BULUT AĞLARINA YÖNELİK DAĞINIK ÖNBELLEK YÖNETİM SİSTEMİ'NDE FARKLI OPTİMİZASYON VE ATAMA TEKNİKLERİNİN PERFORMANS KARŞILAŞTIRMASI

Hüseyin Seçkin Dikbayır[1] Asım Egemen Yılmaz[2] Ali Arda Diri[3]
[1,3]Dirisoft Bilgi ve İletişim Teknolojileri Ltd. Şti.
Ankara Teknoloji Geliştirme Bölgesi, Cyberpark, Cyberplaza A Blok Kat 7 No:A704, Bilkent, Ankara, Türkiye
[2]Elektrik-Elektronik Müh. Bölümü, Ankara Üniversitesi Mühendislik Fakültesi
[1]e-posta: sdikbayir@dirisoft.com [2]e-posta: aeyilmaz@eng.ankara.edu.tr
[3]e-posta: ardadiri@dirisoft.com

ÖZET: Bu çalışmada, bulut ağlarında çalışmak üzere geliştirilmiş sezgisel ve deterministik optimizasyon teknikleriyle çalışan dağınık önbellek yazılımının optimizasyon modülü için geliştirilmiş olan farklı algoritmalar ve bu algoritmaların bütçe, performans, konum ve güvenlik kıstasları doğrultusunda yaptıkları eşleme ve optimizasyon performansları incelenmiştir. Bu amaçla, Peer-To-Peer Algoritması, Peer-To-Peer Modified Algoritması, Auction Algoritması, Munkres Algoritması, Divide And Conquer Munkres Algoritması, Divide And Conquer Auction Algoritması, Brute Force Algoritması ve Genetik Algoritma, performanslarına göre kıyaslamalı olarak incelenmiştir. Optimizasyon algoritmalarının kullanıldığı yazılım, birbirine bağlı birçok sunucunun tek bir sunucuymuş gibi davranmasını sağlayarak heterojen ağlarla bulut ağlarını birbirine bağlayarak, birbirine bağlı işlemcilerde ortak bir önbellek indeksi oluşturmaktadır. Bu yazılımın en önemli özelliği, sunucu–istemci eşleme işlemini bir "genel atama problemi"ne benzeterek modellemesi, ardından bütçe, performans, konum ve güvenlik kıstasları doğrultusunda en optimize şekilde yapmasıdır. Bunu gerçekleştirebilmek için de istemci sunucu sayılarına ve kullanıcı kıstaslarına göre en efektif algoritmayı belirlemesi gerekmektedir. Algoritmaların, optimizasyon alanındaki etkinliklerini objektif olarak ölçebilmek için, bu dokümana konu olan testler her algoritma için sabit kıstas kümesi kullanılarak geliştirilmiştir. Testlerde Monte Carlo simülasyon yöntemi uygulanarak 1-100 ve 1-1000 arası rastgele maliyet değerleriyle, her bir matris boyutu için 50 tane olmak üzere, 10×10, 50×50, 100×100, 250×250, 500×500, 750×750, 1000×1000, 10×2, 10×5, 10×7, 10×9, 50×2, 50×5, 50×7, 50×9, 100×2, 100×5, 100×7, 100×9, 250×2, 250×5, 250×7, 250×9, 500×2, 500×5, 500×7, 500×9, 750×2, 750×5, 750×7, 750×9, 1000×2, 1000×5, 1000×7, 1000×9 boyutlarında test amaçlı kullanıcı-kaynak matrisleri oluşturulmuş ve algoritmalar bu matrislerle test edilmiştir.

Anahtar Kelimeler: Bulut Ağları, Önbellek Yönetimi, Genel Atama Problemi Optimizasyon, Sezgisel ve Deterministik Algoritmalar

1. GİRİŞ

Bu çalışmaya konu olan dağınık önbellek yazılımı, ağ üzerinde birbirine bağlı bulunan sunucularda ortak bir önbellek oluşturarak, önbellek ve ön belleği kullanan işlem arasında, çeşitli optimizasyon teknikleriyle çalışan bir eşleme yazılımıdır. Dağınık önbellek yazılımının amacı heterojen ağlarda, işletim sisteminden bağımsız olarak, birbirine bağlı birçok sunucunun tek bir sunucuymuş gibi davranarak ortak bir önbellek indeksi oluşturmasıdır. Bu yazılımın çözdüğü optimizasyon problemi ise sürekli değişen *işletme politikası* ve *uygulama alanı politikası* kısıtlamaları doğrultusunda çizelgeleme (*scheduling*) problemini çözmek ve sunucu-istemci eşlemesini yapmaktır. Optimizasyon, şu kısıtlamaları göz önünde bulundurarak yapılmaktadır;

- Bütçe kısıtlamaları: Sabit Disk, Bellek, İşlem ve Ağ Genişliği maliyetleri
- Performans kısıtlamaları: Maksimum Sabit Disk, Bellek, İşlem ve Ağ Genişliği kullanımı
- Konum kısıtlamaları: Çalışma zamanı, ağ paketinin maksimum seyahat süresi (*Maximum Travel Time*) ve atlama limiti (*Maximum Hop Time*)
- Güvenlik kısıtlamaları: Kullanıcı kısıtlamaları, Uygulama Alanı kısıtlamaları, Ağ grubu kısıtlamaları.

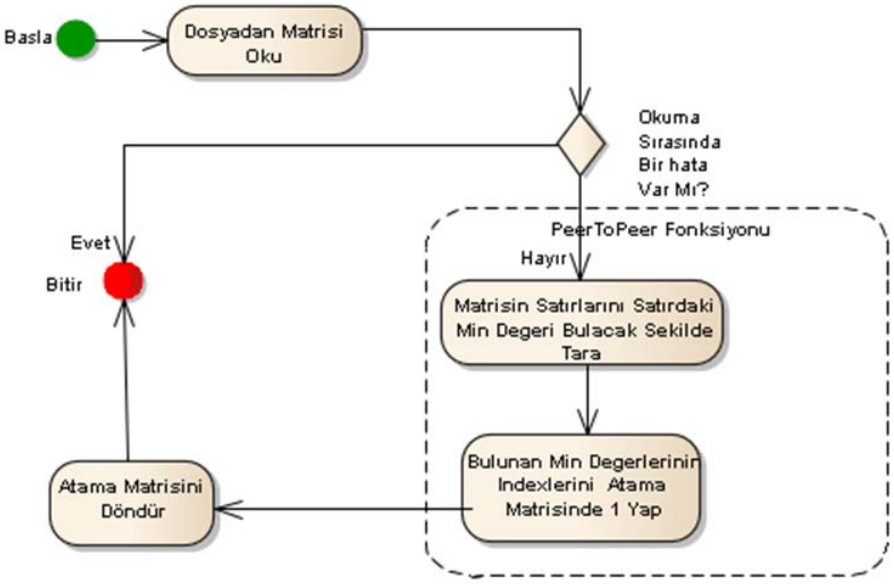
Bu optimizasyon alanında uygulanacak olan çizelgeleme problemi en kaba anlamda, bir küme/liste halinde verilen bir takım görevlerin minimum zamanda yerine getirilmesi için kaynak planlamasının yerine getirilmesi olarak tanımlanabilir. Öncelikli görevlerin (varsa) ilk başta yerine getirilmesi, unsurlara görev ataması yapılırken her bir unsurun kabiliyetinin göz önünde bulundurulması vb. hususlar, problemin kısıtlarını (*constraint*) oluşturmaktadır. Burada kısıtlar çok amaçlı (*multi-objective*) optimizasyon problemine dönüşmekte ve bu kısıtların gerçek zamanlı olarak değişmesi de problemin zorluğunu bir kat arttırmaktadır.

2. TEST KURGUSU VE YAKLAŞIM

Test yazılımları C++ yazılım dili kullanılarak geliştirilmiştir. Algoritmaların tepkilerini ve doğruluğunu ölçmek amacıyla oluşturulan test yazılımında şu noktalar göz önüne alınmıştır:

- Test esnasında Monte Carlo metodu uygulanarak rastgele olarak üretilen 50 adet matris her algoritmada koşturulmuştur (Monte Carlo metodu, sistemde yer alan değişkenlerin olasılık dağılımlarının belirlenmesini gerektirir. Bu dağılımlardan hareketle rastgele sayılar kullanarak yapılan örnekleme yoluyla sistemin davranışına ilişkin veri toplanır. Modelde yer alan rastgele değişkenlerin zaman içinde alacakları değerler bir dizi rastgele sayı kullanılarak belirlenir. Rastgele sayılar yoluyla oluşturulan olaylar yapay nitelikte olmakla birlikte gerçek durumu yansıtan özelliktedir).
- Üretilen 50 matrisin birbirinden farklı olup olmadıkları, her üretimden sonra test edilmiştir. Üretilen matrislerin elemanları 1 den 100'e ve 1 den 1000'e

kadar olan sayı dizisi arasından seçilen rastgele bir sayıdır. Test sonuçları, üretilen bu 50 adet matrisin ortalama değerleri göz önünde bulundurularak elde edilmiştir.



Şekil 1 Test Kurgusu

- Matrisler dikdörtgen ve kare olmak üzere iki ayrı yapıdadır.

Kare matrislerin boyutları:

- 10×10
- 50×50
- 100×100
- 250×250
- 500×500
- 750×750
- 1000×1000

Dikdörtgen matrislerin boyutları:

• 10×2	10×5	10×7	10×9
• 50×2	50×5	50×7	50×9
• 100×2	100×5	100×7	100×9
• 250×2	250×5	250×7	250×9
• 500×2	500×5	500×7	500×9
• 750×2	750×5	750×7	750×9
• 1000×2	1000×5	1000×7	1000×9

• Test sırasında, her bir algoritmanın kaynakları ne kadar etkin şekilde kullandıklarının ölçülmesi ve kullanılmayan kaynakların tespiti için çeşitli metrikler tanımlanmıştır. Bu metrikler aşağıdaki gibidir:

- Ortalama Kaynak Kullanım Oranı / Minimum Kaynak Kullanım Oranı
- Maksimum Kaynak Kullanım Oranı/ Minimum Kaynak Kullanım Oranı
- Minimum Kaynak Kullanım Oranı / Ortalama Kaynak Kullanım Oranı
- Kullanılmayan Kaynak Sayısı / Toplam Kaynak Sayısı

3. TANIMLAR

3.1 ORTALAMA ZAMAN

Topolojide kullanılan ortalama zaman ifadesi her bir matrisin çözüm sürelerinin toplamının oluşturulan matris sayısına bölümünden bulunan sonuçtur. Test süresince üretilen toplam matris sayısı her bir algoritma ve matris boyutları için 50'dir. Ortalama zaman ifadesini bu doğrultuda matematiksel olarak ifade etmek gerekirse:

(1)

Testte kullanılan ortalama zaman ifadesi ise bu doğrultuda;

(2)

şeklindedir.

3.2 STANDART SAPMA

Topolojide kullanılan zaman standart sapma ise şu şekilde açıklanabilir; Olasılık kuramı ve istatistik bilim dallarında varyans bir rassal değişken, bir olasılık dağılımını veya örneklem için istatistiksel yayılımın, mümkün bütün değerlerin beklenen değer veya ortalamadan uzaklıklarının karelerinin ortalaması şeklinde bulunan bir ölçüdür.

Standart sapma ise veri değerlerinin yayılımının özetlenmesi için kullanılan bir ölçüdür. Standart sapma varyansın karekökü̈dür. Daha matematiksel bir ifade ile standart sapma veri değerlerinin aritmetik ortalamadan farklarının karelerinin toplamının, veri sayısının bir eksiğine bölümünün karekökü̈dür, yani verilerin ortalamadan sapmalarının kareler ortalamasının karekökü olarak tanımlanır.

(3)

Testte kullanılan standart sapma hesabında ise formül aşağıdaki şekilde düzenlenmiştir:

(4)

Grafiklerde verilen zaman standart sapması ifadesi standart sapmanın aritmetik ortalamaya bölümünün yüzdelik ifadesidir.

– (5)

4. ALGORİTMALAR

4.1 PEER-TO-PEER ALGORİTMASI

Talepler ve kaynaklar arasında, her talebi kendi içerisinde maliyeti en düşük olan kaynağa atamayı hedef alan algoritmadır. İlk gelen talepten başlanarak sırayla her talep kendisine en uygun kaynağa atanır [1]. Bu algoritmada talepler çoğalsa da önceki atanan değerler değiştirilmez. Algoritmanın dezavantajı, bir kaynakta aşırı talep yüklenmesinin olmasıdır.

4.2 PEER-TO-PEER MODIFIED ALGORİTMASI

Bu algoritma da amaçlanan Peer-To-Peer algoritmasının dezavantajı olan aşırı kaynak yüklenmesini engellemektir. Algoritma ilk gelen talebi en uygun kaynağa atar ve o atama yapılan kaynağı tüm kaynaklar bir talebe atanana kadar kullanıma kapatır. Diğer talepler de var olan kaynaklar eşliğinde kendilerine en uygun kaynaklara atanırlar.

4.3 AUCTION (AÇIK ARTIRMA / MÜZAYEDE) ALGORİTMASI

Algoritmanın, "teklif verme" ve "atama" olmak üzere iki ana fazı bulunmaktadır. Teklif verme fazında, kendisine herhangi bir atama yapılmamış her bir talep için "en iyi" kaynak bulunur ve o kaynak için teklif verilir. Söz konusu kaynağın, başka bir talebe atanması durumunda, ataması iptal edilen talep, "en iyi ikinci" kaynağa atanır. Atama işlemi, her bir talep "tatmin olana" kadar devam eder ("Tatmin olma"nın tanımı, ilerideki paragraflarda detaylı olarak yapılacaktır). Herhangi bir j talebi, kendisinin bir i_j kaynağına atanması durumundaki kazanç değerinin, en uygun kazanç değerinin ε kadar yakınında olması durumunda "tatmin olmuş" olarak kabul edilir. Bir başka deyişle, $a_{i,j}$ değeri, j talebi ile i_j kaynağı arasındaki atamanın kazancı; P_{ij} değeri ise i_j kaynağının maliyeti olarak tanımlanırsa, "tatmin olma" durumu aşağıdaki gibi ifade edilebilir:

$$[\max_i(a_{ij} - P_i)] - (a_{i_j j} - P_{i_j}) \leq \varepsilon \quad (6)$$

Eğer en az bir uygun atama var ise, açık artırma algoritması en uygun çözümün nε komşuluğunda sonlanır. Bir başka deyişle, nihai atamanın toplam kazancı, en uygun kazancın nε komşuluğunda olur. Bütün a_{ij} değerleri tamsayı ve $\varepsilon < 1/n$ ise, en uygun sonuç bulunmuş olur [2].

4.4 MUNKRES ALGORİTMASI

Minimum maliyeti bularak, her kullanıcıyı farklı kaynaklara atamayı ön gören algoritmadır. En iyi maliyet değerini bulmak amacıyla maliyet matrisini sürekli olarak tarar. Amaç maliyet matrisindeki sıfırları kapatacak şekilde sanal çizgiler oluşturmak ve bu çizgilerin sayısını maliyet matrisinin satır sayısına eşit kılmayı sağlamaktır [3].

4.5 DIVIDE&CONQUER MUNKRES ALGORİTMASI

Divide and Conquer Munkres Algoritması en iyi maliyet matrisini en kısa sürede bulmayı amaçlar. Munkres Algoritmasında matris sürekli taranmaktadır, sürekli tarama mekanizması büyük matrislerde atama süresini uzatmakta, amaçlanan süreyi her zaman gerçekleyememektedir. Divide and Conquer Munkres Algoritması alınan matrisi sütun sayısına göre düzenler. Sonrasında düzenlenmiş matrisi sütun boyutuna bölerek tek tek işler ve atama işlemini tamamlar.

4.6 DIVIDE&CONQUER AUCTION ALGORİTMASI

Divide and Conquer Auction Algoritması en iyi maliyet matrisini, Auction algoritmasının "teklif verme" ve "atama" fazlarını daha kısa sürede sonuçlanmasını sağlayarak bulmayı amaçlar. Algoritma alınan matrisi sütun sayısına göre düzenler. Sonrasında düzenlenmiş matrisi sütun boyutuna bölerek tek tek işler ve atama işlemini tamamlar.

4.7 BRUTE FORCE (KABA KUVVET) ALGORİTMASI

Brute force algoritması maliyet matrisine yapılacak en uygun atamayı, tüm olası durumları sınayarak yapan algoritmadır. Algoritma matrisin sutun sayısına göre oluşturduğu permütasyon matrisleriyle, maliyet matrisinin elemanlarını sırasıyla atar. Atama sonucunda çıkan maliyet oranını hafızasında tutarak, bir sonraki sonuçla karşılaştırır. Eğer yeni çıkan sonuç bir öncekine göre daha uygunsa maliyet oranını değiştirerek permütasyon matrisleri üzerinde atama işlemlerine devam eder. Tüm olasılıkları taradıktan sonra, en uygun atama değerlerini ve maliyeti bularak sonlanır.

4.8 GENETİK ALGORİTMA

Genetik algoritma(GA) sezgisel arama ve optimizasyon algoritmaların bir sınıfıdır. Genel olarak algoritmada doğal seleksiyonun "en iyi olan hayatta kalır" prensibi işlenmektedir. Algoritma tekrarlayan optimizasyon tekniği kullanarak yeni popülasyonlar elde eder. Popülasyon üyeleri matrisin elemanları olarak alınır ve kromozom olarak kabul edilir. Doğal terminolojide kromozomlar genleri taşıyan birimlerdir. Algoritma da aynı genlerde olduğu gibi seleksiyon, çaprazlama ve mutasyon operasyonlarını her iterasyonda uygulamaktadır. Toplam iterasyon sayısına ulaştığında uygun maliyet değerini döndürerek sonlanmaktadır [4].

4. SONUÇLAR VE TARTIŞMA

Yapılan testler sonucunda şu sonuçlara varılmıştır:

a) Munkres algoritmasına giren matris, ne kadar kare matrise yakın bir yapıya sahipse çözüme ulaşmak da o kadar kısa süre almıştır.

b) Peer-To-Peer ve Peer-To-Peer Modified algoritmaları test sırasında oluşturulan her türlü matrisi sorunsuz bir şekilde 1sn'in altında bir sürede çözmüştür.

c) Tüm kare matrislerde sorunsuz (1sn'in altında) olarak çalıştırılabilecek algoritmalar şunlardır:

Peer-To-Peer	(0.8sn'in altında)
Peer-To-Peer Modified	(0.8sn'in altında)

d) Kare matrislerde 1sn altındaki zamanı elde edecek şekilde koşullu olarak uygulanabilecek algoritmalar şunlardır:

Auction Algoritması (Matris boyutu 100×100'den küçük olmalı)
Munkres Algoritması (Matris boyutu 500×500'den küçük olmalı)
Divide & Conquer Munkres (Matris boyutu 500×500'den küçük olmalı)
Divide & Conquer Auction (Matris boyutu 250×250'den küçük olmalı)

e) Tüm dikdörtgen matrislerde sorunsuz (1sn'in altında) olarak çalıştırılabilecek algoritmalar şunlardır:

PeerToPeer
PeerToPeer Modified
Divide & Conquer Munkres
Divide & Conquer Auction

f) Dikdörtgen matrislerde 1sn altındaki zamanı elde edecek şekilde koşullu olarak uygulanabilecek algoritmalar ise şu şekildedir:

Auction (Talep sayısı 250'den küçükse)
Munkres (Talep sayısı 75'den küçükse)
BruteForce (Permütasyon sayısı 5! – 6! mertebesinde ise)

g) En uygun maliyet değerleri sırasıyla Peer-To-Peer, Munkres, Divide & Conquer Munkres, Auction, Divide & Conquer Auction, Peer-To-Peer Modified algoritmalarında gözlenmiştir. BruteForce algoritması 1sn altında sonuçlara her zaman ulaşamadığından, bu kısımda değerlendirilmemiştir. Uygun değerlerde maliyet değeri olarak Peer-To-Peer algoritmasıyla aynı sıradadır.

h) Kaynakları etkili bir biçimde kullanan algoritmalar Auction, Divide & Conquer Auction ve Peer-To-Peer Modified olarak gözlenmiştir. Fakat Auction algoritması 1sn altında çözüme ulaşma isterini matris boyutlarına bağlı olarak gerçekleştiğinden, tercih sebebinde matrisin boyutlarının göz önünde bulundurulması gerektiği sonucuna varılmıştır.

i) En kısa sürede atama işlemini gerçekleyen algoritmalar sırasıyla Peer-To-Peer, Peer-To-Peer Modified, Divide & Conquer Munkres, Divide & Conquer Auction, Munkres, Auction, Brute Force olarak gözlenmiştir.

j) Divide & Conquer tipindeki algoritmaların en etkili çalıştığı matrislerin satır sayıları sütun sayılarından büyük olan matrisler olduğu gözlenmiştir.

Tablo 1 Peer to Peer Algoritması

Matris Boyutu	**Ortalama Zaman(ms)**	**Zaman Standart Sapması(%)**
10x10	0.5984113	151.38%
50x50	1.8877146	5.08%
100x100	4.9890142	3.93%
250x250	25.953428	0.68%
500x500	103.2806	0.70%
750x750	232.66576	0.33%
1000x1000	798.10548	9.79%

Tablo 2 Peer to Peer Modified Algoritması

Matris Boyutu	Ortalama Zaman(ms)	Zaman Standart Sapması(%)
10x10	1.303915	35.11%
50x50	1.8701538	6.22%
100x100	5.4640016	2.71%
250x250	27.602478	2.73%
500x500	105.60824	0.59%
750x750	235.70906	0.41%
1000x1000	791.86202	1.77%

Tablo 3 Auction Algoritması

Matris Boyutu	Ortalama Zaman(ms)	Zaman Standart Sapması(%)
10x10	2.087153	41.33%
50x50	125.33131	61.70%
100x100	1537.5125	42.73%
250x250	0	0
500x500	0	0
750x750	0	0
1000x1000	0	0

Tablo 4 Munkres Algoritması

Matris Boyutu	Ortalama Zaman(ms)	Zaman Standart Sapması(%)
10x10	2.1707568	34.36%
50x50	20.106536	23.04%
100x100	51.16341	24.42%
250x250	206.9778	1.53%
500x500	1105.4312	1.32%
750x750	3312.817	1.12%
1000x1000	13844.814	1.81%

Tablo 5 Divide & Conquer Munkres Algoritması

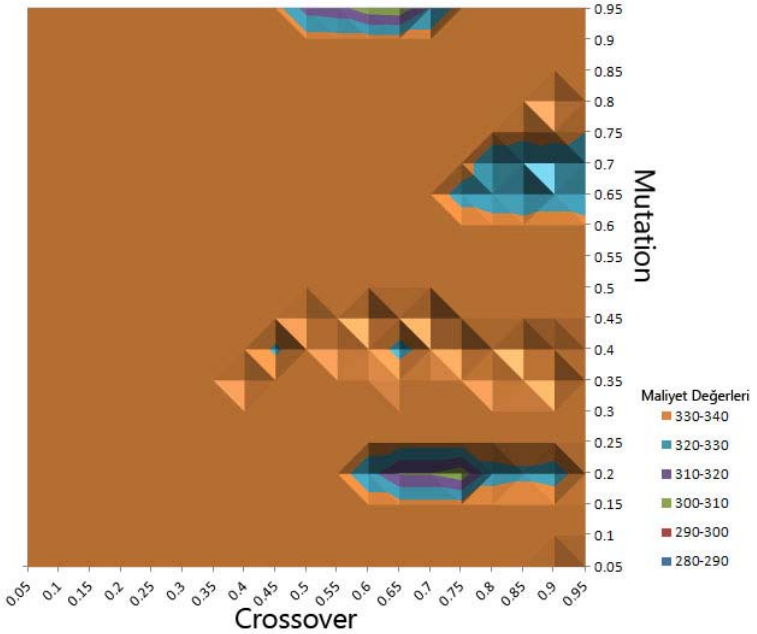
Matris Boyutu	Ortalama Zaman(ms)	Zaman Standart Sapması(%)
10x10	2.8758294	60.35%
50x50	21.958544	20.61%
100x100	58.555166	21.03%
250x250	253.0424	1.27%
500x500	1291.6872	1.19%
750x750	3733.3154	0.92%
1000x1000	15368.344	2.66%

Tablo 6 Divide & Conquer Auction Algoritması

Matris Boyutu	Ortalama Zaman(ms)	Zaman Standart Sapması(%)
10x10	144.54	38.85%
50x50	184.44	63.09%
100x100	214.04	43.35%
250x250	0	0
500x500	0	0
750x750	0	0
1000x1000	0	0

Tablo 7 Brute Force Algoritması

Matris Boyutu	Ortalama Zaman(ms)	Zaman Standart Sapması(%)
10x10	143.12	1.45%
50x50	0	0
100x100	0	0
250x250	0	0
500x500	0	0
750x750	0	0
1000x1000	0	0



Şekil 1 Genetik Algoritma Crossover-Mutation oranlarına göre maliyet sonuçları

Tablo 8 Algoritmalara göre maliyet, dengeli kaynak kullanımı ve zaman karşılaştırmaları.

Algoritma Adı	Maliyet Karşılaştırması	Kaynakların Dengeli Kullanımı	Zaman
Peer-To-Peer	*******		*******
Peer-To-Peer Modified	*	*	******
Auction	***	*	**
Munkres	*****		***
Divide & Conquer Munkres	****		*****
Divide & Conquer Auction	**	*	****
Brute Force	******		*

Tabloda verilen (*) ifadesi ve sayısı algoritmanın etkinliğini göstermektedir. Dolayısıyla (*******) en başarılı olanı ifade etmektedir. Kaynakları dengeli olarak kullanma başlığı altındaki (*) ifadesi ise ilgili algoritmaların bu işlevi başarabildiğini simgelemektedir.

Bu çalışmanın yapılmasının temel nedeni, günümüz yazılım uygulamaları için temel çalışma platformu olarak değerlendirilen bulut ağlarında kaynak kullanım optimizasyonu problemini çözmek olarak özetlenebilir. Gerçekleştirilen Ar-Ge çalışmalarının sonucunda bulut ağlarında bütçe, performans ve güvenlik problemlerini en uygun şekilde değerlendirerek çalışan kural tabanlı ve sezgisel optimizasyon tekniklerine dayalı bir dağınık önbellek yönetim yazılımı geliştirilmiştir. Bundan sonraki amaç, bu çalışmadan elde edilen deneyim ve sonuçlarla bulut ağlarına yönelik daha farklı çözüm uygulamaları geliştirmektir.

TEŞEKKÜR

Bu çalışma, Türkiye Bilimsel ve Teknik Araştırma Kurumu (TÜBİTAK) Teknoloji ve Yenilik Destek Programları Başkanlığı (TEYDEB) tarafından 7110265 numaralı sözleşme kapsamında desteklenmiştir. Yazarlar, söz konusu destekten ötürü TÜBİTAK'a teşekkürü bir borç bilir.

KAYNAKÇA

[1] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, A. Y. Wu, "An Optimal Algorithm for Approximate Nearest Neighbor Searching in Fixed Dimensions", *Journal of the ACM*, vol. 45, no. 6, pp. 891–923, 1994.

[2] D. P. Bertsekas, "An Auction Algorithm for Shortest Paths", *SIAM Journal on Optimization*, vol. 1, no. 4, pp. 425–447, 1991.

[3] J. Munkres, "Algorithms for the Assignment and Transportation Problems", *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[4] M. B. Akbulut, A. E. Yılmaz, "A Modified Genetic Algorithm for the Generalized Asssignment Problems", *IU - Journal of Electrical & Electronics Engineering*, vol. 9, no. 2, pp. 951-958, 2009.