Avrupa Bilim ve Teknoloji Dergisi
Özel Sayı 26, S. 262-269, Temmuz 2021
© Telif hakkı EJOSAT'a aittir
Araştırma Makalesi

www.ejosat.com ISSN:2148-2683

European Journal of Science and Technology
Special Issue 26, pp. 262-269, July 2021
Copyright © 2021 EJOSAT
Research Article

Çok Etmenli Sistemlerde Bir Dağıtık Denklem Çözüm Algoritmasının Yakınsama Hızı En İyilemesi

Onur Cihan

Marmara Üniversitesi, Mühendislik Fakültesi, Elektrik-Elektronik Mühendisliği Bölümü, İstanbul, Türkiye (ORCID: 0000-0002-5729-2417), onur.cihan@marmara.edu.tr

(3rd International Congress on Human-Computer Interaction, Optimization and Robotic Applications June 11-13, 2021)

(**DOI:** 10.31590/ejosat.952456)

ATIF/REFERENCE: Cihan, O. (2021). Çok Etmenli Sistemlerde Bir Dağıtık Denklem Çözüm Algoritmasının Yakınsama Hızı En İyilemesi. *Avrupa Bilim ve Teknoloji Dergisi*, (26), 262-269.

Öz

Bu çalışmada, çok etmenli bir ağ üzerinde tanımlanan ve doğrusal denklem sistemlerini çözmek için önerilen bir algoritmanın yakınsama hızının en iyilenmesi problemi ele alınmıştır. Sistemde bulunan her bir etmen, doğrusal denklem sisteminin yalnızca bir alt kümesini bilmekte; bu yerel denklem bilgisi ve komşu etmenlerin çözüm tahminlerini kullanarak kendi tahminlerini güncellemekte ve denklem sisteminin eşsiz çözümüne ulaşmayı amaçlamaktadırlar. Etmenlerin çözüm hatası dinamikleri bir doğrusal dinamik sistem olarak ifade edilerek yakınsama hızını en iyileme problemi, sistem matrisinin en büyük özdeğerini en küçükleme problemi olacak şekilde formüle edilmiştir. Literatürde yakın zamanda önerilmiş ve metasezgisel bir optimizasyon algoritması olan Aritmetik Optimizasyon Algoritması kullanılarak örnek bir denklem sisteminin en hızlı çözümünü sağlayan tasarım parametreleri belirlenmiştir. Arama ajanı ve yineleme sayılarının elde edilen en iyi hızlı yakınsama değerine etkisini incelemek amacıyla benzetim çalışmaları farklı sayıda arama ajanı ve maksimum yineleme sayısı için tekrarlanmış ve sonuçlar tablo halinde sunulmuştur. Elde edilen en hızlı yakınsamayı sağlayan tasarım parametreleri için etmenlerin çözüm tahminlerinin zamana bağlı değişimi verilmiştir.

Anahtar Kelimeler: En iyileme, Dağıtık algoritmalar, Çok etmenli sistemler.

Convergence Rate Optimization of a Distributed Algorithm for Solving Linear Equations Over Multi-Agent Networks

Abstract

In this study, we investigate the problem of convergence rate optimization of an algorithm for solving linear equations over multi-agent systems. Each agent in the system is assumed to know only a subset of the linear equation system, and by using its local equation and the solution estimates of the neighboring agents, each agent updates its estimate and aims to compute the unique solution of the linear equation. We express the error dynamics of the agents as a linear dynamical system and relate the convergence speed of the optimization problem as the problem of minimizing the largest eigenvalue of the system matrix. By using a recently proposed metaheuristic optimization algorithm, Arithmetic Optimization Algorithm, the design parameters that provide the fastest solution have been determined. To examine the effect of the numbers of search agents and iterations on the performance of the optimization algorithm, simulation studies were repeated for different numbers of search agents and iterations. The results of the simulations are presented in a table. The evolution of the solution estimates of the agents is given for the design parameters that provide the fastest convergence.

Keywords: Optimization, Distributed Algorithms, Multi-agent systems.

1. Giriş

$Ax = b$ biçiminde temsil edilen bir doğrusal denklem sistemini çözme problemi çok uzun bir geçmişe sahiptir (Wang vd., 2019). Jacobi yöntemi, Gauss-Seidel yöntemi ve Kaczmarz yöntemi (Hackbusch, 1994) gibi birçok klasik algoritma başarıyla geliştirilmiş olup genellikle $Ax = b$'yi merkezi bir şekilde çözmek amaçlanmıştır. Bu nedenle bunlara merkezi algoritmalar denir. Bununla birlikte, merkezi algoritmaların $Ax = b$'yi çok sayıda bilinmeyen değişkenle çözmesi çok zordur ve bu durum genellikle kısmi türevli diferansiyel denklemler (Krstic ve Smyshlyaev, 2008), hesaplamalı akışkanlar dinamiği (Anderson, 1995), büyük ölçekli doğrusal regresyon (Frank, Fabregat-Traver ve Bientinesi, 2016), elektromanyetizma hesaplamaları (Carpentieri, Duff, Giraud ve Magolu monga Made, 2004), ışık saçılım hesaplamaları (Rahola, 1996), arama motorları için PageRank algoritmaları (Silvestre, Hespanha ve Silvestre, 2018) gibi gerçek dünya problemlerinde ortaya çıkmaktadır. Büyük ölçekli doğrusal denklem sistemlerini içeren bu uygulamalar için, gerekli tüm hesaplamalar tek bir yerde gerçekleştirilmek istendiğinde, yüksek hesaplama kaynağına gereksinim vardır. Öte yandan, büyük ölçekli karmaşık ağ destekli sistemleri içeren uygulamalar için (örneğin, güç sistemleri), merkezi algoritmaların diğer alt sistemlerden denklemler hakkında bilgileri toplaması da gereklidir (Wang, Ren vd., 2018). Bu durumda merkezi algoritmalar, sınırlı bant genişliğinin getirdiği sorunlara çözüm getirmek durumundadır.

Dağıtık algoritmalar, merkezileştirilmiş algoritmaların aksine, çok etmenli bir sistemde çok sayıda bilinmeyen değişkenle $Ax = b$ 'yi çözmede hesaplama zorluklarının üstesinden gelmek için umut verici ve uygulanabilir bir alternatif sunar. İlk olarak, büyük doğrusal denklem sistemini, genellikle birbirine bağlanan birçok küçük doğrusal alt sisteme ayırırlar. Daha sonra, orijinal doğrusal denklem sisteminin çözümü, bireysel aracıların doğrusal alt sistemleri çözmek için koordine edilmesiyle elde edilir. Örneğin, You, Song ve Tempo (2016) 'da gösterildiği gibi, önce $Ax = b$ sistemi satırlara ayrılabilir ve ardından dağıtık bir şekilde çözülebilir. Dağıtık hesaplama ile ağır hesaplama yükü, hesaplama verimliliğine ulaşmak için farklı hesaplama birimleri arasında bölünecek ve dağıtılacaktır. Ağ destekli sistemleri içeren uygulamalar için, doğrusal denklem sistemi genellikle doğal olarak birçok küçük ölçekli doğrusal denklem alt sisteminden oluşur. Bu durumda, doğrusal denklemlerin tek tek alt sistemlerini büyük bir sistemle birleştirmek yerine, ağa bağlı sistemlerin dağıtık doğasını tam olarak kullanmak ve çözümü elde etmek için dağıtık hesaplamayı uygulamak daha mantıklıdır. Bu aynı zamanda alt sistemlerden büyük miktarda bilgi toplamak için iletişim üzerindeki yükü büyük ölçüde azaltacaktır. Örneğin, büyük ölçekli bir güç sisteminde DC güç akışı problemini çözmek için (Stott, Jardim ve Alsac, 2009) bireysel yük veriyollarında hem aktif hem de reaktif gücü ölçmek gerekir. Bu ölçümler aslında güç akış dengesini temsil eden doğrusal denklem sistemi için farklı satırlara katkıda bulunur. Bu durumda, tüm ölçüm verilerinin merkezi bir yere iletilmesini önlemek için dağıtık algoritmaların benimsenmesi daha uygun olabilir. Bunun yanısıra, literatürde dağıtık çözüm sağlayan algoritmaların kullanılmasının güvenlik ve gizlilik avantajlarının olduğu belirtilmişse bile, bu durumun her zaman geçerli olmadığı konusunda çalışmalar da mevcuttur (Cihan, 2019).

Yukarıda belirtilen birçok avantaj nedeniyle $Ax = b$ 'yi çözmek için dağıtık algoritmaların geliştirilmesi, birçok araştırmacının önemli ilgisini çekmiştir ve son zamanlarda bireysel etmenler arasında bilgi paylaşımı miktarına ilişkin farklı gereksinimlerle birçok yeni algoritma önerilmiştir. Örneğin Pasqualetti, Carli ve Bullo (2012), güç sistemi tahmin problemleri için $Ax = b$ sistemini çözen dağıtık bir algoritma tasarlamıştır. Bu algoritmada, bireysel etmenlerin durum tahminlerinin yanı sıra bazı diğer matris bilgilerinin paylaşılması gerkemektedir. Daha sonra aynı problem için Mou, Liu ve Morse (2015), yalnızca bireysel ajanların durum tahminlerinin paylaşılmasını gerektiren ve Pasqualetti, Carli ve Bullo (2012)'nun geliştirdiği algoritmadan daha gürbüz olan bir dağıtık algoritma geliştirmiştir. Bu çalışmalardan ilham alarak, ayrık zamanlı ve sürekli zamanlı birçok dağıtık algoritma geliştirilmiştir (Anderson vd., 2016; Wang, Fullmer ve Morse, 2016; Wang, Ren ve Duan, 2018). Bu algoritmaların arkasındaki ana fikir, oydaşım ilkesidir (Mou vd., 2015). Oydaşım ilkesinde, çok etmenli sistemde bulunan her bir etmen komşularının durum değerlerini kullanarak aynı durum değeri üzerinde uzlaşmaya çalışırlar (Cihan ve Akar 2020a, Cihan ve Akar 2020b, Cihan 2020). Etmenler çözüm tahminlerini $Ax = b$ denklemi için çözüm tahminlerini güncellerken yalnızca kendi doğrusal denklem alt sistemlerini sağlamakla kalmamakta, aynı zamanda komşu etmenlerle çözüm konusunda bir fikir birliğine varmayı amaçlamaktadırlar. Böylece etmenlerin tamamı aynı çözüm üzerinde oydaşımı sağlarken, yerel denklemlerin tamamının sağlanması nedeniyle denklem sisteminin çözümüne ulaşmış olurlar.

Dağıtık algoritmaların büyük avantajlarına rağmen, yapılan benzetim çalışmalarında yakınsama hızlarının tatmin edici seviyede olamadıkları bilinmektedir (Cihan, 2019). Denklem sisteminin çözüm hızının artırılması için önerilen algoritmaların katsayılarının yakınsama hızını maksimum olacak şekilde seçilmesi uygun bir yöntemdir. Bu çalışmada, dağıtık denklem sistemlerinin çözümü için literatürde önerilmiş bir algoritmanın parametrelerinin, yakın zamanda önerilmiş bir metasezgisel optimizasyon yöntemi olan Aritmetik Optimizaston Algoritması (Abualigah vd., 2020) kullanılarak yakınsama hızını maksimum yapacak şekilde belirlenmesi problemi ele alınmıştır.

Çalışmanın geri kalanı şu şekilde organize edilmiştir. Bölüm 2'de çok etmenli bir sistemi temsil etmekte kullanılan çizge kuramı hakkında bilgiler verilerek, literatürde doğrusal denklem sistemlerinin çözümünde kullanılmak üzere önerilmiş olan bir algoritma tanıtılmıştır. Bölüm 3'te algoritmanın yakınsama hızının en iyilenmesi için kurulan optimizasyon problemi ve çözümünde kullanılacak olan Aritmetik Optimizasyon Algoritması tanıtılmıştır. Bölüm 4'te örnek bir denklem sistemi ve çok etmenli sistem için en hızlı yakınsamayı sağlayan sistem parametreleri aritmetik optimizasyon algoritması ile belirlenerek benzetim sonuçları verilmiştir. Elde edilen sonuçlar ve gelecek çalışmalar hakkında bilgiler ise Bölüm 5'te sunulmuştur.

2. Çizge Kuramı ve Matematiksel Formülasyon

Bu bölümde çizge kuramı ve doğrusal denklem sistemlerinin çözümü için kullanılan bir dağıtık algoritmanın matematiksel modeli verilecektir.

2.1. Çizge Kuramı

Çok etmenli bir ağdaki bilgi akışı, yönlendirilmiş bir çizge olan $G = (V, E)$ ile temsil edilir. Burada $V = \{v_1, \dots, v_n\}$, çizgenin köşe kümesi ve $E \subseteq V \times V$, etmenler arası iletişimi temsil eden sonlu bir kenar kümesidir. (v_i, v_j) ile gösterilen bir kenar, v_i'den v_j'ye yönlendirilmiş bir bilgi akışını gösterir, yani v_j bir adımda v_i'den bilgi alabilmektedir. v_i köşesinin komşular kümesi $N_i = \{j | (v_j, v_i) \in E\}$ olarak tanımlanır.

2.2. Matematiksel Formülasyon

$G = (V, E)$ ile temsil edilen ve n etmenden oluşan bir çok etmenli sistemde $i.$ etmen tarafından bilinen denklem sisteminin $A_i\, x = b_i$ olarak ifade edilebildiği durumda, Morse vd. (2015) tarafından önerilmiş olan ayrık zamanda tanımlanmış aşağıdaki algoritmayı ele alalım.

$$x_i(t+1) \;=\; x_i(t) \;-\; P_i \left(x_i(t) - \sum_{j \in N_i} w_{ij} x_j(t) \right), \qquad i = 1, \dots, n. \tag{1}$$

Burada $x_i(t) \in R^m$ vektörü $i.$ etmenin $t.$ adımdaki çözümünü ifade eder ve her bir etmenler başlangıç koşulunu yerel denklemini sağlayacak şekilde seçer. P_i matrisi, A_i matrisinin sıfır uzayına izdüşüm matrisi olup $P_i = I - A_i^T (A_i A_i^T)^{-1} A_i$ olarak tanımlanır. w_{ij} ağırlıklandırma katsayıları ise algoritmanın tasarım parametreleridir ve aşağıdaki varsayımı sağlar.

Varsayım 1

i) Eğer $(v_j, v_i) \in E$ ise, $w_{ij} \geq 0$ dır.
ii) Eğer $(v_j, v_i) \notin E$ ise, $w_{ij} = 0$ dır.
iii) Her $i \in \{1, \dots n\}$ için $\sum_{j=1}^{n} w_{ij} = 1$ dir.

Varsayım 1(i)'ye göre eğer iki etmen arasında bir iletişim mevcutsa çözüm algoritmasında bunun negatif olmayan katsayılarla kullanılması gerektiği, (ii)'ye göre aralarında iletişim olmayan etmenlerin birbirlerinin çözümünü kullanamayacağı belirtilmektedir. Varsayım 1(iii) ile her etmenin kullandığı ağırlık katsayılarının toplamının 1'e eşit olması gerektiği ifade edilmekte olup, algoritmanın kararlılığı için gerekli bir şarttır.

$Ax = b$ şeklinde ifade edilen doğrusal denklem sisteminin bir çözümü x^* ile ifade ediliyor olsun: $A\, x^* = b$. Bu durumda, x^* çözümü tüm etmenlerin denklemlerini sağlamalıdır, yani her i için $A_i x^* = b_i$ dir. Ağdaki $i.$ etmenin çözüm hatası vektörü

$$e_i(t) \;=\; x_i(t) \;-\; x^* \tag{2}$$

olarak ifade edilebilir. Burada her t için $e_i(t) \in null(A_i)$ dir. Bu nedenle $P_i e_i(t) = e_i(t)$ denklemi yazılabilir. Denklem 1 kullanılarak $i.$ etmenin hata dinamikleri

$$e_i(t+1) \;=\; P_i^2\, e_i(t) \;-\; P_i \left(P_i\, e_i(t) - \sum_{j \in N_i} w_{ij} P_j e_j(t) \right) \tag{3}$$

ya da daha basit bir şekilde

$$e_i(t+1) \;=\; w_{ij} P_i \sum_{j \in N_i} P_j e_j(t) \tag{4}$$

olarak elde edilebilir (Mou v.d., 2015). Tüm etmenlerin hata vektörlerini içeren genişletilmiş hata vektörü $e(t) = [e_1(t)^T, \dots, e_n(t)^T]^T$ olarak tanımlandığında hata dinamikleri matris formunda

$$e(t+1) \;=\; P(W \otimes I) P\, e(t) \tag{5}$$

olarak ifade edilir. Burada $P = diag\{P_1, \dots, P_n\}$ matrisi A_i'nin sıfır uzayına izdüşüm matrislerini içeren bir blok köşegen matris ve $W = [w_{ij}]$ ise katsayılar matrisidir. Denklem sisteminin çözülmesi için tüm etmenlerin hata vektörlerinin sıfır vektörüne yakınsaması gerektiği için, Denklem 5 ile ifade edilen doğrusal dinamik sisteminin kararlı olması gerekir. Bu sistemin kararlı olması için $P(W \otimes I)P$ matrisinin tüm özdeğerleri 1'den küçük olmalıdır ve bunun için Varsayım 1'in sağlanması gerekmektedir (Mou v.d., 2015).

3. Yakınsama Hızı Analizi

Bu bölümde doğrusal denklem sistemlerinin çözümü için analiz edilen algoritmanın yakınsama hızının en iyilenmesi konusunda yapılan çalışmalar açıklanmıştır.

3.1. En İyileme Problemi Formülasyonu

Denklem 1 ile verilen ve kararlı olduğu bilinen bir sistemin yakınsama hızı, $P(W \otimes I)P$ matrisinin mutlak değeri en büyük olan özdeğeri ile ilişkilidir. En hızlı yakınsamanın sağlanabilmesi için W matrisinin elemanları tasarım parametreleri olarak uygun şekilde seçilerek $P(W \otimes I)P$ matrisinin en büyük özdeğerini en küçükleyecek bir en iyileme problemi aşağıdaki gibi formüle edilebilir.

$$\begin{aligned} \min \quad & \lambda_{\max}\,(\mathrm{P}(\mathrm{W} \otimes \mathrm{I})\mathrm{P}) \\ \text{kısıtlar} \quad & \sum_{j=1}^{n} w_{ij} = 1 \quad \text{her } i = 1, \dots, n \text{ için} \\ & w_{ij} = 0 \quad \text{eğer } (v_j, v_i) \notin E \text{ ise} \\ & w_{ij} \geq 0 \quad \text{eğer } (v_j, v_i) \in E \text{ ise} \end{aligned} \tag{6}$$

3.2. Aritmetik Optimizasyon Algoritması

Abualigah v.d. (2020) tarafından geliştirilen Aritmetik Optimizasyon Algoritması (AOA), dört işlem olarak adlandırılan çarpma, bölme, çıkarma ve toplama işlemlerinin dağıtım davranışını kullanan yeni bir metasezgisel yöntemdir. AOA, çok çeşitli arama alanlarında optimizasyon süreçlerini gerçekleştirmek için matematiksel olarak modellenmiştir.

Genel olarak, popülasyon tabanlı algoritmalar en iyileme süreçlerine rastgele oluşturulan bir dizi aday çözümle başlar. Oluşturulan bu çözüm seti, aşamalı olarak bir dizi optimizasyon kuralıyla geliştirilir ve belirli bir hedef işlevi tarafından yinelemeli olarak değerlendirilir. Popülasyon tabanlı algoritmalar, optimizasyon problemlerinin optimal çözümünü stokastik olarak bulmaya çalıştığından, tek seferde bir çözüm elde etmek garanti edilmez. Bununla birlikte, verilen problem için genel en iyi çözümü elde etme olasılığı, yeterli sayıda

rastgele çözüm ve optimizasyon yinelemeleri ile artırılmaktadır (Mirjalili, 2016).

Popülasyon tabanlı optimizasyon yöntemleri arasında yer alan metasezgisel algoritmaların aralarındaki farklılıklara rağmen, optimizasyon süreci iki ana aşamadan oluşmaktadır: keşif ve sömürü. Keşif aşamasında yerel çözümlerden kaçınmak için arama aracılarını kullanarak geniş çaplı bir arama alanını kapsamak hedeflenirken, sömürü aşamasında keşif aşamasında elde edilen çözümlerin iyileştirilmesi amaçlanmaktadır. AOA, optimizasyon problemlerini türevlerini hesaplamadan çözebilen popülasyon tabanlı bir metasezgisel algoritmadır.

3.2.1. Başlatma Aşaması

AOA'da optimizasyon süreci, Denklem 7'de gösterildiği gibi rastgele oluşturulan bir dizi aday çözüm ile başlar ve her yinelemedeki en iyi aday çözüm, şimdiye kadar elde edilen en iyi çözüm veya neredeyse optimum olarak kabul edilir.

$$Y = \begin{bmatrix} y_{1,1} & \cdots & \cdots & y_{1,j} & y_{1,n-1} & y_{1,n} \\ y_{2,1} & \cdots & \cdots & y_{2,j} & \cdots & y_{2,n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ y_{N-1,1} & \cdots & \cdots & y_{N-1,j} & \cdots & y_{N-1,n} \\ y_{N,1} & \cdots & \cdots & y_{N,j} & y_{N,n-1} & y_{N,n} \end{bmatrix} \quad (7)$$

AOA çalışmaya başlamadan önce, hangi aşamada çalışılacağı belirlenmelidir. Bu amaçla MOA (Math Optimizer Accelerated) katsayısı aşağıdaki gibi hesaplanır.

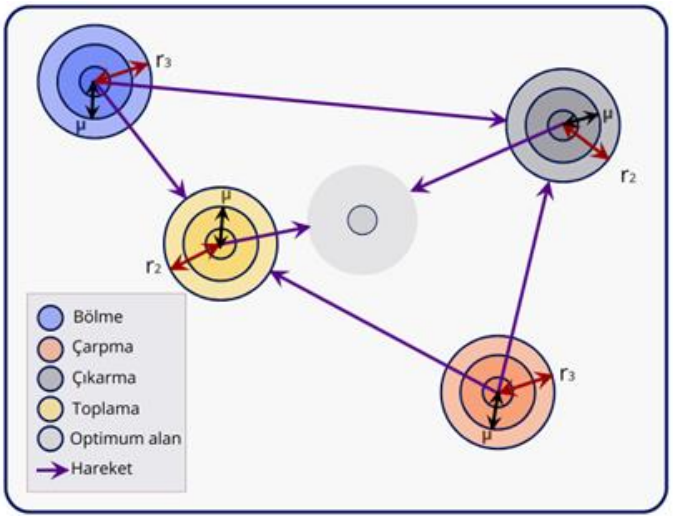
$$MOA(k) = min + k\left(\frac{max - min}{k_{max}}\right) \quad (8)$$

Burada $MOA(k)$ değeri fonksiyonun k. yinelemedeki değerini, k_{max} maksimum yineleme sayısını ifade etmektedir. min ve max değerleri ise sırasıyla fonksiyonun alabilecekleri en büyük ve en küçük değerleri göstermektedir. Mevcut yinelemeyi belirten k ifadesi, 1 ile k_{max} arasında bir değer almaktadır.

3.2.2. Keşif Aşaması

Aritmetik operatörlerden Bölme (D) ve Çarpma (M) operatörünü içeren matematiksel hesaplamalar, keşif arama mekanizmasına uygun bir şekilde yüksek dağılıma sahip değerler sağlar. Bununla birlikte, Bölme ve Çarpma işlemleri, Çıkarma (S) ve Toplama (A) işlemlerinin aksine, yüksek dağılımlarından dolayı hedefe kolayca yaklaşamazlar. Farklı operatörlerin dağılım değerlerinin etkisini göstermek için dört işlemi de kullanmaya dayalı bir fonksiyon kullanılır. Keşif araştırması, birkaç yinelemeden sonra çıkarılabilecek optimale yakın çözümü tespit eder. Bunun yanında, bölme ve çarpma işlemleri, aralarındaki gelişmiş iletişim yoluyla sömürü aşamasını da destekler.

AOA'nın keşif operatörleri, arama alanını birkaç bölgede rastgele keşfeder ve Denklem 9'da modellenen iki ana arama stratejisine (Bölme (D) ve Çarpma (M) arama stratejileri) dayalı daha iyi bir çözüm bulmaya çalışır. Bu aşamada hangi stratejinin kullanılacağı (Bölme veya Çarpma), Denklem 8'de verilen MOA işlevi ile belirlenir. Şekil 1'de, kullanılan operatörlerin optimum alana nasıl yaklaştığı gösterilmiştir. Bu aşamada, $r_2 < 0.5$ durumunda Bölme operatörü kullanılır ve diğer operatör (Çarpma) bu operatör mevcut görevini bitirene kadar ihmal edilir. $r_2 \geq 0.5$ durumunda ise Çarpma operatörü kullanılacaktır. r_2 her yinelemede 0 ile 1 arasında rastgele seçilen bir sayıdır. Bu rastsallık, arama alanının farklı bölgelerini keşfedilebilmesini sağlamaktadır.



Şekil 1. Kullanılan aritmetik operatörlerin optimum alana yaklaşma mekanizması (Abualigah v.d., 2020)

Keşif aşamasında Bölme ve Çarpma operatörlerine göre çözümlerin güncellenme kuralı şu şekildedir:

$$y_{i,j}(k+1) = \begin{cases} \frac{best(y_j)}{MOP + \epsilon} \cdot ((UB_j - LB_j) \cdot \mu + LB_j) & \text{eğer } r_2 < 0.5 \text{ ise} \\ best(y_j) \cdot MOP \cdot ((UB_j - LB_j) \cdot \mu + LB_j) & \text{eğer } r_2 \geq 0.5 \text{ ise} \end{cases} \quad (9)$$

Burada $y_{i,j}(k)$, i. çözümün k. yinelemedeki j. konum değerini, $best(y_j)$ o adıma kadarki elde edilen en iyi çözümün j. konum değerini, UB_j ve LB_j değerleri j. konumun üst ve alt sınırlarını temsil etmektedir. Küçük ve pozitif bir sayı olan ϵ adım boyu olup, μ arama sürecini kontrol eden bir parametredir ve 0.5 olarak seçilmesinin uygun olduğu belirtilmiştir (Abualigah v.d., 2020). Denklem 9'daki MOP (Math Optimizer Probability) bir katsayı olup

$$MOP(k) = 1 - \left(\frac{k}{k_{max}}\right)^{1/\alpha} \quad (10)$$

olarak tanımlanır. Burada α değeri yinelemeler üzerinden sömürü doğruluğunu tanımlamaktadır ve 5 olarak seçilmesinin uygun olduğu ifade edilmiştir (Abualigah v.d., 2020).

3.2.3. Sömürü Aşaması

Aritmetik operatörlerden Çıkarma'nın (S) veya Toplama'yı (A) içeren matematiksel hesaplamalar, sömürü arama mekanizmasına karşılık gelen yüksek yoğunluklu değerler sağlar. Çıkarma ve Toplama operatörleri diğer operatörlerin aksine, düşük dağılımlarından dolayı hedefe kolayca yaklaşabilirler. Bu nedenle, sömürü araştırması birkaç denemeden (yinelemeden) sonra çıkarılabilecek optimuma yakın çözümü tespit eder ve sömürü operatörleri (S ve A) aralarındaki gelişmiş iletişim yoluyla sömürü aşamasını desteklemek için optimizasyonun bu aşamasında çalıştırılırlar.

Çıkarma ve Toplama işlemlerinin kullanıldığı sömürü araması, r_1'in mevcut MOA(k) değerinden büyük olmadığı durumda çalıştırılır (Denklem 11). AOA'da, sömürü operatörleri

(S ve A) arama alanını birkaç yoğun bölgede aşağıdaki denklemde modellendiği gibi derinlemesine araştırır.

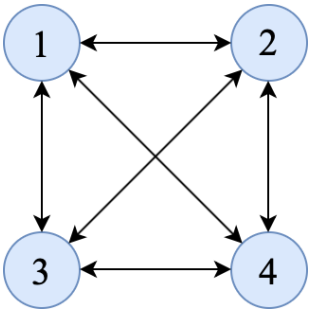
$$y_{i,j}(k+1) = \begin{cases} best(y_j) - MOP \times \left((UB_j - LB_j) \times \mu + LB_j \right), & r_3 < 0.5 \text{ ise} \\ best(y_j) + MOP \times \left((UB_j - LB_j) \times \mu + LB_j \right), & r_3 \geq 0.5 \text{ ise} \end{cases} \quad (11)$$

Bu aşama, derin bir arama yaparak arama uzayını sömürür. Bu aşamadaki Çıkarma operatörü, (Denklem 11 'deki ilk kural), $r_3 < 0.5$ durumunda çalışır ve Toplama operatörü bu operatör mevcut görevini bitirene kadar ihmal edilir. Aksi takdirde, Çıkarma yerine mevcut görevi gerçekleştirmek için Toplama devreye girecektir. Bu aşamadaki bu prosedürler, önceki aşamanın bölümlerine benzemektedir. Ancak, sömürü operatörleri (Çıkarma ve Toplama) genellikle yerel arama alanında sıkışıp kalmaktan kaçınmaya çalışır. Bu prosedür, keşif arama stratejilerine en uygun çözümü bulmada ve aday çözümlerin çeşitliliğini korumada yardımcı olur. Keşfi yalnızca ilk yinelemelerde değil, son yinelemelerde de sürdürmek için her yinelemede μ parametresi rastsal bir değere sahip olacak şekilde tasarlanmıştır. Özellikle son yinelemelerde aramanın bu kısmı, yerel optimal durgunluk durumunda çok faydalıdır.

Şekil 1, bir arama çözümünün konumlarını iki boyutlu bir arama uzayında Bölme (D), Çarpma (M), Çıkarma (S) ve Toplama (A)'ya göre nasıl güncellediğini açıklamaktadır. Nihai elde edilen pozisyonun, arama kapsamındaki D, M, S ve A pozisyonları tarafından belirlenen bir aralık dahilinde stokastik bir pozisyonda olabileceği görülebilir. Sonuç olarak D, M, S ve A, optimuma yakın çözümün konumunu tahmin eder ve diğer çözümler optimuma yakın çözüm bölge etrafındaki konumlarını stokastik olarak günceller.

4. Benzetim Çalışmaları

Şekil 2'de verilen ve 4 etmenden oluşan bir ağı ele alalım.



Şekil 2. Dört etmenden oluşan tam bağlı bir ağ

Bu ağda aşağıdaki denklem sisteminin dağıtık çözümü elde edinmek isteniyor olsun.

$$\begin{aligned} 3x_1 + 7x_2 + x_3 + 7x_4 &= 48 \\ -9x_1 + 10x_2 + 5x_3 - 2x_4 &= 18 \\ 3x_1 - 5x_3 - 7x_4 &= -40 \\ 10x_1 - 4x_2 + x_3 &= 5 \end{aligned} \quad (12)$$

Sistemdeki i. etmenin $(i = 1,2,3,4)$, Denklem 12'deki i. denklemi bildiğini durumda, etmenlerin Denklem 1'de verilen algoritmalarda kullanacakları P_i matrisleri

$$P_1 = \begin{bmatrix} 11/12 & -7/36 & -1/36 & -7/36 \\ -7/36 & 59/108 & -7/108 & -49/108 \\ -1/36 & -7/108 & 107/108 & -7/108 \\ -7/36 & -49/108 & -7/108 & 59/108 \end{bmatrix}$$

$$P_2 = \begin{bmatrix} 43/70 & 3/7 & 3/14 & -3/35 \\ 3/7 & 11/21 & -5/21 & 2/21 \\ 3/14 & -5/21 & 37/42 & 1/21 \\ -3/35 & 2/21 & 1/21 & 103/105 \end{bmatrix}$$

$$P_3 = \begin{bmatrix} 74/83 & 0 & 15/83 & 21/83 \\ 0 & 1 & 0 & 0 \\ 15/83 & 0 & 58/83 & -35/83 \\ 21/83 & 0 & -35/83 & 34/83 \end{bmatrix} \quad (13)$$

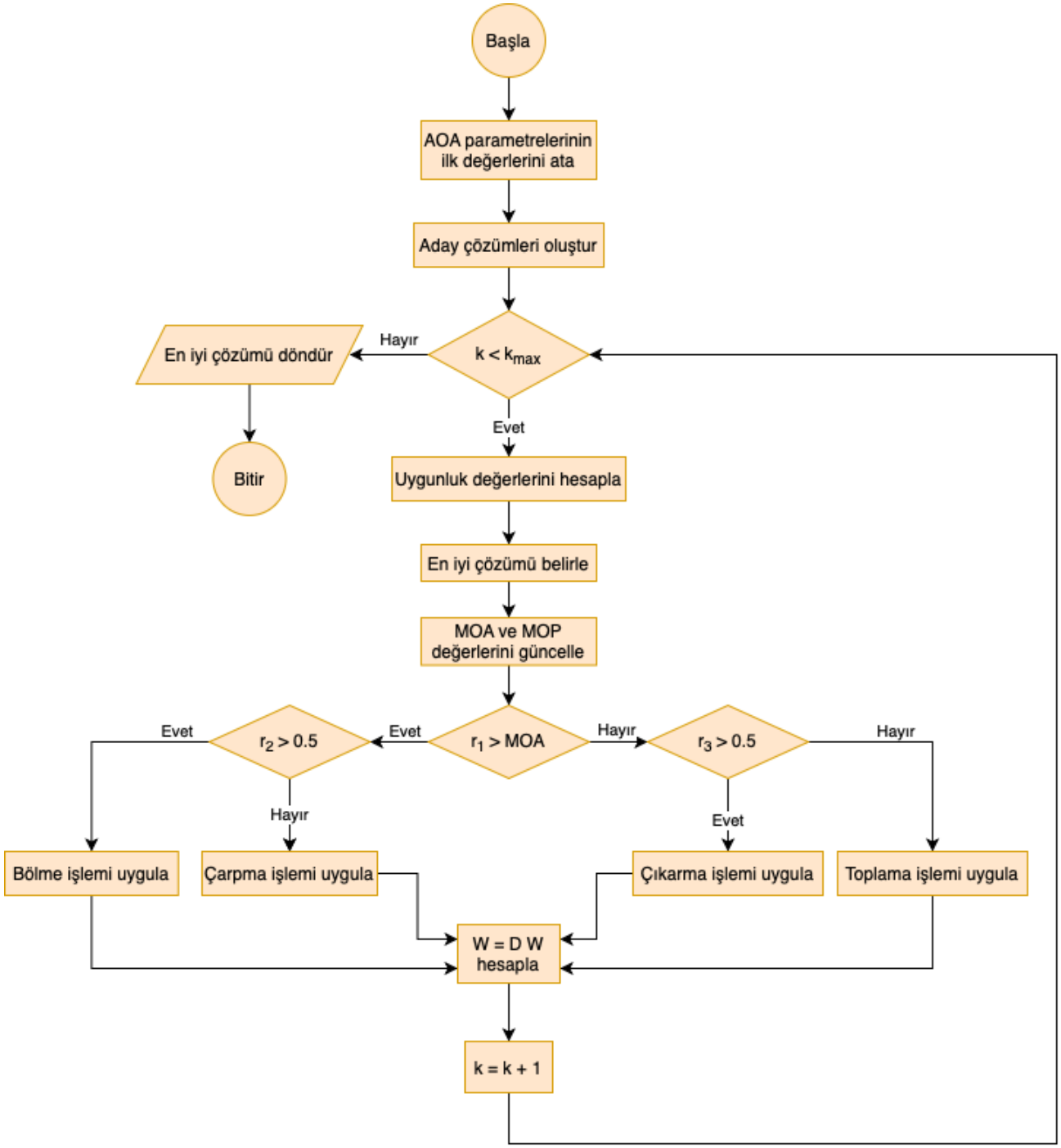
$$P_4 = \begin{bmatrix} 17/117 & 40/117 & -10/117 & 0 \\ 40/117 & 101/117 & 4/117 & 0 \\ -10/117 & 4/117 & 116/117 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

olarak hesaplanır. Etmenlerin denklem sistemini en hızlı şekilde çözebilmeleri için Denklem 6'da formülasyonu verilen optimizasyon probleminin çözümü bulunmalıdır. Bu amaçla, Şekil 3'te akış diyagramı verilen Aritmetik Optimizasyon Algoritması kullanılarak w_{ij} ağırlıklandırma katsayıları belirlenecektir. Optimizasyon algoritmasının bulacağı w_{ij} katsayılarının Varsayım 1(i) ve (ii)'nin sağlamasını garanti etmek için alt limitler 0 ve üst limitler 1 olarak belirlenmiş olup, her yinelemenin sonunda elde edilen katsayılar matrisi $W = [w_{ij}]$ soldan

$$D = diag(W\mathbf{1})^{-1} \quad (14)$$

ile çarpılarak Varsayım 1(iii) sağlanmaktadır. Burada $\mathbf{1}$, tüm elemanları 1'den oluşan bir sütun vektördür.

Aritmetik Optimizasyon Algoritmasının arama ajanı ve yineleme sayısının algoritmanın performansına etkisini incelemek amacıyla, optimizasyon algoritması probleminin çözümü 4 farklı senaryo için yirmişer kere çalıştırılmış olup, elde edilen sonuçlar Tablo 1'de verilmiştir. Tablodan da görüleceği üzere arama ajanı sayısı ve yineleme sayısı arttıkça, elde edilen ortalama λ^*_{max} değerleri artmıştır ancak en iyi λ^*_{max} değerleri değişmemiştir. Bunun nedeni, aritmetik optimizasyon algoritmasının keşif aşamasında arama uzayının çok farklı alanlarının keşfedilmesini sağlaması ve en iyi çözüme düşük ajanı sayısıyla bile kısa sürede ulaşabilmesidir.



Şekil 3. Aritmetik Optimizasyon Algoritması'nın akış diyagramı

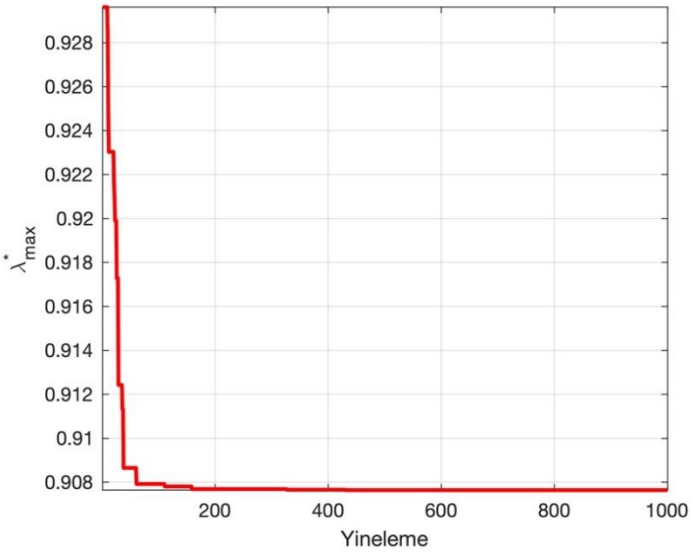
Tablo 1. Arama Ajanı ve Yineleme Sayısının Aritmetik Optimizasyon Algoritmasının Performansına Etkisi

Ölçüt	**Senaryo 1 Arama Ajanı: 25 Yineleme: 500**	**Senaryo 2 Arama Ajanı: 25 Yineleme: 1000**	**Senaryo 3 Arama Ajanı: 25 Yineleme: 2000**	**Senaryo 1 Arama Ajanı: 50 Yineleme: 500**	**Senaryo 2 Arama Ajanı: 50 Yineleme: 1000**	**Senaryo 3 Arama Ajanı: 50 Yineleme: 2000**
En iyi λ^*_{max}	0.9076	0.9076	0.9076	0.9076	0.9076	0.9076
Ortalama λ^*_{max}	0.9092	0.9086	0.9084	0.9087	0.9081	0.9080
En kötü λ^*_{max}	0.9122	0.9122	0.9122	0.9123	0.9124	0.9122
Standart sapma	0.0021	0.0017	0.0016	0.0018	0.0011	0.0011

Yapılan tüm optimizasyon çalışmalarından elde edilen en iyi sonucu sağlayan katsayılar matrisi 25 arama ajanı ve 1000 yineleme senaryosu için

$$W = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0.1187 & 0 & 0.5004 & 0.3809 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad (15)$$

olarak bulunmuş olup, yinelemeye bağlı olarak λ^*_{max} değerinin değişimi Şekil 4'teki gibidir.

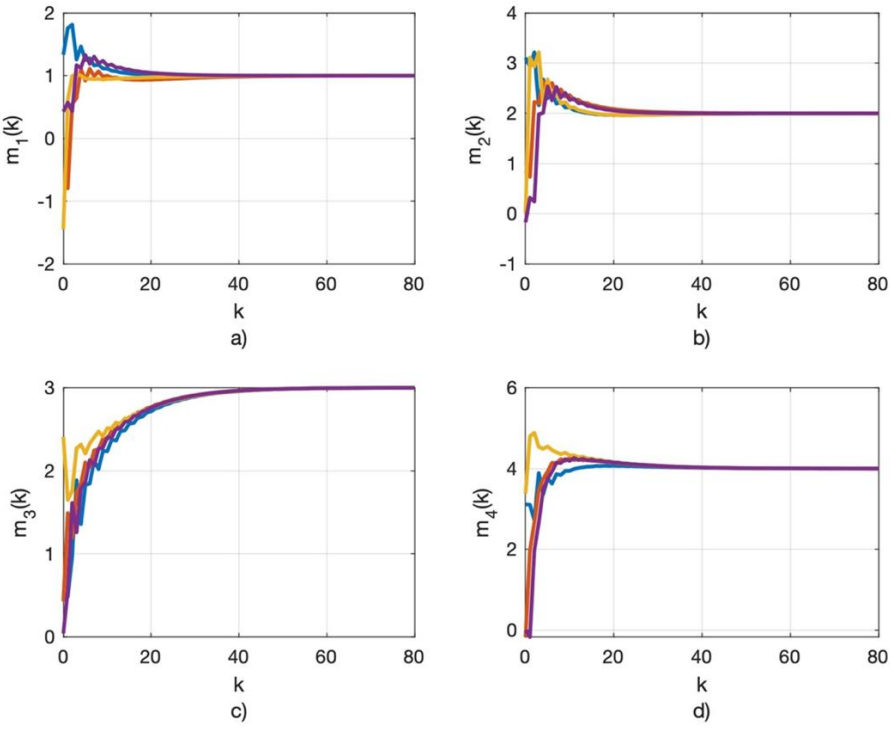


Şekil 4. Aritmetik Optimizasyon Algoritması'nın bulduğu λ^*_{max} değerinin yinelemeye bağlı değişimi

Denklem 15'teki katsayılar matrisi kullanılarak Denklem 12'de verilen doğrusal denklem sistem Şekil 2'de verilen çok etmenli ağ üzerinde Denklem 1 algoritması kullanılarak çözüldüğünde, etmenlerin doğrusal denklem sisteminin eşsiz çözümü olan $x^* = [1, 2, 3, 4]^T$'e yakınsaması Şekil 5'te gösterilmiştir.

5. Sonuç

Bu çalışmada bir doğrusal denklem sistemini çok etmenli sistemler üzerinde dağıtık olarak çözmek için önerilmiş olan bir algoritmanın yakınsama hızının en iyilenmesi problemi ele alınmıştır. Doğrusal denklem sisteminin yalnızca bir alt kümesinin sistemdeki etmenler tarafından bilindiği durumda; etmenlerin yerel denklem bilgilerinin yanısıra komşu etmenlerin çözüm tahminlerini de kullanarak kendi tahminlerini güncelleyebilmekte ve denklem sisteminin eşsiz çözümüne ulaşmaya çalışmaktadırlar. Çözüm hatası dinamikleri bir doğrusal dinamik sistem olarak formüle edildiğinde, algoritmanın yakınsama hızını en iyilemek amacıyla sistem matrisinin en büyük özdeğerini en küçüklemek gerekmektedir ve bu amaçla literatürde yakın zamanda önerilmiş bir metasezgisel optimizasyon algoritması olan Aritmetik Optimizasyon Algoritması kullanılmıştır. Benzetim çalışmalarında örnek bir denklem sistemi ele alınmış ve bu denklem sisteminin dağıtık olarak en hızlı çözümünü sağlayan tasarım parametreleri belirlenmiştir. Benzetim çalışmaları farklı sayıda arama ajanı ve maksimum yineleme sayısı için tekrarlanarak arama ajanı ve yineleme sayılarının elde edilen en iyi hızlı yakınsama değerine etkisi incelenmiştir. En hızlı yakınsamayı sağlayan tasarım parametreleri kullanılarak denklem sisteminin çok etmenli sistem üzerindeki çözümünün zamana bağlı değişimi gösterilmiştir.



Şekil 5. Etmenlerin doğrusal denklem sisteminin çözüm tahmininin değişimi: a) 1. bilinmeyen, b) 2. bilinmeyen, c) 3. bilinmeyen d) 4. bilinmeyen

6. Teşekkür

Bu çalışma TÜBİTAK Proje No: 117E204 tarafından desteklenmiştir.

Kaynakça

Abualigah, L., Diabat, A., Mirjalili, S., Abd Elaziz, M., & Gandomi, A. H. (2021). The Arithmetic Optimization Algorithm. Computer Methods in Applied Mechanics and Engineering, 376, 113609.

Anderson, J. (1995). Computational fluid dynamics: The basics with applications. Mc- Graw-Hill Education.

Anderson, B.D.O., Mou, S., Morse, A., & Helmke, U. (2016). Decentralized gradient algorithm for solution of a linear equation. Numerical Algebra, Control and Optimization, 6(3), 319-328.

Carpentieri, B., Duff, I. S., Giraud, L., & Magolu monga Made, M. (2004). Sparse symmetric preconditioners for dense linear systems in electromagnetism. Numerical Linear Algebra with Applications, 11(89), 753-771.

Cihan, O. (2019). Rapid Solution of Linear Equations with Distributed Algorithms over Networks. IFAC-PapersOnLine, 52(25), 467-471.

Cihan, O. (2020). Topology Design for Group Consensus in Directed Multi-Agent Systems, Kybernetika, 56(3), 578-597.

Cihan, O., & Akar, M. (2020a). Multi-consensus of second-order agents in discrete-time directed networks. International Journal of Systems Science, 51(10), 1847-1861.

Cihan, O., & Akar, M. (2020b). Necessary and Sufficient Conditions for Group Consensus of Agents With Third-Order Dynamics in Directed Networks. Journal of Dynamic Systems, Measurement, and Control, 142(4).

Frank, A., Fabregat-Traver, D., & Bientinesi, P. (2016). Large-scale linear regression: Development of high-performance routines. Applied Mathematics and Computation, 275, 411-421.

Hackbusch, W. (1994). Iterative solution of large sparse systems of equations: Vol. 95. Springer.

Krstic, M., & Smyshlyaev, A. (2008). Boundary control of PDEs: A course on backstepping designs: Vol. 16. SIAM.

Mirjalili, S. (2016). SCA: A Sine Cosine Algorithm for solving optimization problems. Knowledge-Based Systems, 96, 120-133.

Mou, S., Liu, J., & Morse, A. S. (2015). A Distributed Algorithm for Solving a Linear Algebraic Equation. IEEE Transactions on Automatic Control, 60(11), 2863-2878.

Pasqualetti, F., Carli, R., & Bullo, F. (2012). Distributed estimation via iterative projections with application to power network monitoring. Automatica, 48(5), 747-758.

Rahola, J. (1996). Solution of dense systems of linear equations in the discrete-dipole approximation. SIAM Journal on Scientific Computing, 17(1), 78-89.

Silvestre, D., Hespanha, J., & Silvestre, C. (2018). A pagerank algorithm based on asynchronous Gauss–Seidel iterations. In 2018 annual American control conference (pp. 484-489).

Stott, B., Jardim, J., & Alsac, O. (2009). DC power flow revisited. IEEE Transactions on Power Systems, 24(3), 1290-1300.

Wang, L., Fullmer, D., & Morse, A. S. (2016). A distributed algorithm with an arbitrary initialization for solving a linear algebraic equation. In 2016 American control conference (pp. 1078-1081).

Wang, P., Lin, P., Ren, W., & Song, Y. (2018). Distributed subgradient-based multiagent optimization with more general step sizes. IEEE Transactions on Automatic Control, 63(7), 2295-2302.

Wang, P., Mou, S., Lian, J., & Ren, W. (2019). Solving a system of linear equations: From centralized to distributed algorithms. Annual Reviews in Control, 47, 306-322.

Wang, P., Ren, W., & Duan, Z. (2019). Distributed Algorithm to Solve a System of Linear Equations With Unique or Multiple Solutions From Arbitrary Initializations. IEEE Transactions on Control of Network Systems, 6(1), 82-93.

You, K., Song, S., & Tempo, R. (2016). A networked parallel algorithm for solving linear algebraic equations. In 2016 IEEE 55th conference on decision and control (pp. 1727-1732).