağın çıktısının kural-tabanlı bir işlemciye girdi olarak verilmesi sayılabilir.

Çoğunlukla, çıkışta verilere uygulanacak işlemler ön işlemenin tam tersidir. Bu çalışmada da ağın çıktıları, ondalık ölçekleme işleminin tersi uygulanarak, tekrar orijinal değer aralıklarına dönüştürülmüştür

7. SİMÜLASYON SONUÇLARININ DEĞERLENDİRİLMESİ

Bu bölümde, Matlab- Neural Network Toolbox paket programı yardımıyla bilgisayar ortamında çalıştırılan modellerin simülasyon sonuçları gösterilmiştir. Ayrıca, paralel sistemlerin aritmetik işlem ve haberleşme performanslarının tahmininde kullanılmak üzere tasarlanan modeller için, simülasyonlar neticesinde elde edilen sonuçlar değerlendirilmiştir.

Sunulan yapay sinir ağı modellerinin, eğitim, doğrulama ve test başarıları, ortalama karesel hata (mean squared error - mse) fonksiyonu (7.1) kullanılarak değerlendirilmiştir.

$$okh = \frac{1}{Q}\sum_{k=1}^{Q} e(k)^2 = \frac{1}{Q}\sum_{k=1}^{Q} (t(k) - a(k))^2 \tag{7.1}$$

Burada *t(k)* ve *a(k)*, sırasıyla, *k*. hedef değeri ve yapay sinir ağının tahmin ettiği çıkış değerini göstermektedir. *Q* ise veri kümesindeki eleman sayısıdır. Bağlantıların ağırlıkları bu hatayı en küçük hale getirecek şekilde ayarlanmaktadır.

Ayrıca, her veri kümesi için ortalama yüzde hata ve standart sapma hesapları da (7.2) ve (7.3) ile ifade edildikleri şekilde yapılmıştır.

$$Ortalama\ Hata\ (\%) = \frac{1}{Q}\sum_{i=1}^{Q} \frac{\left|ölçülen\ değer_i - tahmin_i\right|}{ölçülen\ değer_i} x100 \tag{7.2}$$

$$Standart\ Sapma = \sqrt{\frac{\sum_{i=1}^{Q} (Hata(\%)_i - Ortalama\ Hata(\%))^2}{Q-1}} \tag{7.3}$$

Yukarıda özellikleri verilen veri kümeleri hem ileri beslemeli, hem de geri dönüşümlü YSA modelleri ile kullanılarak 14 ayrı test yapılmıştır.

7.1 Veri Kümelerinin Özellikleri

Oluşturulan yapay sinir ağı modellerinin performanslarını değerlendirmek ve simülasyonlarda kullanmak üzere 7 adet veri kümesi oluşturulmuştur. Bu veri kümeleri içerdikleri örneklerin seçimi ve bu örneklerin ağa gösterim şekilleri açısından farklılıklar göstermektedir.

Bu çalışmada kullanılan verilerin nasıl işlendiği Bölüm 6.6'da açıklanmıştır. Ancak, örneklerin ağa sunulma şekli de öğrenme performansını etkileyebilir. Oluşturulan veri kümeleri, ağa sıralı ve rasgele sunulmak üzere de birbirlerinden ayrılmıştır. Test için kullanılacak verilerin miktarı eğitim verilerinin en az % 25'ini oluşturacak şekilde seçilmiştir.

Oluşturulan veri kümelerine ait temel özellikler aşağıda gösterilmiştir (Çizelge 7.1).

Çizelge 7.1 Simülasyonlarda kullanılan veri kümelerinin özellikleri

Veri Kümesi	Eğitim için Kullanılan Veri Miktarı	Doğrulama için Kullanılan Veri Miktarı	Test için Kullanılan Veri Miktarı	Verilerin Türü	Verilerin Ağa Sunulma Şekli
1	160	16	80	FFT- Aritmetik işlem süreleri	Rasgele
2	68	12	48	MC- Aritmetik işlem süreleri	Sıralı
3	50	25	50	MC- Aritmetik işlem süreleri	Sıralı
4	220	40	60	MC- Aritmetik işlem süreleri	Rasgele
5	30	10	20	FFT-Haberleşme süreleri	Sıralı
6	150	25	50	MC-Haberleşme süreleri	Sıralı
7	220	40	60	MC-Haberleşme süreleri	Rasgele

Örneklerin seçimi, ağ ağırlıkları bu örnekler dikkate alınarak ayarlandığı için, ağın performansını da yakından ilgilendirir. Seçilen örneklerin problem uzayını temsil edecek nitelikte olması gerekir. Ancak, modellerin verilerin eksik olması durumundaki başarılarını da kıyaslayabilmek amacı ile 1 numaralı veri kümesi yüksek sayıda örnek içermekle birilikte uç değerleri içermeyecek şekilde oluşturulmuştur. 2 numaralı veri kümesi de serinin önemli bir kısmını içermekle birlikte uç değerlerde az örnek içermektedir. Diğer veri kümeleri ise problem uzayını iyi temsil edecek şekilde oluşturulmuştur.

7.2 Simülasyon Sonuçları ve YSA Performanslarının Değerlendirilmesi

Yukarıda özellikleri verilen veri kümeleri hem ileri beslemeli, hem de geri dönüşümlü YSA modelleri ile kullanılarak 14 ayrı test yapılmıştır. Bu testler için hangi YSA modelinin kullanıldığı ve hesaplanan başarı oranları Çizelge 7.2'de gösterilmiştir. Ayrıca, simülasyonlar ile elde edilen sonuçlar bu bölümün sonunda grafiksel olarak verilmiştir.

Simülasyon sonuçları, modellerin eğitiminde kullanılan verilerin yapısı açısından incelendiğinde, ileri beslemeli YSA modeli için verilerin ağa gösterim şeklinin sıralı ya da rasgele olmasının bir önem taşımadığı ancak eğitim verilerinin yeterli çeşitlilikte ve sayıda örnek içermemesi halinde başarının büyük ölçüde düştüğü gözlenmiştir.

Geri dönüşümlü model ise, verilerin ağa sıralı gösterilmesi durumunda daha hızlı eğitilebilmekte ve daha başarılı olmaktadır.

Daha da önemlisi, geri dönüşümlü YSA modeli, eğitim verilerinde eksik örneklerin olduğu durumları, veriler ağa rasgele gösterilmiş olduğunda dahi, ileri beslemeli modelden daha iyi

tolere edebilmiş ve daha başarılı sonuçlar vermiştir. Yapılan testlerden ilki buna iyi bir örnek teşkil etmektedir. Tipik olarak geri dönüşümlü YSA modeli için elde edilen başarı oranları %95'in üzerindedir. Eğitim verilerinin iyi seçilmesi durumunda ileri beslemeli model de, paralel sistemlerin performansının tahmininde, her seferinde benzer yüksek başarı oranlarını yakalamaktadır (Yavuz, 2005c).

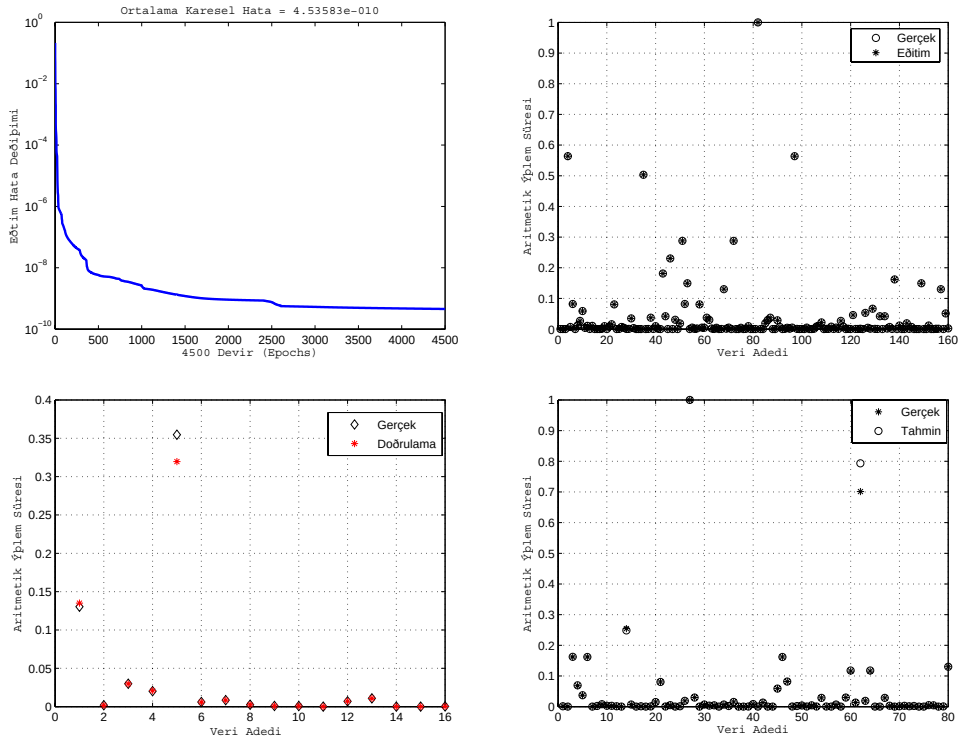
Eğitim süreleri açısından incelendiğinde, Elman ağının eğitimi daha yavaş olmakla birlikte, öğrenmenin gerçekleşmesi için verilerin ağa gösterilme sayısı (devir sayısı) daha düşük tutulabilmektedir. Özellikle verilerin sıralı olarak ağa gösterilmesi durumunda Elman ağının toplam eğitim süresinin ileri beslemeli ağa oranla daha kısa olduğu gözlenmiştir.

Çizelge 7.2 Simülasyonlarda kullanılan veri kümelerinin özellikleri

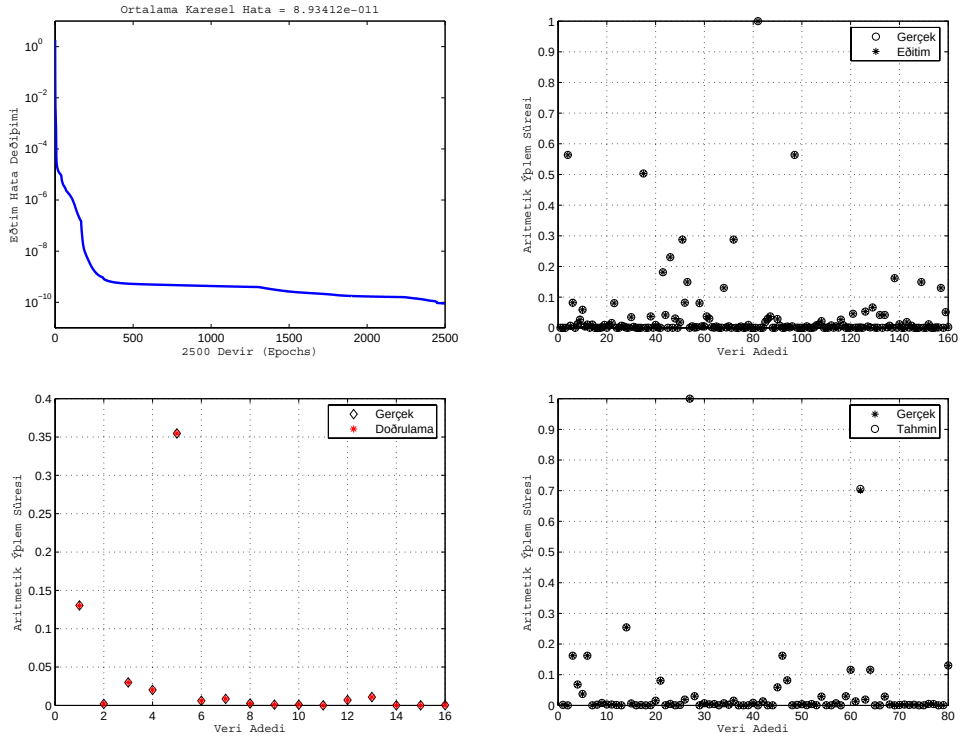
Kullanılan YSA Modeli	Verilerin Ağa Sunulma Şekli	Kullanılan Veri Kümesi	Eğitim Başarımı (okh)	Doğrulama Başarımı (okh)	Test Başarımı (okh)	Test Sonuçları için	
						% Hata Ortalaması	Standart Sapma
İleri Beslemeli	Rasgele	1	4.54E-10	7.80E-05	1.06E-04	9.865205	23.6927
Geri Dönüşümlü	Rasgele	1	8.93E-11	6.57E-09	2.46E-07	1.93003	4.752885
İleri Beslemeli	Sıralı	2	3.92E-14	1.40E-10	9.21E-09	4.33158	4.355392
Geri Dönüşümlü	Sıralı	2	4.70E-15	1.09E-14	3.24E-11	0.204894	0.302628
İleri Beslemeli	Sıralı	3	2.11E-13	1.94E-11	2.83E-10	0.455746	0.331489
Geri Dönüşümlü	Sıralı	3	3.42E-13	6.17E-11	9.20E-10	0.979992	0.411128
İleri Beslemeli	Rasgele	4	2.42E-11	9.81E-06	5.06E-04	0.449739	2.356225
Geri Dönüşümlü	Rasgele	4	3.13E-10	3.83E-08	1.98E-06	0.226874	0.587377
İleri Beslemeli	Sıralı	5	1.77E-13	4.21E-10	6.97E-10	0.876622	1.275427
Geri Dönüşümlü	Sıralı	5	1.62E-13	1.60E-12	9.99E-12	0.455203	0.838889
İleri Beslemeli	Sıralı	6	8.90E-08	1.83E-07	9.81E-08	0.222201	0.171142
Geri Dönüşümlü	Sıralı	6	8.58E-08	1.56E-07	9.33E-08	0.210915	0.190792
İleri Beslemeli	Rasgele	7	1.66E-08	2.08E-05	1.27E-06	1.004455	1.708429
Geri Dönüşümlü	Rasgele	7	4.91E-07	3.13E-05	2.71E-06	2.13881	3.215285

Sonuç olarak geri dönüşümlü ağların, verilerin ağa sıralı veya rasgele gösterilmesi durumlarının her ikisinde de, daha başarılı olduğu söylenebilir. Bu başarıda, kullanılan içerik elemanları vasıtası ile girdiler arasındaki ilişkiyi temsil eden modelin yanı sıra aktivasyon fonksiyonlarının da bir modelinin oluşturulması etkilidir.

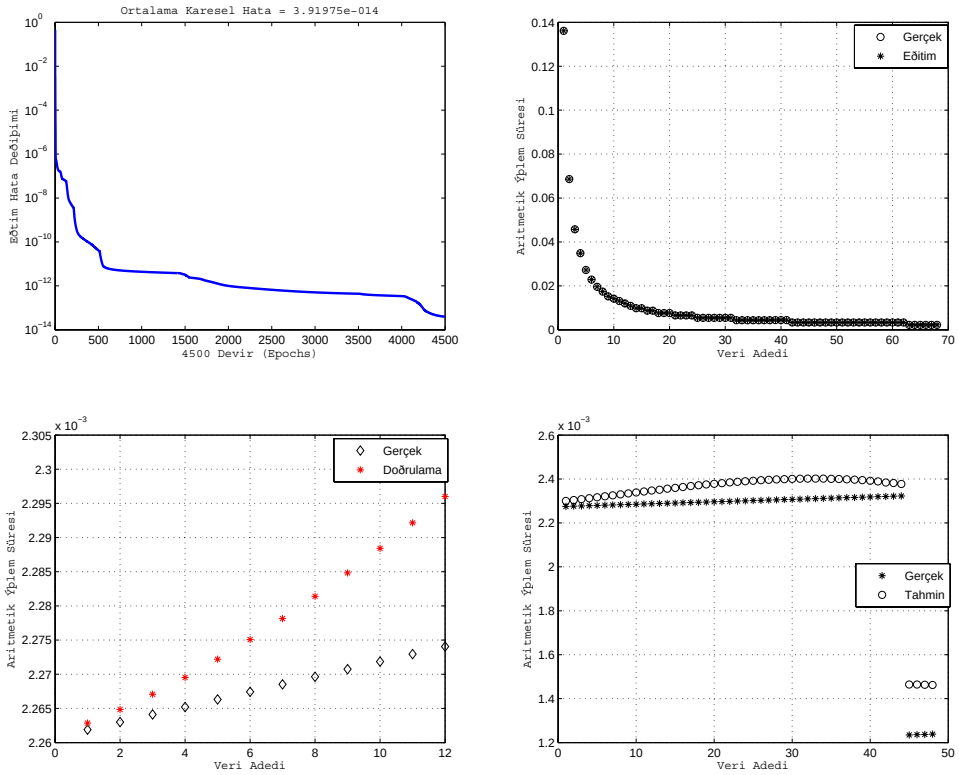
Çizelge 7.2 de özetlenen testlere ait simülasyon sonuçları Şekil 7.1 ila 7.14'de grafiksel olarak da verilmiştir. Şekillerde, soldan sağa doğru sırası ile ağın eğitim sürecinde ortalama karesel hatanın değişimini, eğitim başarımını, doğrulama verileri için başarımını ve test verileri için başarımını göstermektedir.



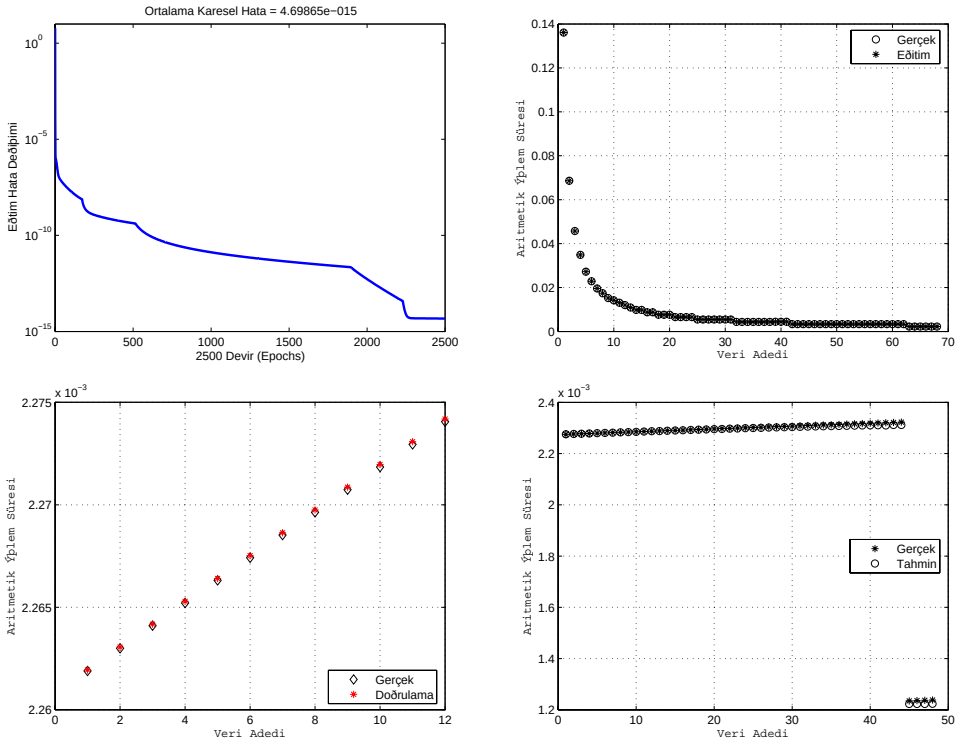
Şekil 7.1 İleri beslemeli YSA için 1 numaralı veri kümesi ile elde edilen sonuçlar



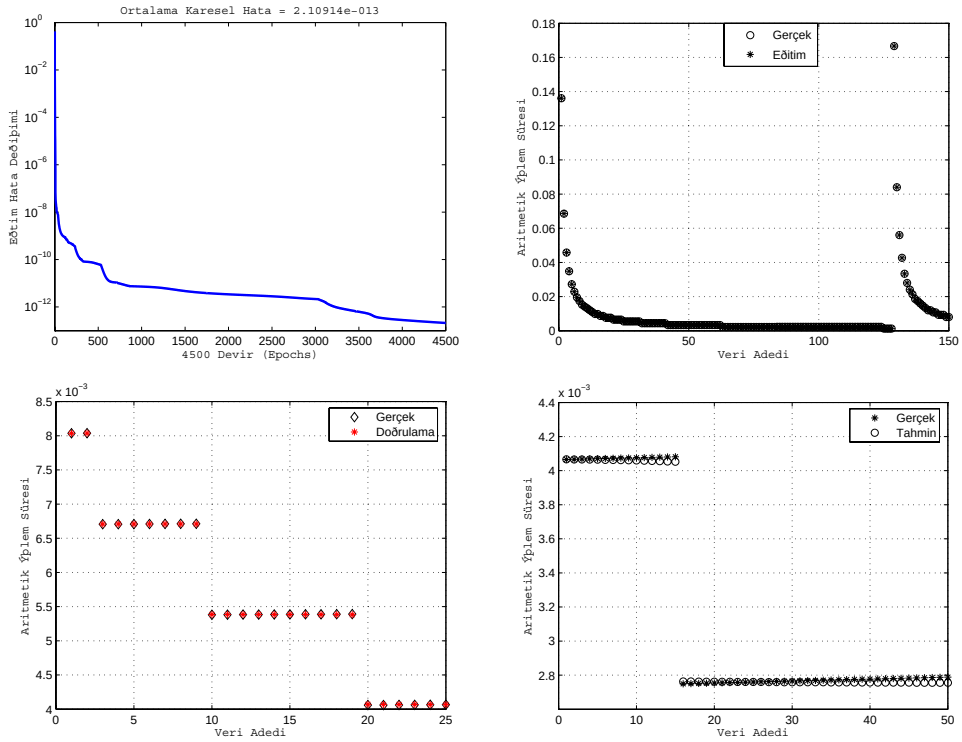
Şekil 7.2 Geri Dönüşümlü Elman ağı için 1 numaralı veri kümesi ile elde edilen sonuçlar



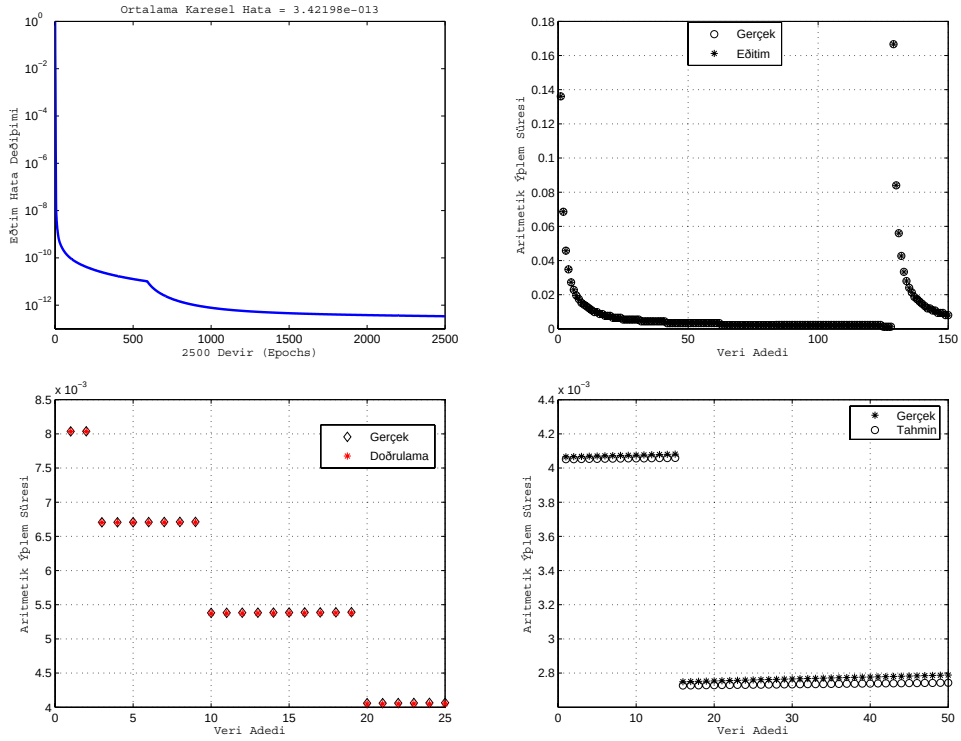
Şekil 7.3 İleri beslemeli YSA için 2 numaralı veri kümesi ile elde edilen sonuçlar



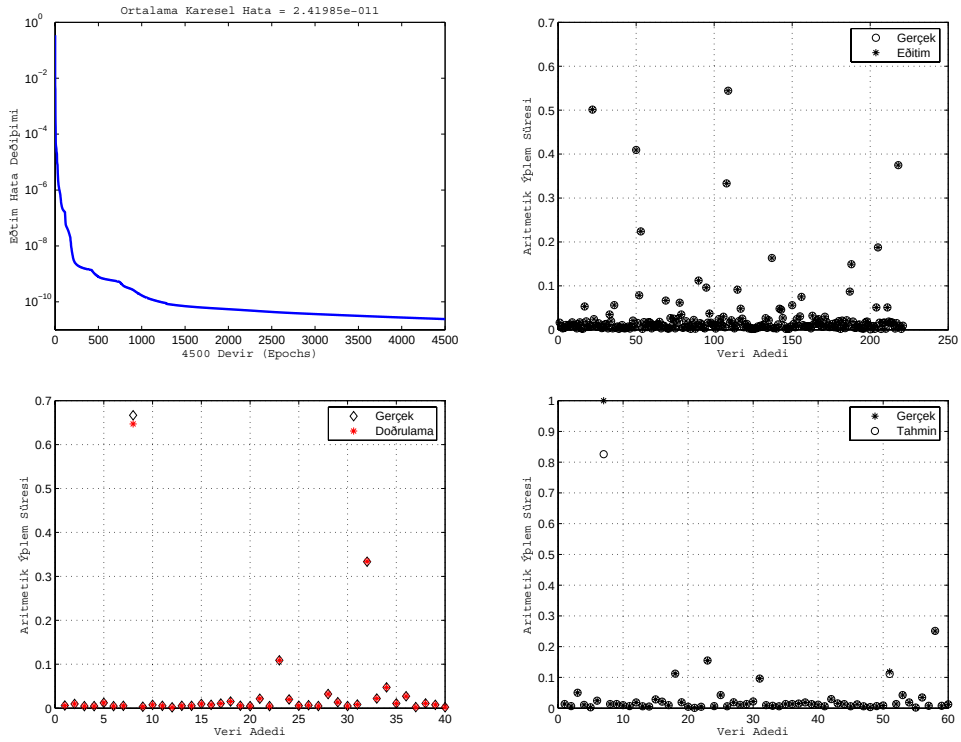
Şekil 7.4 Geri Dönüşümlü Elman ağı için 2 numaralı veri kümesi ile elde edilen sonuçlar



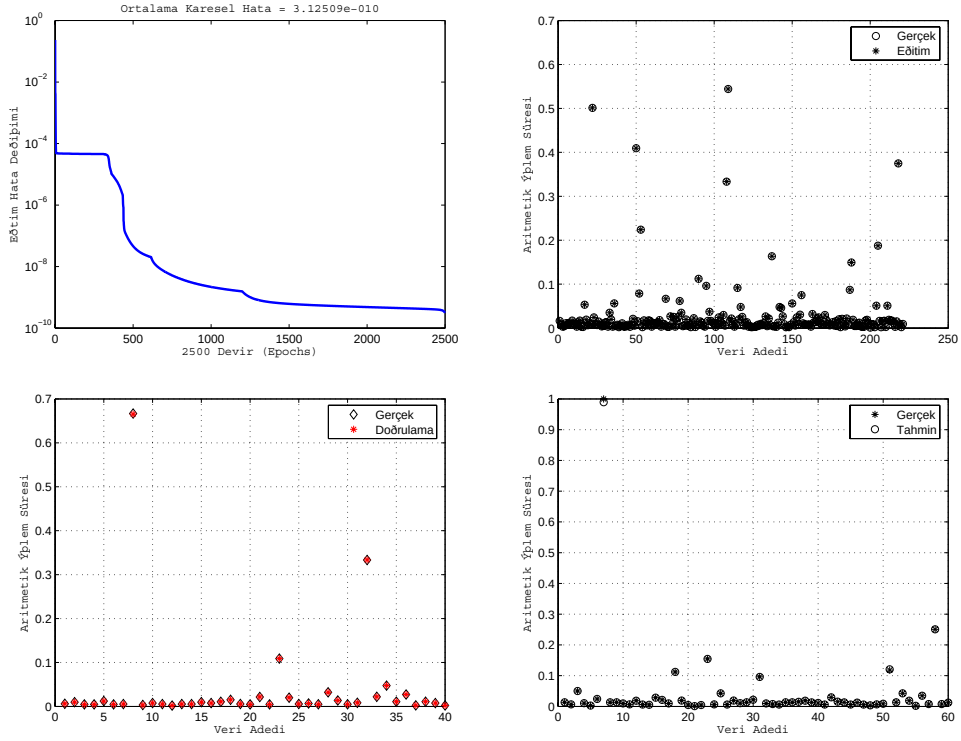
Şekil 7.5 İleri beslemeli YSA için 3 numaralı veri kümesi ile elde edilen sonuçlar



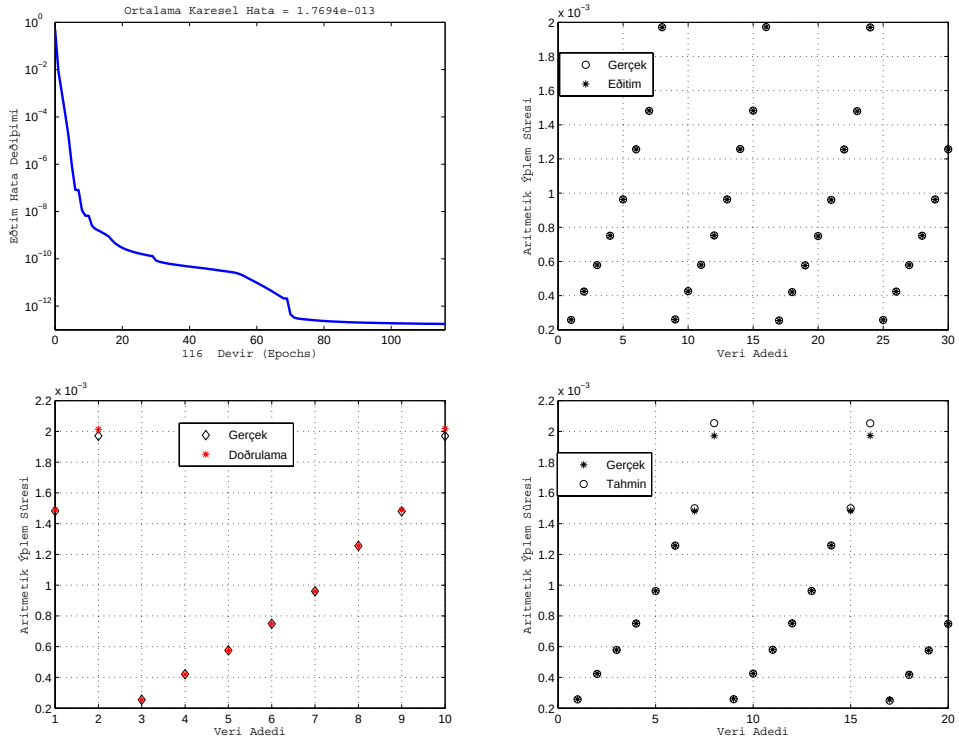
Şekil 7.6 Geri Dönüşümlü Elman ağı için 3 numaralı veri kümesi ile elde edilen sonuçlar



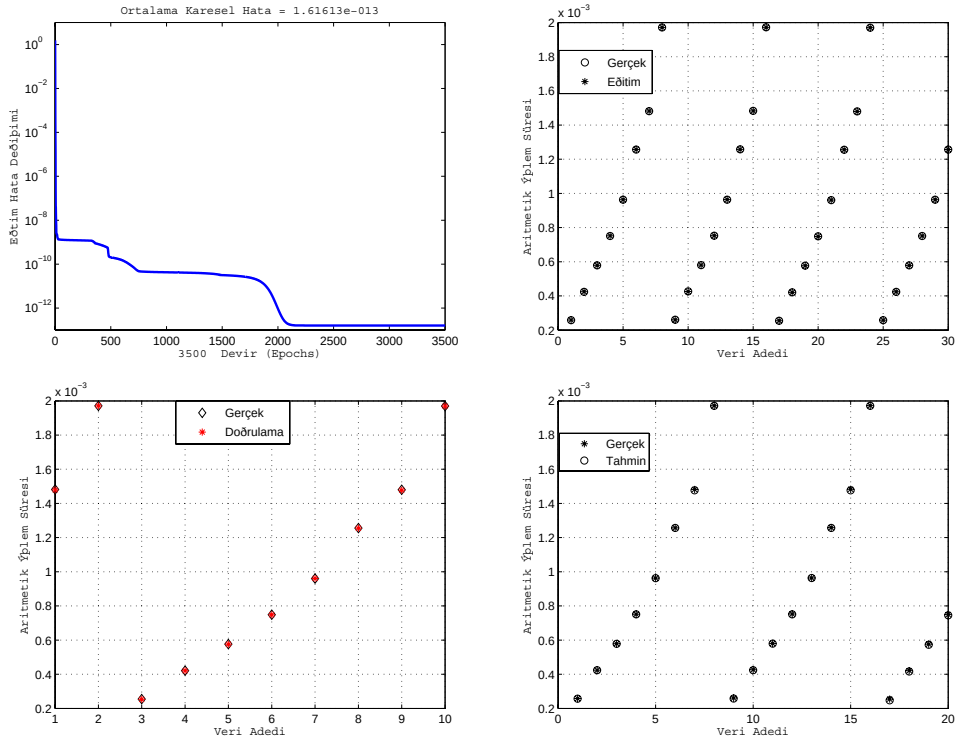
Şekil 7.7 İleri beslemeli YSA için 4 numaralı veri kümesi ile elde edilen sonuçlar



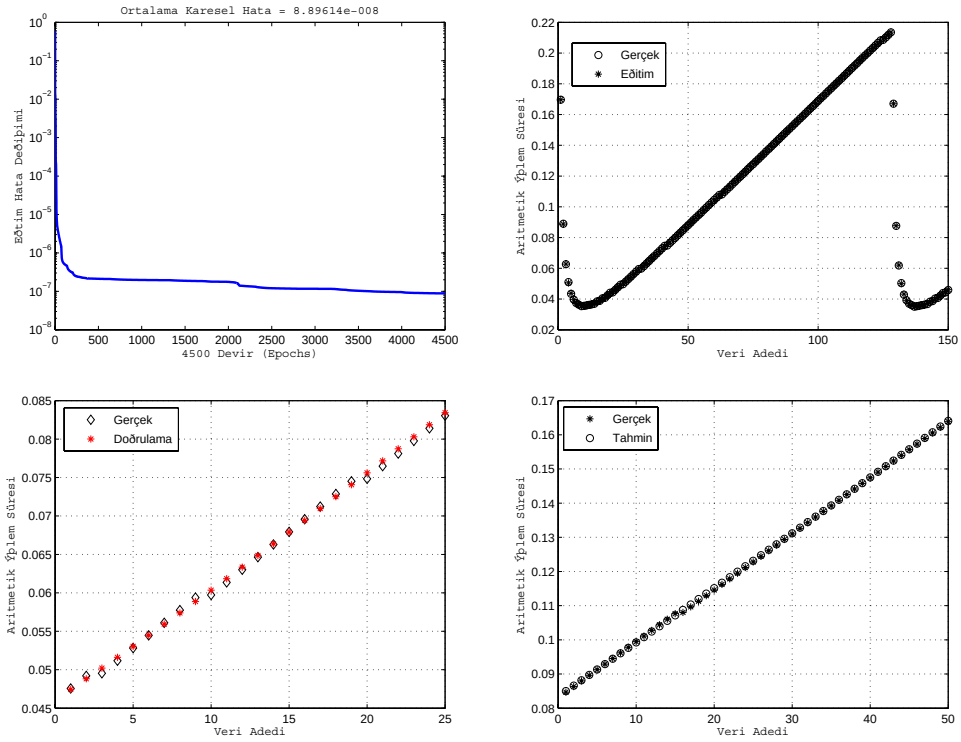
Şekil 7.8 Geri Dönüşümlü Elman ağı için 4 numaralı veri kümesi ile elde edilen sonuçlar



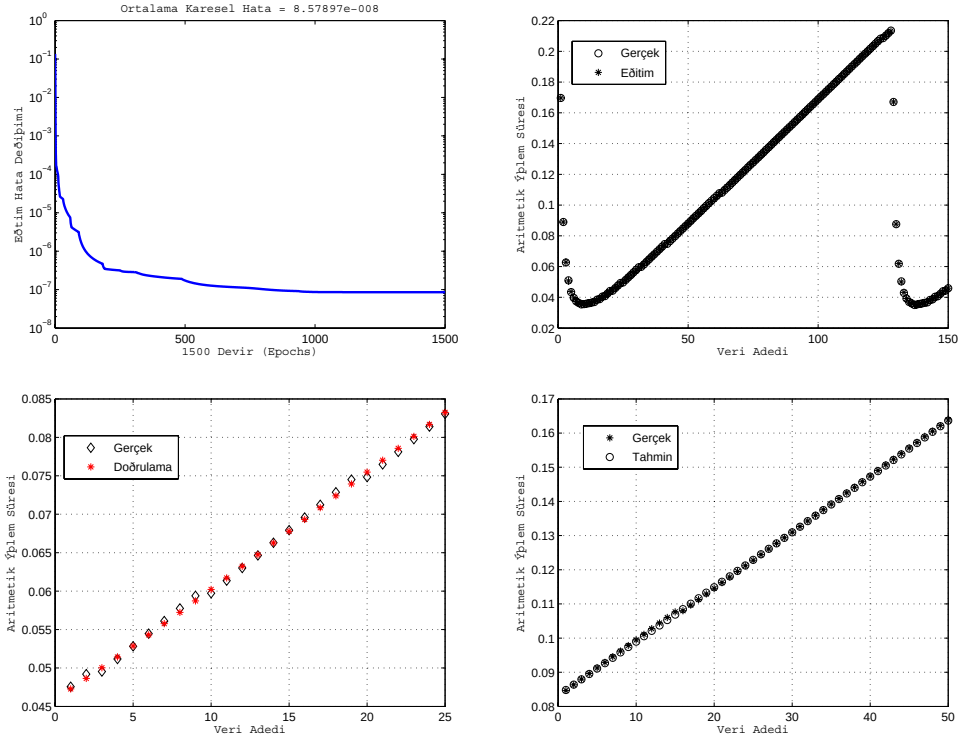
Şekil 7.9 İleri beslemeli YSA için 5 numaralı veri kümesi ile elde edilen sonuçlar



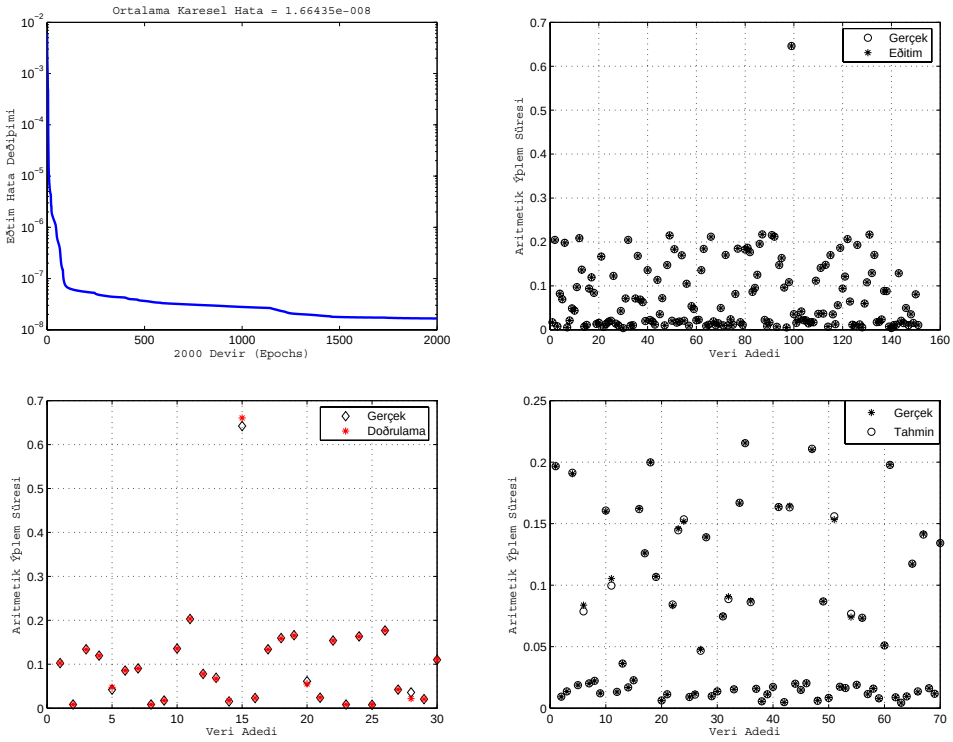
Şekil 7.10 Geri Dönüşümlü Elman ağı için 5 numaralı veri kümesi ile elde edilen sonuçlar



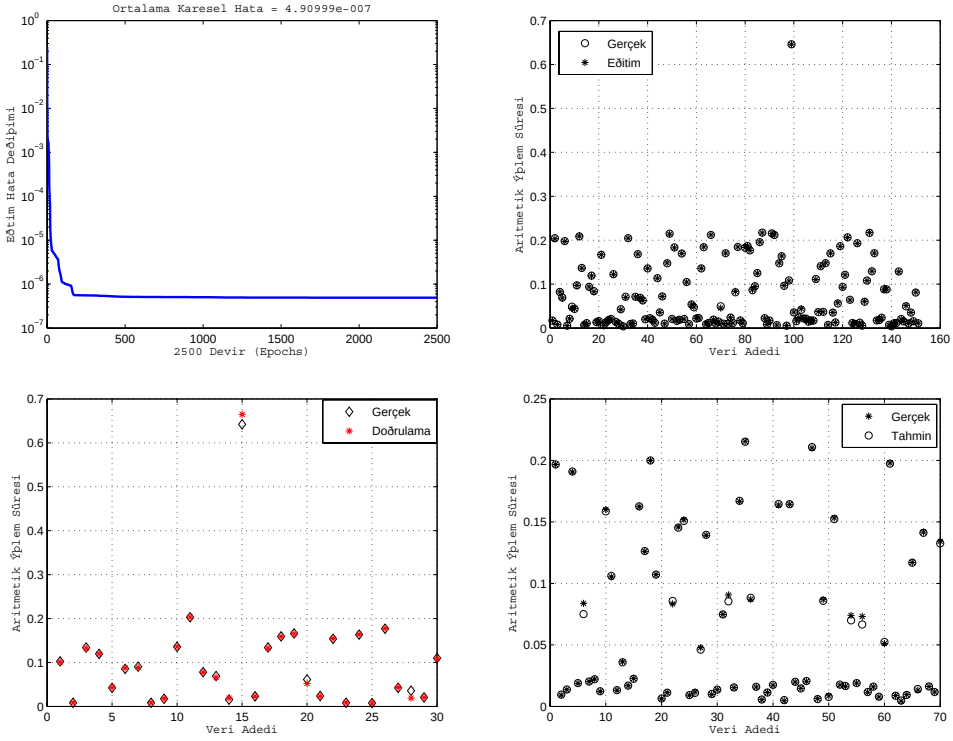
Şekil 7.11 İleri beslemeli YSA için 6 numaralı veri kümesi ile elde edilen sonuçlar



Şekil 7.12 Geri Dönüşümlü Elman ağı için 6 numaralı veri kümesi ile elde edilen sonuçlar



Şekil 7.13 İleri beslemeli YSA için 7 numaralı veri kümesi ile elde edilen sonuçlar



Şekil 7.14 Geri Dönüşümlü Elman ağı için 7 numaralı veri kümesi ile elde edilen sonuçlar

8. SONUÇLARIN TARTIŞILMASI

Gelişen teknolojiler ve paralel sistemlerin yaygınlaşması sayesinde performans analizi giderek artan bir gereklilik haline gelirken geleneksel yöntemler arasında oluşan boşluğu dolduracak, evrensel ve kullanımı daha kolay olan modellere ihtiyaç da artmaktadır. Bu tez çalışmasında, özellikle haberleşme ağları üzerinde paralel çalışan kümelenmiş bilgisayarlar için yapay sinir ağlarını kullanarak bir performans tahmin ve analiz yöntemi geliştirilmiştir.

Performans tahmini alanında istatistiksel yöntemler şimdiye kadar yaygın olarak kullanılmış olmasına rağmen, yapay sinir ağlarının bu amaçla kullanımı ilk kez bu çalışmada önerilmiştir. Elde edilen sonuçlar yapay sinir ağlarının ve özellikle geri dönüşümlü ağların bu alanda başarı ile kullanılabileceği yönündedir.

Performans değerlendirme alanında eskiden beri kullanılan en temel iki teknik analitik modelleme ve simülasyondur. Ancak gelişen teknolojiler ve paralel sistemlerin yaygınlaşması bu yöntemlerin kullanılmasının gittikçe zorlaştırmıştır.

Analitik ve simülasyon yöntemleri ile oluşturulan modeller ne kadar detaylı olursa olsun, gerçek sistemlerde bu tür modeller ile yakalanması güç olan girişimler bulunabilir. Bu durum, modellerin geliştirilmesindeki güçlükler de göz önüne alındığında, karşılaştırmalı değerlendirmelerin kullanımını artırmıştır.

Karşılaştırmalı değerlendirmelerin performans analizinde kullanılmasının en önemli dezavantajı, kullanılan kodun özellikleri ile sistemin özellikleri arasında herhangi bir ilişki kurmaya olanak vermemeleri, elde edilen sonuçların neden elde edildiğine dair ipucu içermemeleridir. Karşılaştırmalı değerlendirmeler, gerçek sistemler üzerinde çalışan gerçek kodlar olmakla birlikte, son kullanıcının uygulamasını temsil etmezler.

Bu çalışmada oluşturulan yapay sinir ağı modelleri, gerçek kullanıcı kodlarının kullanılmasına ve sunulan modellerin girdilerini oluşturan donanım ve yazılım parametreleri arasındaki etkileşimleri izlemeye olanak verdiği için, karşılaştırmalı değerlendirmelerden daha sağlıklı ve detaylı sonuçlar verecektir (Yavuz, 2005d).

Oluşturulan modeller, gerçeklenmelerinin kolaylığı ve modellerin oluşturulması sırasında birtakım varsayımlara ihtiyaç bırakmaması açısından, simülasyon ve analitik yöntemlere de alternatif oluşturmaktadır.

PACE (Nudd, 1999) gibi hem analitik hem deneysel modelleme yöntemlerini kullanan performans analiz araçları için hata oranları %10 olarak verilmektedir. Bu hata oranı, performans tahmini alanında kabul edilebilir bir orandır ve bu tür araçlar oldukça başarılı

olarak nitelendirilebilirler. Önerilen YSA modellerinin başarısı incelendiğinde, özellikle Elman ağı kullanılarak oluşturulan model için, hata oranı -eğitim verilerinin yeterli ve sıralı olması halinde- %1'in altına düşebilmekte ve her zaman %5'in altında kalmaktadır. Bu verilere dayanarak yapay sinir ağlarının performans analizi ve tahmini alanında daha başarılı sonuçlar elde edilmesinin yolunu açacak önemli araçlar olduğu söylenebilir.

Burada önerilen YSA modellerinin girişleri, analiz edilmek istenen diğer parametrelerin de eklenmesi ile artırılabilir. Benzer şekilde, kullanıcılar kendi uygulamalarına ait verileri kullanarak daha sağlıklı ve detaylı analizler yapma ve donanım ihtiyaçlarını doğru belirleyebilme olanağına sahiptirler.

Örneklerin ağa sunulma şekli ve eğitim verilerinin miktarı öğrenme performansını büyük ölçüde etkileyebilir. Oluşturulan yapay sinir ağı modelleri, bu açıdan karşılaştırıldığında kısmi geri dönüşümlü ağların çok katmanlı algılayıcılardan daha istikrarlı ve başarılı sonuçlar verdiği ortaya çıkmaktadır. Kısmi dönüşümlü ağların bu başarısı, içerik elemanları aracılığı ile önceki durumları hatırlayabilmeleri ve girdiler arasındaki ilişkiyi daha doğru oluşturmaları ile ilgilidir. Sonuçlar bu açıdan incelendiğinde, tam geri dönüşümlü ağların kullanılması durumunda daha fazla girdi kullanılabilmesi ve ilişkilerin daha detaylı oluşturulması mümkün olabilir.

Yapay sinir ağı modellerinin kullanımını kolaylaştırabilmek ve hataları daha da azaltabilmek için, girdiler arasında iyi bilinen ilişkilerin bulanık mantık kuralları ile ifade edilmesi ve hibrid sistemler oluşturulması faydalı olabilecektir.

Kullanılacak uygulamaların nesne tabanlı olması durumunda, uygulama özelliklerini temsil eden daha kolay elde edilebilir girdiler seçilebilecektir.

KAYNAKLAR

ATM Forum, (1995), "ATM User Level Network Interface Specification", Prentice Hall, NJ.

Addision C.A., Getov V.S., Hey A.J.G., Hockney R.W. ve Wolton I.C., (1993),"The

GENESIS Distributed-Memory Benchmarks", Computer Benchmarks, North-Holland.

Agarwal A., (1991),"Limits on Interconnection Network Performance", IEEE Transactions on Parallel and Distributed Systems, 2(4),398—412.

Atiya A. F., (2001) , "Bankruptcy Prediction For Credit Risk Using Neural Networks: A Survey and New Results", IEEE-NN 12, 929-935.

Atiya A.F., El-Shoura S.M., Shaheen S.I. ve El-Sherif M. S., (1999), "A Comparison between Neural-Network Forecasting Techniques--Case Study: River Flow Forecasting", IEEE-NN 10, 402.

Bailey D.H., Barszcz E., Barton J.T., Browning D.S., Carter R.L. vd., (1991), "The NAS Parallel Benchmarks", The International Journal of Supercomputer Applications, 5, 63-73.

Bailey D.H., Harris T., Sephir W., Wijngeert R., Woo A., Yarrow M., (1995), "The NAS Parallel Benchmarks 2.0. Report", NAS-95-020, Numerical Aerodynamic Simulation Facility, NASA Ames Research Center.

Basu A., Srinivas S., Kumar K.G., Paulraj A., (1990), "A Model for Performance Prediction of Message Passing Multiprocessors Achieving Concurrency by Domain Decomposition", Proceedings of the Joint International Conference on Vector and Parallel Processing, 75-85.

Blume W. ve Eigenmann R., (1992), "Performance Analysis of Parallelizing Compilers on the Perfect Benchmarks Programs", IEEE Transactions on Parallel and Distributed Systems, 3(6), 643-656.

Boden N. vd., (1995), "Myrinet - A Gigabit-per-Second Local-Area Network", IEEE Micro.

Bomans I. ve Roose D., (1989), "Benchmarking the iPSC/2 Hypercube Multicomputer", Concurrency: Practice and Experience, 1(1), 3-18.

Brewer E. A., Dellarocas C. N., Colbrook A., Weihl W. E., (1991), "PROTEUS: A High-Performance Parallel-Architecture Simulator", Technical Report MIT/LCS/TR-516, Massachusetts Institute of Technology, Cambridge, MA 02139.

Browne S., Dongarra J. ve London K., (1998), "Review of Performance Analysis Tools for MPI Parallel Programs", NHSE Review.

Buyya R., (1999), "High Performance Cluster Computing, Programming and Applications", Prentice Hall.

Chunrong Y. ve Niemann H., (2000), "An Appearance Based Neural İmage Processing Algorithm for 3-D Object Recognition", International Conference on Image Processing, 344-347.

Crawford J., (2000), "Introducing the Itanium Processors", IEEE Micro, 9-11.

Curnow H.J. ve Wichmann B.A., (1976), "A Synthetic Benchmark", The Computer Journal 19(1), 43 – 49.

Cybenko G., Kipp L., Pointer L. ve Kuck D., (1990), "Supercomputer Performance Evaluation and the Perfect Benchmarks",Technical Report 965, University of Illinois at Urbana-Champaign, Center for Supercomputing Research and Development, Urbana, IL.

Dally W. J., (1990), "Performance Analysis of k-Ary n-Cube Interconnection Networks", IEEE Transactions on Computers 39(6), 775-785.

Demuth H. ve Beale M., (2001), "User's Guide for the Neural Network Toolbox for MATLAB version 4", The Mathworks Inc,, Natick, MA.

Dennis J.E. ve Schnabel R.B., (1983), "Numerical Methods for Unconstrained Opti-

mization and Nonlinear Equations", Prentice Hall, New Jersey.

Dixit K. ve Reilly J., (1991), "SPEC Developing New Component Benchmark Suites", SPEC Newsletter 3(4), 14-17.

Dongarra J.J., (1993), "Linear Algebra Libraries for High-Performance Computers. A personal Perspective", IEEE parallel and distributed technology, systems and applications 1(1), 17-24.

Dongarra J. J., (1995), "Performance of Various Computers Using Standard Linear Equations Software", Tech. Rep. CS-89-85, University of Tennessee and Oak Ridge National Laboratory.

Dongarra J. J., Martin J. ve Worlton J., (1987), "Computer Benchmarking Paths And Pitfalls", IEEE Computer 24, 38-43.

Elman, J. L., (1990), "Finding Structure in Time", Cognitive Science, 14, 179-211.

Fine T.L., (1999), "Feedforward Neural Network Methodology", Springer Series in Statistics, Springer Verlag, ISBN: 0387987452.

Fleming P.J., Wallace J.J., (1986), "How Not To Lie With Statistics: The Correct Way To Summarize Benchmark Results", Communications of the ACM 29(3), 218-221.

Fletcher R., (2000), "Practical Methods of Optimization", 2nd edition, John Wiley & Sons Ltd., ISBN: 0471494631.

Foster I., Kesselman C. ve Tuecke S., (2001), "The Anatomy of the Grid: Enabling Scalable Virtual Organization", The International Journal of High Performance Computing Applications 15(3), 200-222.

Foster I., Gropp W. ve Stevens R., (1991), "The Parallel Scalability of the Spectral Transform Method", Technical report, MCS Division, Argonne National Laboratory, Argonne, IL.

Geist G.A., Beguelin A., Dongarra J., Jaing W., Manchek R., Sunderam V., (1994), "PVM: Parallel Virtual Machine - A Users' Guide and Tutorial for Networked Parallel Computing", Scientific and engineering computation, MIT Pres.

Gloria A., Ancarani F., Bellotti F., Olivieri M., (1997), "Instruction Level Analytic Prediction Of Parallel Cpu Architecture Performance" IEEE, 530-534.

Góes, L.F.W., Ramos, L.E.S., Martins C.A.P.S., (2004), "ClusterSim: a Java-Based Paralell Discrete-Event Simulation Tool for Cluster Computing", 2004 IEEE International Conference on Cluster Computing, 401 – 410.

Gropp W., (2005), "Tutorial on MPI. The Message Passing Interface", Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Illinois.

Gustafson J., (1991), "SLALOM Update"" Supercomputing Review, March 1991, 56-61.

Hagan M. ve Menhaj M., (1994), "Training Feedforward Networks with the Marquardt Algorithm", IEEE Transactions on Neural Networks 5(6), 989-993.

Hagan M. T., Demuth H. B. ve Beale M. H., (1996), "Neural Network Design", PWS

Publishing, Boston, MA.

Han J. ve Kamber M., (2001), "Data Mining Concepts and Techniques", San Francisco, Academic Pres.

Heath M.T., Etheridge J.A., (1991), "Visualising the Performance of Parallel Programs", IEEE Software, 29-39.

Hebb, D.O., (1949), "The Organization of Behavior", John Wiley, New York.

Hennessy J.L., Patterson D.A., (1996), "Computer Architecture, A Quantitative Approach", Morgan Kaufmann.

Henning J.L., (2000), "SPEC CPU2000. Measuring CPU Performance in the New Millennium", Computer 33(7), 28-35.

Hockney R.W., (1996), "The Science of Computer Benchmarking: Software, Environments, Tools", Society for Industrial and Applied Mathematics (SIAM).

Hockney R.W., (1993), "Performance Parameters And Benchmarking Of Supercomputers, Computer Benchmarks" Elsevier Science Publishers, 41-63.

Hopfield J.J., (1982), "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", Proc. National Academy of Sciences, 2,554-2,558.

Jackel P., (2002), "Monte Carlo Methods in Finance", John Wiley & Sons Ltd.,England.

Jain A. K., Mao J., Mohiuddin K. M., (1996), "Artificial Neural Networks: A Tutorial", IEEE Computer 29(3), 31-44.

Jain R., (1991), "The Art of Computer Systems Performance Analysis - Techniques for Experimental Design, Measurement, Simulation and Modelling", John Wiley & Sons.

James D.V., Laundrie A.T., Gjessing S. ve Sohi G.S., (1990), "Scalable Coherent Interface", IEEE Computer 23, 74-77.

John L. K., (2004), "More on Finding a Single Number to İndicate Overall Performance of a Benchmark Suite", ACM SIGARCH Computer Architecture News, 32(1).

Jordan, M. I., (1986), "Attractor Dynamics and Parallelism in a Connectionist Sequential Machine", Proceedings of the Eighth Annual Conference of the Cognitive Science Society, 531-546, Amherst.

Kerbyson D.J., Papaefstathiou E., Nudd G.R., Atherton T.J., (1995), "Evaluation of Parallel Operating Systems And Languages", ESPRIT Project 6942 (PEPS) Report 5.5, Department of Computer Science, University of Warwick, Coventry, U.K.

Kleinrock L., (1993), "On the Modeling and Analysis of Computer Networks", Special Issue of IEEE Proceedings, 1179-1191.

Knuth E. D., (1981), "The Art Of Computer Programming Vol. 2", 2 nd Edition, Addison-Wesley.

Kohonen, T., (1988), "Self-Organization and Associative Memory", Springer-Verlag, New York.

Kohonen, T., (1995), "Self-Organizing Maps", Springer-Verlag, Berlin Heidelberg.

Kröse B. ve Smagt P., (1996), "An Introduction to Neural Networks", 8th edition, University of Amsterdam.

Lenoski D., (1997), " Multiprocessor Design Options and the Silicon Graphics SSMP Architecture", Silicon Graphics Inc.

Lewis B., (1993), “An Ethernet Performance Simulator for Undergraduate Networking”, Proceeding of ACM SIGCSE Technical Symposium.

Marshall T. ve Tazelaar J.M., (1989), "Worth the RISC: RISC Technology Is Here, It's Usable, and It's Low-Cost Now”, Byte 14(2), 245.

Martin J.L., (1987), "Performance Evaluation: Applications and Architectures”, Second International Conference on Supercomputing, 369 – 373.

Masters T., (1993), "Practical Neural Network Recipes in C++”, Academic Press Inc.

McMahon F. H., (1988), "The Livermore Fortran Kernels Test of the Numerical Performance Range”, Performance Evaluation of Supercomputers: Special Topics in Supercomputing, 143-186, North-Holland.

Miller B.P., Callaghan M.D., Cargille J.M., Hollingsworth J.K., Irvin R.B., Karavanic K.L., Kunchithapadam K. ve Newhall T., (1995), "The Paradyn Parallel Performance Measurement Tools”, IEEE Computer 28(11), Special issue on performance evaluation tools for parallel and distributed systems.

Miller B.P., Clark M., Hollingsworth J., Kierstead S., Lim S.S., Torewski T., (1990), "IPS-2, The Second Generation of Parallel Program Measurement System”, IEEE Transactions on Parallel and Distributed Systems, 206-217.

Minsky M. ve Papert S., (1969), “Perceptrons”, MIT Press, Cambridge.

Mitchell M.T., (1997), "Machine learning”, The McGraw-Hill Companies, New York.

Neji Z. ve Beji F.M., (2000), “Neural Network and Time Series Identification and Prediction”, IEEE INNS-ENNS International Joint Conference on Neural Networks (IJCNN'00), Italy.

Neves K.W. ve Simon H.D., (1987), "Supercomputer Performance Evaluation: Benchmarking Applications on Supercomputers. In Second International Conference on Supercomputing, pages 374 – 379.

Normoyle K.B., Csoppenszky M.A., Tzeng A., Johnson T.P., Furman C.D., Mostoufi J., (1998), "ULTRASPARC-III: Expanding the Boundaries of a System on a Chip”, IEEE Micro, 14-24.

Nudd G.R.,Kerbyson D.J., Papaefstathiou E., Harper J.S., Perry S.C., Wilcox D.V., (1999), "PACE: A Toolset For The Performance Prediction Of Parallel And Distributed Systems”, Journal of High Performance and Scientific Applications.

Nudd G.R., Papaefstathiou E., Papay J., Therton T.J., Clarke C.T., Kerbyson D.J., Stratton A.F., Ziani R.F., Zemerly J., (1993), "A Layered Approach To The Characterisation Of Parallel Systems For Performance Prediction”, Proc. of the Performance Evaluation of Parallel Systems Workshop (PEPS 93), Coventry, U.K.

Öztemel E., (2003), “Yapay Sinir Ağları”, Papatya Yayıncılık, İstanbul.

Pacheco P.S., (1997), "Parallel Programming with MPI”, Morgan Kaufmann.

Papaefstathiou E., (1995), "A Framework for Characterising Parallel Systems for Performance Evaluation”, PhD thesis, University of Warwick.

Papaefstathiou E., Kerbyson D.J., Nudd G.R., Atherton T.J., (1995), "An Introduction to the CHIP3S Language For Characterising Parallel Systems In Performance Studies”, Technical report, Parallel Systems Group, University of Warwick.

Papaefstathiou E., Kerbyson D.J., Nudd G.R., (1994), "A Layered Approach to Parallel

Software Performance Prediction. A Case Study”, Proc. International Conference Massively Parallel Processing, Delft, Holland.

Patterson D.A ve Sequin C. H., (1981), "RISC I: A reduced Instruction Set VLSI Computer”, 8th International Symposium on Computer Architecture.

Patterson D.A., (1985), "Reduced Instruction Set Computers”, Communications of the ACM 28(1), 8-21.

Press W.H., Flannery B.P, Teukolsk S.A. ve Vetterling W.T., (1988), "Numerical Recipes in C, The Art of Scientific Computing”, Cambridge University Press, Cambridge, U.K.

Qin X., Baer J.L., (1997), "A Performance Evaluation of Cluster Architectures”, ACM SIGMETRICS, Proceedings of the International Conference on Measurement and Modeling of Computer Systems, Seattle, 237-247.

Rash W. Jr., Scovell S., Chee B., (2000), "The Leading Edge of Gigabit Ethernet”, The Networking Newspaper.

Reed D.A., Aydt R.A., Noe R.J., Roth P.C., Shields K.A., Schwartz B. ve Tavera L.F., (1993), "Scalable Performance Analysis, The Pablo Performance Analysis Environment”, Proceedings of the Scalable Parallel Libraries Conference, IEEE Computer Society.

Reed R. D. ve Marks R. J., (1999), “Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks”, MIT Press.

Reilly J., (1996), "A Brief Introduction to the SPEC CPU95 Benchmark”, IEEE -CS TCCA Newsletter.

Rose L.A. ve Reed D. A., (1999), "A Multi-Language Architecture-Independent Performance Analysis System”, Proceedings of the 28th International Conference on Parallel Processing.

Rumelhart D.E. ve Mcclelland J.L., (1986), "Parallel Distributed Processing: Explorations in the Microstructure of Cognition”, Vols I and II, MIT Press, Cambridge.

Rumelhart D.E., Widrow B., Lehr M.A., (1994), "Neural Networks: Applications in Industry, Business and Science”, Communications of the ACM (CACM) 37(3), 93-105.

Rumelhart D.E., Widrow B., Lehr M.A., (1994), "The Basic Ideas in Neural Networks”, Communications of the ACM (CACM) 37(3), 87-92.

Rumelhart. D.E., (1989), "The Architecture of Mind. A Connectionist Approach Foundations of Cognitive Science”, 133-159, The MIT Press.

Rumelhart, D.E., Hinton G.E. ve Williams, R. J., (1986), "Learning Internal Representations by Error Propagation”, Parallel Distributed Processing Vol. I, MIT Press, Cambridge, MA.

Saavedra R.H., Smith A.J. ve Miya E., (1989), "Machine Characterization Based on an Abstract High-Level Language Machine”, IEEE Transactions on Computers 38, 1659 – 1679.

Saavedra R.H., Smith A.J., (1996), "Analysis of Benchmark Characteristics and Benchmark Performance Prediction”, ACM Transactions on Computer Systems 14(4), 344-384.

Shavlik J.W., Dietterich T.G., (1990), "Readings in Machine Learning”, The Morgan Kaufmann Series in Machine Learning, Section 2.2.5 Learning Internal Representations by Error Propagation, by D.E. Rumelhart, G.E. Hinton, and R.J. Williams.

Siegel L.J., Siegel H.J. ve Swain P.H., (1982), "Performance Measures For Evaluating Algorithms for SIMD Machines”, IEEE Transactions on Software Engineering, SE-8(4), 319 – 330.

Sima D., (1997), "Superscalar Instruction Issue", IEEE Micro 17(5), 28-39.

Singh J.P., Weber W. ve Gupta A., (1991), "SPLASH: Stanford Parallel Applications for Shared-Memory", Technical report, Computer Systems Laboratory, Stanford University, CA.

Stallings W., (1988), "Tutorial - Local Network Technology", Third Edition, IEEE Computer Society Press.

van der Steen A. J., (1991), "The Benchmark of the EuroBen group", Parallel Computing, 1211-1221.

van der Steen, A. J. ve Dongarra, J., (2002), "Overview of High Performance Computers" In Handbook of Massive Data Sets, J. Abello, P. M. Pardalos, and M. G. Resende, Eds. Kluwer Academic Publishers, Norwell, MA, 791-852.

Taheri H.R., (1990), "An Analysis of the Neal Nelson Business Benchmark", Performance Evaluation Review 18(3), 13-18.

Tanenbaum A.S., (1990), "Structured Computer Organization", Third Edition, Prentice-Hall, Englewood Cliffs, NJ.

Tron C., Plateau B., (1994), "Modelling of Communication Contention in Multiprocessors", Lecture Notes in Computer Science, 794, 406-424.

McVoy L. ve Staelin C., (1996), "Imbench: Portable Tools for Performance Analysis", USENIX 1996 Annual Technical Conference, San Diego, CA.

Weaver D. ve Germond T., (1994), "The SPARC Architecture Manual, Version 9", Prentice Hall.

Weicker R.P., (1984), "Dhrystone A Synthetic Systems Programming Benchmark", Communications of the ACM 27(10), 1013 – 1030.

Worley P.H., (1992), "A New PICL Trace File Format", Oak Ridge National Lab, OR.NL/TM-12125, Oak R.idge, TN, USA.

Yan J., Sarukhai S. ve Mehra P., (1995), "Performance Measurement, Visualization and Modeling of Parallel and Distributed Programs Using the AIMS toolkit" Software - Practice and Experience, 25(4), 429-461.

Yavuz S., (2005a), "Performance Prediction of a 2-Dimensional FFT on Different Platforms Using Neural Networks", International Symposium on Innovations in Intelligent Systems and Applications, 136-140.

Yavuz S., (2005b), "Performance Prediction of a Parallel Monte Carlo Application: A Neural Network Approach", WSEAS Transactions on Circuits and Systems, 486-490.

Yavuz S., (2005c), "A Neural Network Based Methodology for Performance Evaluation of Parallel Systems", Lecture Notes in AI 3809, Springer-Verlag, 297-307.

Yavuz S., (2005d), "Paralel Sistemlerin Performansının Yapay Sinir Ağları ile Değerlendirilmesi", BMYS'2005 Bilimde Modern Yöntemler Sempozyumu, 44-52.

Zaki O., Lusk E., Gropp W., Swider D., (1999), "Toward Scalable Performance Visualization with Jumpshot", The International Journal of High Performance Computing Applications 13(3), 277—288.

INTERNET KAYNAKLARI

[1]www.aimtechnology.com

[2]www.sun.com

[3]www.pallas.com/e/products/index.htm

[4]www.myri.com/myrinet

[5]www.emulex.com/products - Giganet cLAN ürünleri

[6]standards.ieee.org/catalog/olis/busarch.html SCI standartı

[7]ftp://ftp.sas.com/pub/neural/FAQ.html

[8]gcc.gnu.org/onlinedocs

[9]www.gnu.org/software

[10]www.spec.org/osg/cpu2000

[11]www.netperf.org

EKLER

Ek 1. FFT ve Monte Carlo Uygulamalarına ait Komut Sayılarının Formülasyonu

IS: Image Size(Görüntü Boyutu)
FS: Filter Size (Filtre Boyutu)
NS: Number of Steps (Adım Sayısı)
NV: Number of Variates (Değişken Sayısı)
Nproc: Number of Processors (İşlemci Sayısı)

İşlem Türü	Sembol	FFT Uygulamasındaki İşlem Adedi
Bölme	DILL	$9+3*(IS-1)*[1/2*\log_2 IS]+8*[IS^2/Nproc]$
	DFDL	$6+6*[IS/Nproc]*\log_2 IS$
	DFSL	$3*[IS/Nproc]*\log_2 IS$
Çarpma	MFDL	6
	MILL	$12+3*[IS/Nproc]*[5+\log_2 IS+4*IS*\log_2 IS]+4*[IS^2/Nproc]$
	MILG	$3+2*IS+9/2*(IS-1)+3/2*[IS/Nproc]*(IS+3)+2*[IS^2/Nproc]$
Toplama	AILG	$3/2*[IS/Nproc]*(IS+3)+2*[IS^2/Nproc]$
	AILL	$6+21*[IS/Nproc]+3*(IS-1)*[2+1/2*\log_2 IS]+[IS^2/Nproc]*[2+\log_2 IS]$
	INLL	$9*IS-3+FS+FS^2+3/2*[IS/Nproc]*[IS+1+(IS+2)*\log_2 IS]+5*[IS^2/Nproc]$
Matematik Fonksiyon	SIND	$12+6*[IS/Nproc]*\log_2 IS$
	LOGD	12
	POWD	$3*[IS/Nproc]*\log_2 IS$
Lojik	CMLG	6
	CMLL	$3+12*IS+3*IS^2+FS+FS^2+[IS/Nproc]*[15/2+(6+3*IS)*\log_2 IS+3/2*IS]+(3/2*IS-1/2)*\log_2 IS+5*[IS^2/Nproc]$
	IFBR	$4+3*IS$
	ANDL	IS^2
Dizi	ARL1	$13+36*IS+IS^2+2FS^2+6*[IS/Nproc]*[3*IS+5+(2+8*IS)*\log_2 IS]+26*[IS^2/Nproc]$
	ARL2	$1+24*IS+IS^2+2*FS^2+3*[IS/Nproc]*[2+4*IS+6*IS*\log_2 IS]+26[IS^2/Nproc]$
Aktarma	TFSL	$3*[IS/Nproc]*[1/2+(2+IS/2)\log_2 IS]$
	TFDL	$[IS^2/Nproc]$
	TILL	$24+20*IS+3*(IS-1)*[7/2+\log_2 IS]+3*[IS/Nproc]*[3*IS+3+(2+11/2*IS)\log_2 IS]$
Saklama	SILL	$80+IS+IS^2+FS+2*FS^2+6*[IS/Nproc]*[1+\log_2 IS]$
Fonksiyon	FCAL	18
Döngü	LWHI	$3*(IS-1)$
	LFOR	$24+3/2*(IS-1)+3*[IS/Nproc]*[1+2*\log_2 IS]+IS$

İşlem Türü	Sembol	Monte Carlo Uygulamasındaki İşlem Adedi
Bölme	DFSL	$8 * nrproc * NS + 9$
	DFSG	5
	DFDL	$10 * nrproc * NS + 6$
Çarpma	MFSL	$nrproc * (23 * NS + 1) + 30$
	MFSG	$nrproc * [3 * NS + NV * (NV + 1)] + 5$
	MFDL	$42 * nrproc * NS + 43$
	MFDG	2
Toplama	AILL	$24 * nrproc * NS + 1$
	AILG	2
	AFSL	$nrproc * [4 + 26 * NS + NV * (NV + 2)] + 14 + [(NV^2 + NV + 3) * (Nproc - 1)]$
	AFSG	$nrproc * (4 * NS + 2) + 10$
	AFDL	$28 * nrproc * NS + 27$
	INLL	$nrproc * [26 * NS + 1 + NV * (NV + 2)] + 2 * NV + [(NV^2 + NV + 2) * (Nproc - 1)]$
Dallanma	BRTN	$28.5 * nrproc * NS + 2$
Lojik	CMFL	$3 * nrproc * NS$
	CMDL	$14 * nrproc * NS + 19$
	CMCG	$nrproc$
	IFBR	$nrproc * (25 * NS + 1) + 19$
Aktarma	TILL	$24 * nrproc * NS$
	TFSL	$nrproc * [9 + (40 * NS) + NV * (NV + 2)] + 44 + [(NV^2 + NV + 3) * (Nproc - 1)]$
	TFSG	$[nrproc * (NS + 2)] + 2$
Saklama	SILL	$2 * nrproc * NS$
	SFSL	$3 * nrproc * (NS + 1) + 3 * NV + 6$
	SFSG	$4 * nrproc + 1$
Matematik Fonksiyon	LOGD	$2 * nrproc * NS + 3$
	EXPD	$11 * nrproc * NS + 12$
	SQRD	$nrproc * NS + 2$
	ABSD	$3 * nrproc * NS + 4$
	POWD	1
Döngü	LFOR	$nrproc * [25 * NS + 2 + NV * (NV + 2)] + 2 * NV + [(NV^2 + NV + 2) * (Nproc - 1)]$
Fonksiyon	FCAL	$30 * nrproc * NS + 4$
	FARF	$11 * nrproc * NS + 4$
Dizi	ARFN	$35 + nrproc$
	ARF1	$nrproc * (4 + 3 * NV) + (2 * NV) + [2 * NV * (Nproc - 1)]$
	ARF2	$nrproc * [4 * (NS + 1) + NV * (NV + 4)] + NV + [2 * NV^2 * (Nproc - 1)]$
Diğer	ILFS	$2 * nrproc * NS + 4$
	FSFD	$66 * nrproc * NS + 73$
	FDIL	$2 * nrproc * NS$
	FDFS	$22 * nrproc * NS + 28$
	LTLG	$nrproc * [32 * NS + 2 + NV * (NV + 2)] + 2 * NV + [(NV^2 + NV + 2) * (Nproc - 1)]$
	LELG	$nrproc * NS$
	BALL	$24 * nrproc * NS$

ÖZGEÇMİŞ

Adı Soyadı:	Sırma Çekirdek Yavuz	
Doğum tarihi:	27.09.1969	
Doğum yeri:	Sarıkamış	
Lise:	1980 - 1986	Erenköy Kız Lisesi
Lisans:	1986 - 1990	Yıldız Teknik Üniversitesi Elektrik-Elektronik Fak. Elektrik Mühendisliği Bölümü
Yüksek Lisans:	1993 - 1995	Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı
Doktora:	1997 – 2000	The University Of Warwick Graduate School Computer Science Dept.
	2005 – 2006	Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı

Çalıştığı kurumlar

	1992 – Devam ediyor	Yıldız Teknik Üniversitesi Elektrik-Elektronik Fak. Bilgisayar Mühendisliği Bölümü