

Özörgütlemeli Yapay Sinir Ağı Modeli'nin Kullanıldığı Kutup Dengeleme Problemi için Paralel Hesaplama Tekniği ile Bir Başarım Eniyileştirme Yöntemi

Bahadır Karasulu, Aybars Uğur

Ege Üniversitesi, Bilgisayar Mühendisliği Bölümü, 35100, Bornova, İzmir.
bahadir.karasulu@ege.edu.tr, aybars.ugur@ege.edu.tr

Özet: Yapay sinir ağları (YSA), yoğun ilgi gösterilen bir araştırma alanı olup, birçok problemin çözülmesinde kullanılan modeller sunmaktadır. İlk olarak hareketli bir düzlem üzerinde dengede olmayan bir çubuğun (kutup) dengede tutulmasını sağlayan yarışmacı bir YSA modeli olan SOM (Özörgütlemeli Harita)'un Destekleyici Öğrenme (Reinforcement Learning) yaklaşımında eğitilmesi araştırılmıştır. Bu eğitime dayalı çözüm MPI (Mesaj Geçme Arayüzü) kullanılarak paralelleştirilmiştir. Uygun bir başarım eniyileme yöntemi geliştirilmiş ve elde edilen değerler yorumlanarak benzeri sistemlerle karşılaştırılmıştır.

Anahtar Sözcükler: Paralel hesaplama tekniği, Yapay sinir ağları, Özörgütlemeli harita, Kutup dengeleme problemi, Öğrenme algoritmaları.

A Performance Optimization Method With Parallel Computation Technique Using Self-Organizing Artificial Neural Network Model For Pole Balancing Problem

Abstract: Artificial neural networks (ANNs), which is an immensely interested research area, provides many models that can be used in the solution of many scientific problems. Firstly, it was searched to educate self-organizing map, SOM, which is a competitive ANN model and provides keeping a pole, an imbalanced stick found on a moving plane, in balance, on reinforcement learning approach. The solution based on this learning is made parallel using MPI (Message Passing Interface). An appropriate optimization method was developed and the calculations obtained using this method were interpreted and compared with similar systems.

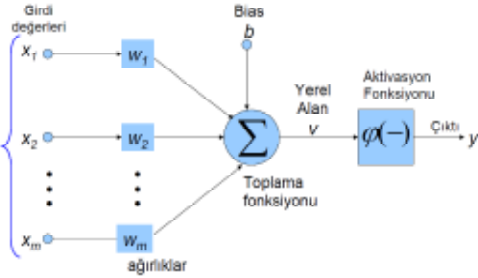
Keywords: Parallel computation technique, Artificial neural networks, Self-organizing map, Pole balancing problem, Learning algorithms.

1. Giriş

Yapay sinir ağları (Artificial Neural Networks), birbirine bağlı çok sayıda işlem elemanlarından oluşmuş, genellikle paralel işleyen yapılar olarak adlandırılabilir. İnsan beynindeki nöronların oluşturduğu ağ elemanları temel alınarak geliştirilmiş ve bilgisayarda uygulanmış bir yöntemdir. İnsan beyninin uyguladığı bazı organizasyonel prensipleri taklit ederler. Yapay sinir ağlarının temel birim işlem elemanı

ya da düğüm (node) olarak adlandırılan Şekil 1 'de görüldüğü gibi yapay bir sinir hücresidir. Yapay sinir ağları, ağırlıklandırılmış şekilde birbirlerine bağlanmış birçok işlem biriminden (nöronlar) oluşan matematiksel sistemlerdir. Bir işlem birimi, aslında sık sık transfer fonksiyonu olarak kullanılan bir denklemdir. Bu işlem birimi, diğer nöronlardan sinyalleri alır; bunları birleştirir, dönüştürür ve sayısal bir sonuç ortaya çıkartır. Sinirsel hesaplamanın merkezinde dağıtılmış, adaptif ve doğrusal olma-

yan işlem kavramları vardır. Yapay sinir ağları, geleneksel işlemcilerden farklı şekilde işlem yapmaktadırlar. Geleneksel işlemcilerde, tek bir merkezi işlem birimi her hareketi sırasıyla gerçekleştirir. Yapay sinir ağları ise herbiri büyük bir problemin bir parçası ile ilgilenen, çok sayıda basit işlem birimlerinden oluşmaktadır. Böylece paralelleştirmeye olan yatkınlıkları ortaya çıkmaktadır.



Şekil 1. Yapay sinir hücresi şeması.

Disiplinlerarası birçok problemin çözülmesinde kullanılan YSA için birbirinden farklı ağ mimarileri ve farklı eğitim algoritmaları geliştirilmiştir. Uygulamalarda önemli bir yer tutan Çok Katmanlı Algılayıcı (MLP) sinir ağları, birçok sezim ve kestirim işlemlerini yürütmek için kullanılmakta olan parametrik olmayan bir yapay sinir ağı modelidir. Bu konu üzerine yapılan bazı çalışmalara bakacak olursak; Destekleyici Öğrenme (Reinforcement Learning, DÖ) yönteminin kullanımında paralelleştirilmenin problemin çözümüne getirdiği iki anahtar katkı vardır. Birincisi, tek-etmenli öğrenme problemlerine daha çabuk ve iyi bir çözüm bulabilmektir. İkincisi ise etmenlerin aynı problem üzerinde bilgi değiş-tokuşu yoluyla çalıştığı (bir takım gibi) ortaklaşa (kooperatif) çoklu-etmen öğrenmesini mümkün kılan uygulamalar geliştirilmesinin sağlanmasıdır [1]. Çalışmamızda etmen yaklaşımının ikinci anahtar katkısında bahsedilen yapı kullanılmıştır. Ayrıca başka çalışmalarda da, kutup dengeleme problemleri (tek-kutuplu ve çok-kutuplu modeller, teleskopik kutup modelleri) ele alınmış olup [2], kazanan herşeyi alır yaklaşımı (winner-takes-all approach) uyarınca,

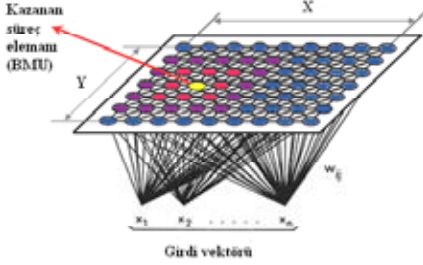
en kuvvetli aktivasyon ile ağıın çıktısını (örneğin sigmoidal çıktılar için çıktı sıfıra veya bire yakın olur) kullanma yaklaşımı denenmiştir [3]. Çalışmamızda da bu yöntem denenmiştir, zaten DÖ yapıldığından bir de Özörgütlemeli ağıın kendi yapısı gereği getirdiği bu yarışmacı yöntem iyi bir çözüm sağlamaktadır. Bu çalışmada Aktivasyon Fonksiyonu olarak normal dağılım (Gaussian) fonksiyonu seçilmiştir. Yukarıda bahsedilen diğer çalışmaların çoğunda doğrudan bir biçimde Kohonen SOM harita ve kazanan her şeyi alır mantığı kullanılmamakla beraber bazı yarışmacı yaklaşımlar ve DÖ yöntemleri kullanılmaktadır.

Çalışmada yazılım-tabanlı olarak Özörgütlemeli harita (Self-Organizing Map veya SOM) formundaki yapay sinir ağı modelleri için basit bir asimetrik paralelleştirme yöntemi geliştirilerek, var olan seri biçimde tasarlanmış SOM algoritmaları/programlarını hızlandıracak paralel uyarılma gerçekleştirilme çalışılmıştır. Ele alınan kutup dengeleme probleminde seri olarak kalan kısmın haricinde paralelleştirilebilecek olan kısım ağıın eğitimi ve komşuluk hesaplamaları göz önüne alınarak (öz olarak ağırlıkların ve yerleşimlerin ilgili süreç makineleri arasında transferleri ve bilgi paylaşımı) eniyileştirilmiştir. Böylece elde edilen yeni algoritma sayesinde tek işlemcili seri algoritma/programın başarımları artırılmıştır.

2. Özörgütlemeli Harita

Kohonen Özörgütlemeli harita (SOM) topolojikorunmalı bir haritadır. Bu harita yüksek boyutlu (üç veya daha fazla) bir haritadaki verileri, tipik bir iki boyutlu ızgara formundaki haritaya dönüştürür. Özörgütlemeli haritanın ana amacı, girdi uzayındaki komşuluk ilişkilerini mümkün olduğunca koruyan ve birimler arasındaki komşuluk ilişkilerine göre topoloji-korunmalı bir harita yaratmaktır [4]. Böyle bir Özörgütlemeli haritanın eğitiminde başlıca zaman tüketen adımlar verilen bir örnek için kazanan düğümün (winner node) yerleştirilmesi ile ilgili alt-problem boyunca geçen adımlardır [5]. Bir kazanan düğüm

her girdi vektörü için en iyi uyumlu birim (Best Matching Unit veya BMU) şeklinde ifade edilir. Bu durum Şekil 2’de gösterilmektedir.



Şekil 2. Kohonen Özörgütlemeli Haritası.

Bir örnekte en yakın komşuyu bulma problemi için kullanışlı çok sayıda yöntem vardır. En geçerli ve baskın karşılaştırma da, şablon vektörlerin durağan kalacağı varsayımı ile yapılmaktadır. SOM’in bu durumunda tüm döğümlerin ağırlıkları sabit aralıklarla güncellenmektedir.

Formal olarak Özörgütlemeli Harita tanımı:

Girdi vektörü,

$X = [x_1, x_2, \dots, x_n]^T \in R^n$ olsun. Bir i indisi ile düzenlenmiş birimlerin ayrık bir ızgarasını göz önüne alalım. Her döğüm ilgili ağırlık vektörü

$W_i = [w_{i1}, w_{i2}, \dots, w_{in}]^T \in R^n$ içermektedir. X burada, tüm ağırlık vektörleri içerisinde ağırlık vektörü onun en yakın komşusu olan birimi göstermektedir. Buna en iyi uyumlu birim (BMU) denilmektedir ve şu şekilde bulunmaktadır[6].

$$\|X - W_j\| = \min_i \|X - W_i\| \quad (1)$$

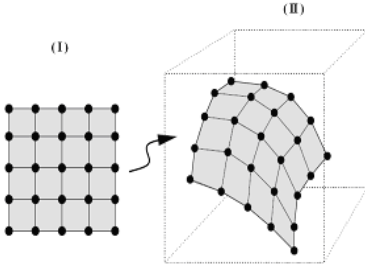
Özörgütlemeli ağın eğitimi için her iterasyon aşığıda özetlendiğı şekilde gerçekleşmektedir:

- Haritadaki döğümler arasından en yakın komşu (kazanan) her bir girdi örneğı için bulunur.
- Kazananın ve tüm komşularının ağırlıkları güncellenir.

En çok zaman harcanan kısım bu komşulukları bulurken geçen süredir. Komşuluk hesapları öklid mesafesi (uzaydaki iki nokta arasındaki mesafe) uyarınca hesaplanır. Özörgütlemeli harita temelde iki katmana sahiptir [7]. Giriş katmanı tamamıyla çift boyutlu Kohonen katmanına bağlanmıştır. Çıkış katmanı ise nicemeleme probleminde kullanılır ve giriş vektörünün ait olabileceğı üç sınıfı temsil eder. Bu çıkış katmanı tipik olarak delta kuralını uygulayarak öğrenir. Kohonen katmanı işlem elemanlarının her biri, gelen giriş değerlerinden onların ağırlıklarının öklid mesafesini ölçmektedir. Birimin ağırlık vektörü ile girdi vektörü arasındaki öklid mesafesi bu durumda aktivasyon fonksiyonu görevi görmektedir. Şekil 3’te görüleceğı gibi fiziksel uzayda iki boyutlu bir ızgara yapısı sergileyen SOM, Ağırlık/Girdi uzayında eğimli bir yapı sergilemektedir. Çalışmamızda elde ettiğimiz sonuçlarda da bu durumla karşılaşmaktadır. Özörgütlemeli haritada DÖ yanı sıra sıklıkla Yarışmacı Öğrenme (competitive learning) kullanılmaktadır [8]. İki tip yarışmacı öğrenme yaklaşımı vardır. Bunlar hard ve soft olarak adlandırılır. Hard yarışmacı öğrenmeye ait çevrimiçi-güncellemeli (online-update) yarışmacı öğrenme algoritması’na bakacak olursak;

1. $P(\xi)$ ‘ye göre rasgele seçilmiş $w_{ci} \in R^n$ olan referans vektörüne sahip A kümesine N tane c_i birimini içerecek şekilde ilk değer atamasını yap,
2. $P(\xi)$ ‘ye göre bir ξ giriş sinyali rasgele olarak yarat,
3. Kazananı $s = s(\xi)$ şeklinde tanımla,
4. Kazananın referans vektörünü ξ ile $\Delta w_{s_i} = a(x - w_{s_i})$ olacak şekilde uyarla,
5. Maksimum sayıda adıma (eğitim) ulaşıncaya kadar algoritmanın 2. adımı ile 5. adımı arasında tekrar devam et.

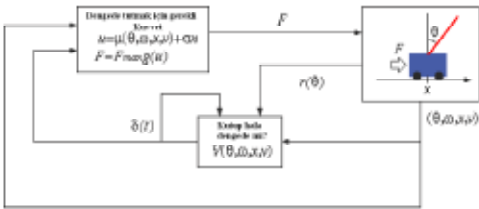
şeklinde olduğu görülür, aslında SOM için öklid mesafesi, (zamanla azalan) öğrenme oranı α ‘nın değişimini sağlamaktadır. Burada $P(\xi)$, sinyal fonksiyonu olarak alınmaktadır.



Şekil 3. (I) Fiziksel uzayda ve (II) Ağırlık/Girdi uzayında Kohonen haritası.

3. Kutup Dengeleme Problemi ve Çalışmamızdaki Kullanımı

Kutup dengeleme veya ters çevrilmiş sarkaç problemi uzun yıllardır yapay zeka ve yapay sinir ağları ile ilgilenen araştırmacıların ortak bir karşılaştırma (benchmark) aracı olmuştur [9]. Özörgütlemeli harita, öz olarak girdi ve çıktı uzaylarının arasında topoloji-korumalı formda haritalama yeteneği bulunan bir yarışmacı ağ'dır. Bu çalışmamızda bu yapay sinir ağı bir kutbu (burada örneğin 1 metre boyundaki bir çubuk gibi düşünebiliriz) temeline kuvvet uygulayarak dengede tutmayı öğrenmektedir. Kutbun davranışının Euler hareket yöntemine göre diferansiyel denklemlerin nümerik integrasyonu ile benzetimi yapılmıştır. Ağın görevi, kutbun durum değişkenleri ile kutbu dengede tutacak optimal kuvvet arasında bir haritalama kurulmasını sağlamaktır. Bu olay DÖ yaklaşımı ile gerçekleştirilir [10].



Şekil 4. Kutbu dengede tutmaya çalışan yapay sinir ağının çalışma şeması.

Kutbun herhangi bir verilen durumunda ağ haritalanmış kuvvetin zayıf bir değişimini dener.

Eğer yeni kuvvet daha iyi bir kontrol sonucu verirse, haritada değişiklik yapılır, böylece sistem kutbun geçerli durum değişkenlerini ve yeni kuvveti bir eğitim vektörü olarak kullanır. Şekil 4 'te görüleceği gibi $\delta(t)$ ve kutup puanı denilen kavram sayesinde karşılaştırma yapılabilmektedir [11].



Yukarıda gösterilen şemadaki gibi bir sistem iki boyutlu bir düzlem üzerinde hareket edebilen bir plakaya bağlanmış bir çubuk (kutup) için dengeleme şartları ve Euler hareket denklemleri uyarınca hareket ettirilmektedir [12]. Bu sistemdeki kutbun denge açısı y eksenini yaptığı açıdır. Yapay sinir ağı burada DÖ sırasında verilen değerler ve elde ettiği değerler ile öğrenmeyi gerçekleştirmektedir. Sisteme ait parametreler (q, w, x, v) algoritmanın / progra-

mın başlangıcında ilk değer olarak sıfır atanmaktadır. Daha sonra rasgele olarak yaratılan ω değeri ve diğer değerler ayarlanmaktadır. Çalışmamızda sistem çalıştırılarak öğrenmeye bırakıldıktan sonra Kutup Benzetimi sırasındaki adımlar hem gerçek çalışma süresi tutularak hem de benzetim adımı için verilen zaman adımları olan 0.1 saniye aralıklarla ölçülmüştür. Kutup Benzetimi sırasında ilgili birimin girdi vektörüne olan benzerliği bir puanlama sistemiyle kontrol edilmektedir. Buna bu çalışmada kutup puanı adı verilmektedir. Kutup dengeleme probleminde kullanılan bazı formüller ve parametreler hakkında bilgiler aşağıda yer almaktadır. Bu formül ve parametreler ilgili seri ve paralel program kaynak kodlarında kullanılmıştır. Tek bir plaka üzerinde tek bir kutbun dengelenmesi sırasındaki Euler hareket denklemleri şöyledir:

$$\ddot{\theta} = \frac{(F - M_p L (\dot{\theta}^2 \sin w - \dot{\theta} \cos w))}{M_c + M_p} \quad (2)$$

$$\dot{\theta} = \frac{\left(G \sin w + \cos w \left(\frac{-F - M_p L (\dot{\theta}^2 \sin w - \dot{\theta} \cos w)}{M_c + M_p} \right) \right)}{L \left(\frac{4}{3} - \frac{M_p \cos^2 w}{M_c + M_p} \right)} \quad (3)$$

Programda kullanılan parametreler Tablo 1’de gösterilmektedir.

Parametre ismi	Açıklama	Örnek
L	Kutbun (çubuğun) uzunluğu	1.0 metre
G	Yer çekimi ivmesi	9.81 m/sn ²
Mc	Plakanın (cart) kütlesi	2.0 kg
Mp	Kutbun (pole) kütlesi	1.0 kg
F	Uygulanan kuvvet	10 Newton
	İvme	m/sn ²
	Açısal ivme	rad/sn ²

Tablo 1. Paralel SOM Kutup Dengeleme programında kullanılan parametreler.

4. Kullanılan Paralel Hesaplama Tekniği ve Ortam

Özörgütlemeli harita’nın seri algoritmasında, eğitim adımlar ile ilerler. Bu durum algoritmanın kazananını bulmak için tüm birimleri taraması ve bulunca da tüm birimlerin değişimi yansıtması için günlemesi anlamına gelmektedir. Ağırlık vektörünün dizisi harita olarak alınır ve bu dizi iki kez dolaşılır. Bellek bantgenişliği sorununun özörgütlemeli haritanın yazılım-tabanlı uygulamalarında en büyük darboğazlardan birisi olduğu dikkate alınmalıdır. Algoritmada eğitimin her iterasyonunda harita üzerindeki her birime iki kez uğranılmaktadır. İlk etapta en iyi uyumlu birimi (BMU) bulmak için dolaşılıp sonra da değişiklik için tekrar dolaşılmaktadır. Bu gerçekten verimi düşüren bir olgudur. En basit olarak 25*25 lik bir haritadaki her birim için (ve dolaşımları da hesaba katacak olursak) gereken bellek miktarı ile eğitim ve hesaplamalar için geçen süre verimi düşüren bir durumu ortaya koymaktadır. Bu yüzden dağıtık ve dinamik bellekli ve bir çok işlemciye (sürece) destek verebilecek bir algoritmanın geliştirilmesi, verimi ve başarımı arttıracaktır. Bu noktadan hareketle geliştirme aşamasına geçildiğinde ilgili algoritmanın geliştirilmesinden sonra, böyle bir sistemin programlanması aşamasında mesaj geçme arayüzü (MPI) ile gerçekleştiriminin ve yapısal ANSI C dilinin (GNU C bedava dağıtımı) olanaklarının kul-

lanması tercih edilmiştir. Program geliştirilmesinde MPI paralel kütüphanesi olan LAM-MPI (Yerel Alan Çoklubilgisayarları – Mesaj Geçme Arayüzü) [13] kullanımı tercih edilerek ilgili SOM uygulaması gerçekleştirilmiş ve kutup dengeleme probleminin eğitim süreleri üzerinden başarımlar elde edilmiştir. Ayrıca yapılan çeşitli deneyler ve elde edilen sonuçlar bu çalışmada ayrıntıları ile ortaya konulmuştur.

5. Başarımların Eniyileştirme için Geliştirilen Paralel Hesaplama Yöntemi

Bu çalışmada kullanılan paralel hesaplama tekniği ilgili verinin birçok süreç makinesi (hesaplama kümesine dahil edilmiş fiziki makineler ve/veya işlemciler) arasında değiş-tokuş yöntemi ile paylaşılması ilkesine dayanmaktadır. Bu mantıkla bakılacak olursa, eğitim öncesi ve eğitim sonrası ve en iyi uyumlu birimin (BMU) aranması sırasında ilgili verilerin bir makineden diğer bir makineye gönderilip-alınması önündeki darboğazların aşılması gerekmektedir [13]. Bu sebeple yüksek hızlı anahtarlar kullanılarak deney laboratuvarı’nda (10/100 Mbps) sisteme etki edebilecek her türlü çevre etkeninin sisteme müdahalesi engellenmiştir. Çeşitli sayıda fiziki makine (256 MB RAM ve Intel Celeron 1.2 Ghz işlemci ve her işlemcinin 256 KB level 2 cache belleği’nin bulunduğu makineler) kullanılarak yapılan deneylerde sistemin başarımlar oranını makine sayısı artırıldıkça arttığı fakat bir noktadan sonra makine artımının başarı etkilemediği (Amdahl Kanunu’nun doğal bir sonucu olarak) görülmüştür. Tepe performansına erişildiği noktada en verimli öğreniminde geliştirilen algoritma tarafından gerçekleştirildiği gözlemlenmiştir. Bu en verimli öğrenim sırasında kutbun dengeye getirilmesinin öğrenilmesi süresinin kısalması eğitimin verimini bize göstermektedir [14]. Ayrıca deneyler sırasında çeşitli büyüklüklerde (deneyde 25*25, 125*125, 250*250, 500*500 büyüklüğünde) YSA’lar denenerek, sistemin gösterdiği tepkiler ve algoritmanın güvenilirliği sınanmıştır. Bu algoritmaya dayanan simülasyon programı gerçekleştirilmiş ve

deneyler çeşitli parametreler için yapılarak uygulama sonuçları elde edilmiştir. Simulasyon programının hem seri hem de paralel sürümleri bulunmaktadır. Geliştirilen paralel hesap yöntemine ait algoritmik akış aşağıdaki gibidir:

1. Başla,
2. İlgili dizi ve değişkenleri tanımla,
3. Gerekli dizi elemanları ve/veya değişkenlerin değerlerini sıfırla,
4. MPI'ı başlat,
5. İşlemci sayısı ve kimliklerini tespit et,
6. Euler Hareket denklemlerinde kullanılacak ilgili parametreleri dosyalardan oku,
7. Hesaplama geçen sürenin tespiti için saat tutmaya başla,
8. AG isimli yapı bloğundan yeni bir DenekAgi isimli Kohonen SOM ağı oluştur,
9. SansSayilariniOlustur() ve RasgeleAgirliklar() isimli prosedürler ile rasgele sayı üretici ile rasgele ağırlıkları oluştur,
10. Ağın eğitimini başlat,
11. Ağın eğitimi sırasında ilgili girdi ve çıktı verilerini ve ağırlıkları işlemciler üzerine MPI_Send() fonksiyonu ile dağıt,
12. Ağın eğitimi boyunca ilgili hesaplamalar işlemcilere eşit miktarda (Formül = [Toplam YSA birimi (düğüm sayısı) / toplam işlemci sayısı * (geçerli işlemcinin sırası +1)]) dağıtılması yoluyla uygun hesaplamayı yaptır,
13. Elde edilen yerel sonuçları MPI_Recv() fonksiyonu ile işlemci sırası (rank) sıfır olan (yani ilk makine veya yönetici makine olarak adlandırılır) üzerinde toparla,
14. Adım 11 ile 13 arasındaki algoritma adımlarını eğitim adımları bitene kadar tekrarla,
15. Her adımda elde edilmiş olan Simulasyon zaman adımı değeri (0.1 saniyelik adımlar), o adımda kutup çubuğunun y eksenini ile arasındaki açı (kutbun hala dengeli olup olmadığının tespiti için) ve uygulanan optimum kuvveti içeren 3 kolonluk bilgiyi ilgili dosyaya yazdır,
16. Saat tutmayı bitir.
17. Ağın en son halini ilgili dosyaya yazdırarak hesap kısmını sonlandır.
18. MPI'ı bitir.
19. Dur.

6. Elde edilen deney sonuçları

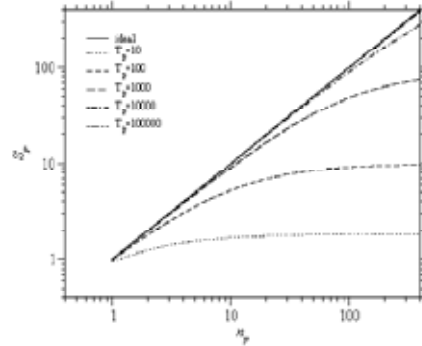
Bu çalışma boyunca yapılan deneyler sırasında elde edilen hızlanma oranları (başarım) ve verim yüzdeleri aşağıdaki tablolar ve grafiklerde verilmektedir. Deney için kullanılan parametreler de tablolar halinde verilmektedir. Hızlanma deyimini (S_p) için Amdahl Kanunu formülü :

$$S_p(n_p) = \frac{T(1)}{T(n_p)} \quad (4)$$

Burada n_p , işlemci sayısıdır, $T(1)$, tek işlemci çalıştırıldığında geçen süre, $T(n_p)$ ise n_p adet işlemci çalıştırıldığında geçen süredir. Verim için kullanılan formül ise:

$$e = \frac{T(1)}{n_p T(n_p)} \quad (5)$$

şeklinde. Amdahl Kanunu'na göre Şekil 5'te hızlanma grafiği gösterilmektedir.



Şekil 5. Amdahl Kanunu grafiği.

Burada $T(n_p) = 10, 100, 1000, 10000$ ve 100000 için hızlanmalar görülmektedir.

Tek ve Çok işlemcili simulasyon programında kullanılan parametreler Tablo 2'de, Seri hesap (yani $n_p=1$) iken ve Paralel hesapta ($n_p=2$ veya $n_p=3$ veya $n_p=4$) iken geçen süreler, bu süreler için hızlanma değerleri ve bu hızlanma değerlerine ait verim değerlerini Tablo 3'te gösterilmektedir. Burada n_p , işlemci sayısını göstermektedir.

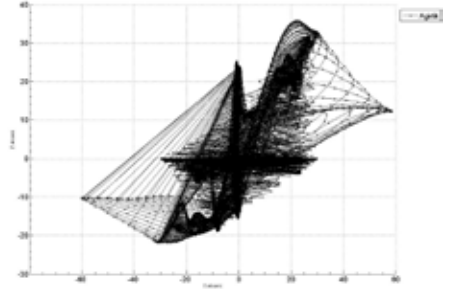
Açıklama	Ağ 25*	Ağ 125*	Ağ 250*	Ağ 500*
SOM Satır Sayısı	25	125	250	500
SOM Sutun sayısı	25	125	250	500
Eğitim adımları	100000	100000	100000	100000
Dengeleme Süresi (sn)	240	240	240	240
Benzetim Adım Sayısı (İterasyon)	1000	1000	1000	1000
Kutbun Denge Açısı (y_ekseni ile)	75	75	75	75
Kutup Çubuğu uzunluğu (metre)	3.0	3.0	3.0	3.0
Plakanın Ağırlığı (kg)	2.0	2.0	2.0	2.0
Kutbun Ağırlığı (kg)	1.0	1.0	1.0	1.0
Yerçekimi	9.81	9.81	9.81	9.81
Benzetim Zaman adımları	0.1	0.1	0.1	0.1

Tablo 2. Simulasyon için kullanılan parametreler.

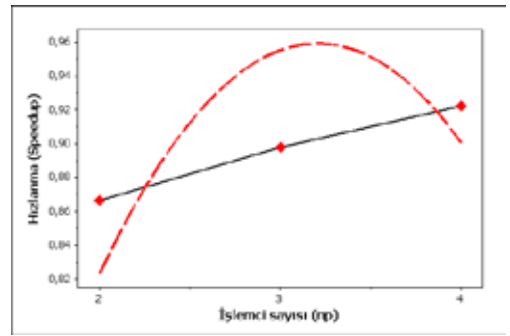
Açıklama	İşlemci Sayısı (n _p)	Ağ (25*) 25)	Ağ (125*) 125)	Ağ (250*) 250)	Ağ (500*) 500)
Geçen Süre (sn)	1	215	4545	17254	72543
	2	245	5370	18816	83732
	3	271	5564	19507	80767
	4	290	5337	19658	78640
Hızlanma Oranı	1	1	1	1	1
	2	0.8775	0.8463	0.9169	0.8663
	3	0.7962	0.8168	0.9053	0.8981
	4	0.7413	0.8516	0.8777	0.9224
Verimlilik (% olarak)	1	100	100	100	100
	2	43	42	45	43
	3	26	27	30	29
	4	18	21	21	23

Tablo 3. Simulasyonun Tek ve Çok-ışlemcili sistemde çalıştırılması sonucu elde edilen geçen süre değerleri, hızlanma ve verim tablosu.

Ayrıca Ağırlık/Girdi uzayında üzerinde çalıştığımız Kohonen Özgütlemeli Haritası'nın nasıl bir görünüme sahip olduğu Şekil 6 'da gösterilmektedir. Ele aldığımız bu sistemin örnek olarak 500*500'lük Kohonen ağına ait hızlanma grafiği Şekil 7' de verilmektedir.



Şekil 6. 125*125'lik Ağ'a ait ağırlıkların X ve Z ekseninde gösterimi (bu grafiğe ağırlık/girdi uzayında Kohonen SOM haritası da denilmektedir).



Şekil 7. 500*500'lik Ağ'a ait hızlanma oranı grafiği. (Grafikteki düz çizgi ile gösterilen eğri gerçek hızlanma değerlerini, kesikli çizgilerle gösterilen eğri ise 'eğri uydurma' ile elde edilen hızlanma grafiğini gösterir).

Şekil 7'de 4 işlemci ile çalıştırılan paralel programın hızlanma grafiğinde tepe performansına 3 işlemcili durumda ulaştığı görülmektedir.

7. Sonuçlar

Çalışmada seri hesabın yanı sıra paralel hesap yapılması sonucu belirli bir hızlanma elde edilmiştir. Fakat bu hızlanma istenilen düzeyde olmamıştır. Bunun başlıca nedenleri problemin doğası gereği fazlaca paralelleştirmeye uygun olmamasıdır. Tepe performansı 3 makinenin aynı anda çalıştığı durumda hızlanma değeri olarak 0.9 civarında elde edilmiştir. Benzeri çalışmalarda da bire bir kutup dengeleme probleminin paralelleştirilmesi

fikri üzerinde çalışıldığı halde etkin olarak tepe performansının yakalanamadığı görülmektedir. Bu yüzden etkin bir algoritmanın ve SOM harita yapısının kutup dengeleme gibi temel bir kıyaslama aracı üzerinde denenmesi fikri oldukça özgün durmaktadır. Konu üzerinde yapılabilecek daha derin çalışmalarda öncelikle paralelleştirme yüzdesinin artırılması için DÖ yöntemine getirilecek değişik yorumlara veya farklı öğrenme yöntemlerinin de denenmesine ihtiyaç duyulmaktadır [15]. Eğer paralelleştirme yüzdesi artırılamazsa problemin çözümü etkinliği arttırmayacak ve bir ilerleme kaydedilemeyecektir. Bunun için temel öneri olarak SOM ve LVQ (Learning Vector Quantization) tarzı ağ yapılarının kullanıldığı problemlerde Kohonen katmanı için ayrıca bir paralel hesaplama tekniği geliştirilmesi, girdi ve çıktı katmanlarının farklı bir yaklaşımla ele alınması daha uygun olacaktır.

8. Kaynaklar

- [1] Grounds M., Kudenko D., “Parallel Reinforcement Learning by Merging Function Approximations”, Department of Computer Science University of York, (2006).
- [2] Gomez F. and Miikkulainen R., “2-d pole balancing with recurrent evolutionary networks.”, In Proc. of the Int. Conf. on Artificial Neural Networks (ICANN-98), Skovde, Sweden, pp:425–430, (1998).
- [3] Pardoe D., Ryoo M., Miikkulainen R., “Evolving Neural Network Ensembles for Control Problems”, In Proc. of the Genetic and Evolutionary Computation Conference (GECCO-2005), (2005).
- [4] Kohonen T., “Self Organization Maps.”, Springer, Berlin, 1995.
- [5] Kohonen, T., “The speedy SOM. Technical Report A33”, Helsinki University of Technology, Laboratory of Computer and Information Science, Espoo, Finland, 1996.
- [6] Vishwanathan S. V. N., Murty M. N., “Kohonen’s SOM with cache”, Pattern Recognition, 33(11):1927–1929 (2000).
- [7] Rauber A., Tomsich P., Merkl D., “parSOM: A Parallel Implementation of the Self-Organizing Map Exploiting Cache Effects: Making the SOM Fit for Interactive High-Performance Data Analysis”, HPCN Europe, (2000).
- [8] Koikkalainen P. and Oja E., “Self-organizing hierarchical feature maps”, In Proceedings of the Int. Joint Conf. on Neural networks (IJCNN), Vol. II:279–285, Piscataway, NJ, IEEE Service Center, (1990).
- [9] O. Stanley K., Miikkulainen R., “Efficient Reinforcement Learning through Evolving Neural Network Topologies”, In Proceedings of the Genetic and Evolutionary Comp. Conf. (GECCO-2002), (2002).
- [10] Sutton, R. S., and Barto, A. G., “Reinforcement learning: An introduction”, Cambridge, MA: MIT Press, (1998).
- [11] Sutton, R.S., “Reinforcement learning architectures”. In Proc. ISKIT’92 Int. Sym. on Neural Information Processing, Fukuoka, Japan, (1992).
- [12] Sutton, R.S., “Reinforcement learning architectures for animats”, In Proc. of the First Int. Conf. on Simulation of Adaptive Behavior: From Animals to Animats, pp. 288-296, (1991).
- [13] LAM-MPI takımı websitesi. (Çevrimiçi : <http://www.lam-mpi.org>).
- [14] Kretchmar, R.M., “Parallel reinforcement learning”, In Proc. of the 6th W Conf. on Systemics, Cybernetics, and Informatics (SCI2002), (2002).
- [15] Vassilas, N. Thiran, P. and Jenne P., (1996), “On modifications of Kohonen’s feature map algorithm for an efficient parallel implementation”, In Proceedings of ICNN96, pp:932-937, (1996).