T.C.
PAMUKKALE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

KARESEL ATAMA PROBLEMİNİN TAVLAMA BENZETİMİ VE PARALEL PROGRAMLAMA TEKNİKLERİ KULLANARAK ÇÖZÜMÜ

YÜKSEK LİSANS TEZİ

SELAHATTİN AKKAŞ

DENİZLİ, TEMMUZ - 2016

T.C.
PAMUKKALE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

KARESEL ATAMA PROBLEMİNİN TAVLAMA BENZETİMİ VE PARALEL PROGRAMLAMA TEKNİKLERİ KULLANARAK ÇÖZÜMÜ

YÜKSEK LİSANS TEZİ

SELAHATTİN AKKAŞ

DENİZLİ, TEMMUZ - 2016

KABUL VE ONAY SAYFASI

SELAHATTİN AKKAŞ tarafından hazırlanan "**KARESEL ATAMA PROBLEMİNİN TAVLAMA BENZETİMİ VE PARALEL PROGRAMLAMA TEKNİKLERİ KULLANARAK ÇÖZÜMÜ**" adlı tez çalışmasının savunma sınavı 14.07.2016 tarihinde yapılmış olup aşağıda verilen jüri tarafından oy birliği / ~~oy çokluğu~~ ile Pamukkale Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı Yüksek Lisans Tezi olarak kabul edilmiştir.

Jüri Üyeleri	İmza
Danışman Doç. Dr. Kadir KAVAKLIOĞLU	.....
Üye Yrd. Doç. Dr. Gürhan GÜNDÜZ Pamukkale Üniversitesi	.....
Üye Yrd. Doç. Dr. Hasan BULUT Ege Üniversitesi	.....

Pamukkale Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 10.08.2016 tarih ve 29/12 sayılı kararıyla onaylanmıştır.

.....

Prof.Dr. Mehmet Ali SARIGÖL

Fen Bilimleri Enstitüsü Müdürü

Bu tezin tasarımı, hazırlanması, yürütülmesi, araştırmalarının yapılması ve bulgularının analizlerinde bilimsel etiğe ve akademik kurallara özenle riayet edildiğini; bu çalışmanın doğrudan birincil ürünü olmayan bulguların, verilerin ve materyallerin bilimsel etiğe uygun olarak kaynak gösterildiğini ve alıntı yapılan çalışmalara atfedildiğine beyan ederim.

SELAHATTİN AKKAŞ

ÖZET

KARESEL ATAMA PROBLEMİNİN TAVLAMA BENZETİMİ VE PARALEL PROGRAMLAMA TEKNİKLERİ KULLANARAK ÇÖZÜMÜ
YÜKSEK LİSANS TEZİ
SELAHATTİN AKKAŞ
PAMUKKALE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
(TEZ DANIŞMANI: DOÇ. DR. KADİR KAVAKLIOĞLU)

DENİZLİ, TEMMUZ - 2016

Karesel atama problemi NP-zor sınıfında bir problem olup çözümü en zor problemlerden biridir. Problemin zorluğu nedeniyle kesin yöntemler kullanılarak boyutu büyük problemler için makul zamanda sonuç bulunamamaktadır. Bu çalışmada karesel atama problemlerinin çözümünde kullanılan meta-sezgisel yöntemlerden birisi olan tavlama benzetimi yöntemi MATLAB ortamında değişik şekillerde paralelleştirilmiştir. Paralel yöntemler ile klasik seri tavlama benzetimi yöntemi arasında süre ve iterasyon olarak karşılaştırmalar yapılmıştır. Paralelleştirme işleminde iş istasyonunda 12 MATLAB işçisi kullanılmıştır. Karşılaştırmalar örnek karesel atama problemlerinin bulunduğu bir kütüphane olan QAPLIB'den alınan 36 örnek problem üzerinde yapılmıştır. İşçiler arasında hiç haberleşmenin yapılmadığı asenkron hesaplamalı tavlama benzetimi ve belirli aralıklarla işçiler arasında veri paylaşımının yapıldığı senkron hesaplamalı tavlama benzetimi yönteminin klasik seri tavlama benzetimine göre daha iyi sonuçlar verdikleri görülmüştür.

ANAHTAR KELİMELER: Karesel Atama Problemi, Optimizasyon, Tavlama Benzetimi, Paralel Programlama

ABSTRACT

SOLVING QUADRATIC ASSIGNMENT PROBLEM USING SIMULATED ANNEALING AND PARALLEL PROGRAMMING TECHNIQUES
MSC THESIS
SELAHATTİN AKKAŞ
PAMUKKALE UNIVERSITY INSTITUTE OF SCIENCE
COMPUTER ENGINEERING
(SUPERVISOR: ASSOC. PROF. DR. KADİR KAVAKLIOĞLU)

DENİZLİ, JULY 2016

Quadratic assignment problem which is a problem under the category of NP-hard is one of the hardest problems to be solved. Because of the difficulty of the problem, it is hard to get results for big problems in a reasonable time period by using exact methods. In this study, simulated annealing method which is one of the meta-heuristic methods used in solving quadratic problems was parallelized in various categories in MATLAB. Parallel methods were compared and contrasted with classical serial simulated annealing method in terms of execution time and number of iterations. On parallelization, 12 workers were used on the workstation. Comparisons have been done for 36 sample problems taken from QAPLIB which is a library that has sample quadratic assignment problems. It has been observed that asynchronous computed simulated annealing method in which there is no communication among workers and synchronous computed simulated annealing method in which communication is done in certain intervals given better results in comparison to serial simulated annealing.

KEYWORDS: Quadratic Assignment Problem, Optimization, Simulated Annealing, Parallel Programming

İÇİNDEKİLER

Sayfa

ŞEKİL LİSTESİ

Sayfa

TABLO LİSTESİ

Sayfa

KISALTMALAR LİSTESİ

GİB : Grafik İşlem Birimi
KAP : Karesel Atama Problemi
MATLAB : Matrix Laboratory
SPMD : Tek Program Çoklu Veri

ÖNSÖZ

Yüksek lisans eğitimim ve tez çalışmam boyunca her aşamada yardımcı olan, değerli bilgilerini ve zamanını esirgemeyen tez danışmanım sayın Doç. Dr. Kadir KAVAKLIOĞLU'na teşekkür ederim.

Ayrıca benden maddi manevi desteğini hiçbir zaman esirgemeyen aileme sonsuz teşekkürü borç bilirim.

1. GİRİŞ

Karesel atama problemi (KAP) ilk olarak Koopmans ve Beckmann (1957) tarafından ekonomik faaliyetler için matematiksel bir model olarak önerilmiştir. Problemin geniş uygulama alanlarının olması ve zorluğu nedeniyle bir çok araştırmacının ilgisini çekmektedir.

Steinberg (1961) KAP'ı devre arkasındaki kabloların bağlantı sayılarını azaltmak için, Heffley (1972) ekonomik problemlerde, Francis ve White (1974) müşterileri için tesis atama problemlerinde (süpermarketler, okullar), Geoffrion ve Graves (1976) çizelgeleme problemlerinde, Pollatschek ve diğ. (1976) daktilo klavyeleri ve kontrol panelleri için en uygun tasarımı belirlemede, Krarup ve Pruzan (1978) arkeolojik problemlerde, Hubert (1987) istatistiksel analizde, Forsberg ve diğ. (1995) reaksiyon kimyası analizinde, Duman ve Or (2007) elektronik bileşenlerin yerleşiminde KAP'ı kullanmıştır. Yine de KAP'ın en popüler uygulama alanı tesis yerleşim problemidir. Dickey ve Hopkins (1972) üniversite kampüsü içerisindeki binaların atamasında, Elshafei (1977) hastane planlamasında, Bos (1993) ormanlarla ilgili bir problemde kullanmıştır.

Problemin zorluğu ve kullanım alanının geniş olması nedeniyle KAP için daha iyi çözümler veren algoritmaların geliştirilmesi oldukça önemlidir. Problemin NP-zor sınıfında olmasından dolayı en iyi çözüm polinom zamanda bulunamamakta, bu da araştırmacıları sezgisel yöntemlere yönlendirmektedir. Sezgisel yöntemlerle çözüm uzayındaki tüm çözümlere bakılmadan yüksek kalitede çözümler bulunması hedeflenmektedir. Bulunan çözümün en iyi olduğu garantisi yoktur.

Metallere ısıl işlem uygulanmasından esinlenilerek geliştirilen sezgisel yöntemlerden birisi olan tavlama benzetimi yöntemi de birçok araştırmacı tarafından KAP'a uygulanmıştır.

Literatürde tavlama benzetimiyle ilgili oldukça fazla çalışma olmasına rağmen, son yıllarda tavlama benzetimi yöntemi KAP için daha çok diğer yöntemlerle performans karşılaştırmalarında kullanılmaktadır. Paul (2010) tabu

aramasıyla, Gamal ve diğ. (2014) tabu araması ve genetik algoritmayla, Hussin ve Stützle (2014) tabu aramasıyla performans karşılaştırma için tavlama benzetimini kullanmışlardır. Bunlar dışında diğer sezgisel yöntemler ile hibritlenen tavlama benzetimi algoritmaları geliştirilmiştir. Ghandeshtani ve diğ. (2010) komşu çözüm arama kısmında tavlama benzetimini açgözlü algoritmayla hibritlemişlerdir. Wang (2012) ve Kaviani ve diğ. (2014) tabu araması ile hibritlenen bir algoritma geliştirmişlerdir. Ayrıca Paul (2012) maliyetlerin bir matriste tutulduğu GİB üzerinde çalışan bir tavlama benzetimi algoritması önermiştir.

Günümüz bilgisayarlarının çok çekirdekli yapısı paralel programlama yapmaya çok elverişlidir. Tavlama benzetimi yöntemi KAP'a çoğunlukla sıralı haliyle uygulanmaktadır. Literatürde tavlama benzetiminin KAP problemine paralel implementasyonu üzerinde çok fazla durulmamıştır. Bu çalışmada modern bilgisayarların çok çekirdekli yapısından faydalanarak MATLAB ortamında Paralel Hesaplama Araç Kutusu kullanılarak tavlama benzetimini farklı paralelleştirme yöntemleri denerek karşılaştırmalı sonuçlar verilmiştir.

Çalışmanın ilk kısmında KAP detaylı olarak açıklanmış, KAP çözümünde kullanılan yöntemler tanıtılmıştır. İkinci kısımda MATLAB ve MATLAB Paralel Hesaplama Araç Kutusu hakkında bilgi verilmiş ve son kısımda yaptığımız testlerdeki sonuçlar paylaşılmıştır.

2. KARESEL ATAMA PROBLEMİ

Karesel Atama Problemi n sayıda konuma n tane tesis atanması problemidir. Problemin polinom zamanda çözümü olmadığından KAP, NP-zor sınıfında yer almaktadır.

KAP ilk olarak Koopmans ve Beckmann (1957) tarafından ekonomik faaliyetlerin modellenmesi için önerilmiştir. Boyutu n olan bir problem için n! tane farklı çözüm vardır. Boyutu n > 30 olduğunda makul bir zamanda tüm çözümler denenememektedir (Loiola ve diğ. 2007).

2.1 Matematiksel Model

Problemde n adet konum ve n adet tesis bulunmaktadır. Tesis *i* ile *j* arasındaki akış f_{ij} olarak ve *k* ile *p* konumları arasındaki uzaklık d_{kp} olarak ifade edilirse, KAP problemi aşağıdaki optimizasyon problemi olarak ifade edilebilir:

$$\begin{aligned} min \quad & \sum_{i,j=1}^{n} \sum_{k,p=1}^{n} f_{ij} d_{kp} x_{ik} x_{jp} \\ kısıtlar \quad & \sum_{i=1}^{n} x_{ij} = 1, \;\; 1 \leq j \leq n \\ & \sum_{j=1}^{n} x_{ij} = 1, \;\; 1 \leq i \leq n \\ & x_{ij} \in \{0,1\}, \;\; 1 \leq i, j \leq n \end{aligned} \tag{2.1}$$

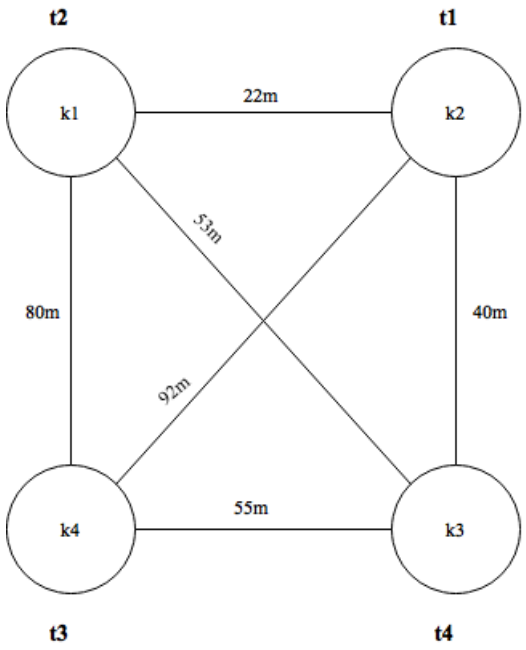
Burada, x herhangi bir olası çözümü temsil etmekte olup hangi konuma hangi tesisin atandığını gösterir ve aslında bir permütasyondur. Standart uygulama x'in değerinin eşleşme olan konum-tesis çiftleri için "1" ve diğerleri için "0" olmasıdır. Burada hedeflenen (2.1) eşitliğinde verilen amaç fonksiyonunun kısıtları sağlayacak şekilde minimize edilmesidir.

2.2 Maliyet Hesabı

Karesel atama probleminde n tane konum ve n tane tesis bulunmaktadır. Her konuma bir tesis atanmakta, hiçbir tesis ve konum boşta kalmamaktadır. Konumlar arasındaki uzaklıklar ve tesisler arası akış verilmektedir.

$$d = \begin{bmatrix} 0 & 22 & 53 & 80 \\ 22 & 0 & 40 & 92 \\ 53 & 40 & 0 & 55 \\ 80 & 92 & 55 & 0 \end{bmatrix} \qquad f = \begin{bmatrix} 0 & 3 & 0 & 2 \\ 3 & 0 & 0 & 1 \\ 0 & 0 & 0 & 4 \\ 2 & 1 & 4 & 0 \end{bmatrix}$$

Şekil 2.1: Örnek KAP uzaklık (d) ve akış (f) matrisi



Şekil 2.2: Örnek bir KAP

Şekil 2.2'de örnek bir KAP verilmiştir. Örnekte dört tane konum (k1,k2,k3,k4) ve dört tane tesis (t1,t2,t3,t4) bulunmaktadır. k1'e t2, k2'ye t1, k3'e t4 ve k4'e t3 atanmıştır. d matrisi konumlar arası uzaklıkları ve f matrisi tesisler arası akışları göstermektedir. Atama işlemi vektör olarak gösterilmektedir. Örnekteki atama için çözüm vektörü: $s = [2 \quad 1 \quad 4 \quad 3]$' dir.

Elimizdeki çözüm vektörü, d ve f matrisleri kullanılarak (2.1) eşitliğine göre maliyet hesaplandığında (2.2) eşitliğindeki sonuç elde edilir. Örnek çözüm vektörü

için bulunan sonuç 838'dir. Atamalar değiştirilerek daha iyi bir çözümün olup olmadığına bakılır.

$$C = f_{2\,2}d_{1\,1} + f_{2\,1}d_{1\,2} + f_{2\,4}d_{1\,3} + f_{2\,3}d_{1\,4} + f_{1\,2}d_{2\,1} + f_{1\,1}d_{2\,2} + f_{1\,4}d_{2\,3} + f_{1\,3}d_{2\,4} + f_{4\,2}d_{3\,1} + f_{4\,1}d_{3\,2} + f_{4\,4}d_{3\,3} + f_{4\,3}d_{3\,4} + f_{3\,2}d_{4\,1} + f_{3\,1}d_{4\,2} + f_{3\,4}d_{4\,3} + f_{3\,3}d_{4\,4} \quad (2.2)$$

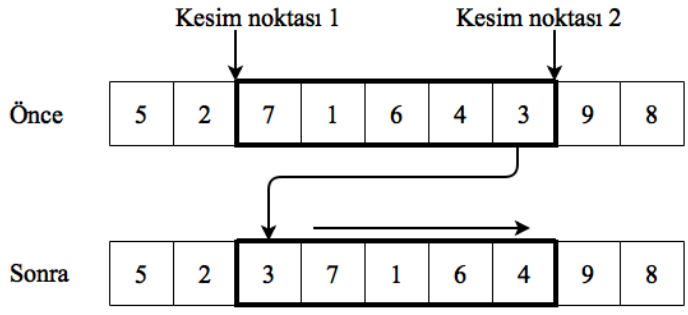
$$C = 0*0 + 3*22 + 1*53 + 0*80 + 3*22 + 0*0 + 2*40 + 0*92 + 1*53 + 2*40 + 0*0 + 4*55 + 0*80 + 0*92 + 4*55 + 0*0 = 838$$

2.3 Yeni Çözüm Oluşturma Yöntemleri

KAP'ta kullanılan yeni çözüm oluşturma yöntemleri kaydırma, yer değiştirme ve tersine çevirmedir.

2.3.1 Kaydırma

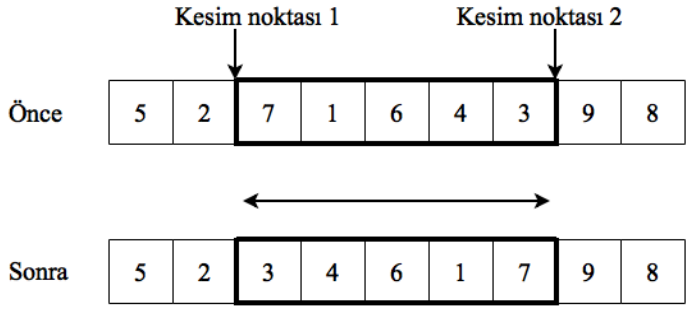
Kaydırma yönteminde rastgele iki kesim noktası seçilerek bu noktalar arasındaki atamalar istenildiği kadar kaydırılır. Şekil 2.3'te örnek bir kaydırma işlemi verilmiştir. Kesim noktası 1 ve kesim noktası 2 arasındaki değerlere sağa doğru bir birim kaydırma işlemi yapılmıştır.



Şekil 2.3: Kaydırma

2.3.2 Ters Çevirme

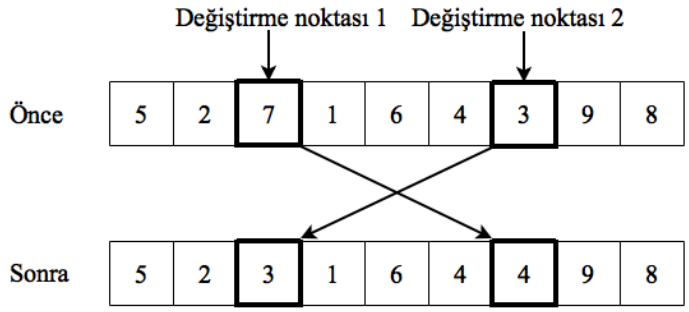
Şekil 2.4'te görüldüğü gibi ters çevirme yönteminde rastgele iki kesim noktası seçilir ve noktalar arasındaki atamalar tersine çevrilir.



Şekil 2.4: Ters çevirme

2.3.3 Yer Değiştirme

Yer değiştirme yönteminde rastgele iki nokta seçilerek bu noktalardaki atamalar yer değiştirilir. En çok kullanılan yöntemdir. Diğer yöntemler ilk başlarda çeşitlilik oluşturması, başlangıç çözümüne bağlı kalmama gibi yönlerden iyi olsa da, en iyi çözüme yaklaşma konusunda başarısız olmaktadır. Bu yöntem kullanıldığında ise sadece iki konumun yeri değiştirilmesine rağmen, tekrarlı yapıldığında daha iyi çözümler sunmaktadır. Tian ve diğ. (1999) yaptıkları çalışmada KAP için en iyi sonuçları yer değiştirme yöntemiyle aldıklarını belirtmişlerdir.



Şekil 2.5: Yer değiştirme

2.4 KAP Çözümünde Kullanılan Yöntemler

KAP çözümünde kullanılan yöntemler kesin çözüm yöntemleri ve sezgisel yöntemler ve meta-sezgisel olmak üzere üç kategoriye ayrılmaktadır.

2.4.1 Kesin Çözüm Yöntemleri

Kesin çözüm yöntemleri çözüm kümesindeki tüm çözümleri deneyerek en iyi çözümü bulmak için kullanılan yöntemlerdir. Karesel atama problemlerinde global optimum değeri bulabilmek için dallanma ve sınırlandırma algoritması, kesme düzlemi ve bu ikisinin birleşimi olan dallanma ve kesme, dinamik programlama yaklaşımları kullanılmaktadır (Loiola ve diğ. 2007). Kesin çözüm yöntemlerinde tüm çözümler denenmesi çok zaman almaktadır. Bu yüzden boyut olarak küçük problemlere uygulanabilmektedir.

2.4.1.1 Dallanma ve Sınırlandırma Yöntemi

Dallandırma ve sınırlandırma yöntemi en iyi bilinen ve en fazla kullanılan kesin çözüm algoritmasıdır. Literatürde istenmeyen sonuçları sınırlandıran ilk çalışmalar Gilmore (1962) ve Lawler (1963) tarafından yapılmıştır. Son yıllarda dallandırma ve sınırlandırma algoritmalarının paralel uygulamalarının kullanılması oldukça yaygındır. Bu yüzden en iyi sonuçlar son yıllarda alınmıştır. Yine de daha büyük örneklerde başarılı sonuçlar alınması donanımdaki gelişmelerle yakından ilişkilidir (Loiola ve diğ. 2007).

Dallanma ve sınırlandırma yönteminde uygun çözümler uzayı dinamik olarak ufak bölgelere parçalanır. Oluşan her alt problem karar ağacındaki bir düğüm tarafından temsil edilmektedir. Her düğüm için bulunabilecek en iyi çözüm, alt problemlerin çözümlerinin doğrusal programlama hafifletmesi yardımıyla çözülmesiyle bulunur. Bulunan sonuç o ana kadar bulunan en iyi sonuçtan daha kötüyse dalda arama sonlandırılır ve başka bir düğüme geçilir. Eğer daha iyi sonuç bulunmuşsa mevcut düğüm alt dallara bölünür, daha küçük uygun çözüm uzayının

olduğu alt problemler oluşturulur. Bu işlem oluşturulabilecek dal kalmayıncaya kadar devam eder (Burkard ve diğ. 2009).

2.4.1.2 Kesme Düzlemi Yöntemi

Kesme düzlemi yöntemi ilk olarak Bazaraa ve Sherali (1980) tarafından tanıtılmıştır. İlk başlarda tatmin edici sonuçlar vermemiştir. Ancak, Bender ayrıştırması ve karma tam sayılı doğrusal programlama yöntemini kullanan bazı sezgisel algoritmaların formülasyonuna katkı sağlamıştır. Bu yöntemin optimal çözüme yavaş yakınsaması, yöntemi sadece ufak problemler için uygun kılmıştır (Loiola ve diğ. 2007).

Bu yöntemde karma tam sayılı doğrusal programlama formülasyonu ana problem ve yan problem olmak üzere ikiye bölünür. Ana problem orijinal atama değerlerini içerirken, yan problem de amaç fonksiyonunu doğrusallaştırma için eklenen değişkenleri içerir. Bir başlangıç çözümü için yan problem bir doğrusal programdır ve polinom zamanda çözülebilirdir. Bu problem çözülür ve yan problemin değişkenlerinin optimum değerleri belirlenir. Ana problem orijinal atama değerleri ve yan problemin değişkenleri kullanılarak doğrusal program olarak ifade edilir. Sonuç olarak, sabit yan problem değerleri için ana problem polinom zamanda çözülebilir hale gelir (Çela 1998).

2.4.1.3 Dallanma ve Kesme Yöntemi

Dallanma ve kesme yöntemi ilk olarak Padberg ve Rinaldi (1991) tarafından önerilmiştir. Yöntem dallanma ve sınırlandırma yöntemi ile kesme düzlemi yönteminin birleştirilmesiyle oluşturulmuştur. Kesme düzlemindeki doğrusal programlama metotlarına benzer şekilde dallanma ve kesme yöntemi problemin doğrusal hafifletilmesi üzerine oluşturulmuştur. Ek olarak orijinal problem için uygun tüm çözümleri tutmak için geçerli eşitsizlikler kullanılır.

Karar ağacındaki bir düğümde, hafifletilmiş alt problemin çözümü orijinal problem için uygun bir çözümse, düğümün keşfi tamamlanır. Uygun bir çözüm

değilse, eşitsizliklerden bazıları ihlal edilmişse kesme işlemi uygulanır: bir veya daha fazla ihlal edilen eşitsizlikler alt problem hafifletmesine eklenir ve yeni bir çözüm elde edilir. Eşitsizliklerde hiç birisi ihlal edilmemişse algoritma dallanma aşamasına geçer (Burkard ve diğ. 2009).

2.4.1.4 Dinamik Programlama

Dinamik programlama yöntemi akış matrisinin bir ağacın komşuluk matrisi olduğu özel durumlarda kullanılan bir tekniktir. Christofides ve Benavent (1989) bu özel durumu hafifletilmiş problemde karma tam sayı doğrusal programlama tekniğini kullanarak çalışmışlardır. Örneklerin bu özel durumunu kullanarak dinamik programlama algoritmasını uygulamışlardır.

2.4.2 Sezgisel Yöntemler

Karesel atama probleminin zorluğu ve tüm çözümleri denemenin zorluğu araştırmacıları sezgisel yöntemlere yönlendirmiştir. Sezgisel yöntemlerde tüm çözümleri denemek yerine bunlardan muhtemel iyi sonuç verecek çözümler denenip bu sonuçlardan en iyisi alınmaktır.

Sezgisel yöntemler bulunan çözümün en iyi olduğunu garanti etmezler. Sezgisel yöntemler en iyi sonucu bulan yöntemler değil, yüksek kaliteli çözümler üreten yöntemler olarak adlandırabilir. Genel olarak yapıcı yöntemler, enümerasyon yöntemleri ve geliştirme yöntemleri olarak üç ayrı kategoride incelenmektedir.

Yapıcı yöntem Gilmore (1962) tarafından önerilmiştir. Yöntemde algoritmanın her adımında bir permütasyon tamamlanır. Bu yöntemde ilk başta boş olan atanan tesislerin ve dolu konumların tutulduğu iki küme kullanılır. Her adımda bir tesis bir konuma atanır ve bu işlem tüm tesisler bir konuma atanıncaya kadar devam eder (Loiola ve diğ. 2007).

Enümerasyon yöntemleri dallanma ve sınırlandırma veya kesme düzlemi yöntemine oldukça benzemektedir. Bu yöntem temel olarak dallanma ve

sınırlandırma yönteminde bulunan iyi bir çözümü temel almaktadır. Optimum ya da optimuma yakın çözüm genelde algoritmanın erken safhalarında bulunmaktadır. Sonrasında geçen zamanın çoğu bulunan çözümün optimum olduğunu kanıtlamaya veya erken safhada bulunan çözümü iyileştirmeye harcanmaktadır. Enümerasyon yöntemleri kabul edilebilir zamanda iyi çözümler bulmayı hedeflemektedir (Çela 1998). Algoritmada sonlandırma koşulu olarak zaman limiti veya iterasyon limiti kullanılmaktadır.

Geliştirme yöntemleri yerel arama algoritmaları kategorisine girmektedir. Yerel arama algoritmalarında bir başlangıç çözümüyle başlanılır. Mevcut çözümün komşularında mevcut çözümden daha iyi bir çözüm bulununcaya kadar arama işlemi devam eder. Bu işlem hiç iyileştirme bulunamayıncaya kadar devam eder. Komşu çözümler iki tesisin konumlarının değiştirilmesiyle elde edilmektedir (Çela 1998).

2.4.3 Meta-sezgisel Yöntemler

Sezgisel yöntemler daha çok probleme dayalı geliştirilmiştir ve her probleme uygulanamamaktadır. Ayrıca sezgisel yöntemler genelde çözümü hep iyileştiren yeni çözümler bulmaya odaklanmakta ve bu da yerel minimumlara takılmaya sebep olmaktadır. Diğer yandan meta-sezgisel yöntemler bazı aşamalarda kötü çözümleri de kabul etmekte, yerel minimumlardan kurtulmaya çalışmaktadır.

Karesel atama probleminde en çok kullanılan meta-sezgisel yöntemler tavlama benzetimi, genetik algoritma, tabu araması, dağınık arama, açgözlü rassallaştırılmış uyarlamalı arama yordamı ve karınca kolonisi optimizasyonudur. Bu yöntemlerin çoğu doğal süreçlerden esinlenilerek geliştirilmiştir.

2.4.3.1 Tavlama Benzetimi

Tavlama benzetimi yöntemi aşırı ısıtılmış metallerin termal soğumasının simülasyonuna dayalı bir optimizasyon yöntemidir. Bir metal erime durumuna gelecek kadar ısıtıldığında, erimiş metaldeki atomlar serbestçe hareket eder. Sıcaklık azaltıldığında ise atomların hareketi kısıtlanır. Sıcaklık düştükçe atomlar düzenli

hale girme eğilimindedir ve uygun sıcaklık profili kullanıldığında, son durumda kristaller minimum olasılıksal içsel enerjiye sahiptirler. Kristallerin dizilimi soğuma oranına bağlı olarak değişmektedir. Hızlı soğutulduğunda metaller kristal durumuna gelemeyip, metalde kusurlar görülebilir. Metalin yüksek sıcaklıklardan yavaş soğutulması işlemine tavlama denilmektedir (Rao 2009).

Tavlama benzetimi fikri Metropolis ve diğ. (1953) tarafından katı metaller için önerilmiştir. Kirkpatrick ve diğ. (1983) Metropolis algoritmasını optimizasyon problemlerine uygulamıştır.

Optimizasyon problemlerinde tavlama benzetimi yöntemi katı metallerdekine benzer şekildedir. Sıcaklık yüksek iken algoritmada farklı çözümler arasında serbestçe geçiş yapılabilmektedir. Yeni bulunan çözüm kötü olsa da kabul edilebilmektedir. Sıcaklık azaldıkça yeni çözüm daha iyiyse kabul edilmekte, kötü çözümlerin kabul edilmesi ise olasılıksal olarak azalmaktadır.

Kötü çözümlerin kabulü Boltzmann'ın olasılık dağılımına göre yapılmaktadır. Termal dengedeki bir sistemin T sıcaklığındaki enerjisi (E) (2.3) eşitliğine göre olasılıksal olarak dağıtılır. Burada P(E) enerji seviyesi, k Boltzmann sabiti olarak adlandırılır. Sistem yüksek sıcaklıklarda iken herhangi bir enerji durumunda olabilirken, sıcaklık azaldıkça yüksek enerji durumunda olma olasılığı çok azdır.

$$P(E) = e^{-E/kT} \tag{2.3}$$

Tavlama benzetimi algoritması aşağıdaki gibi özetlenebilir:

Adım 1: Başlangıç sıcaklığı T , soğuma katsayısı α belirle.

Adım 2: Rastgele bir başlangıç çözümü x oluştur.

Adım 3: x çözümünde iki tesisin yerini değiştirerek yeni çözüm x' oluştur.

Adım 4: Eğer yeni çözüm daha iyiyse yeni çözümü kabul et: x=x'. Eğer yeni çözüm daha kötüyse 0 ile 1 arasında rastgele bir sayı

belirle. Belirlenen sayı (2.3) eşitliğinden küçükse yeni çözümü kabul et: x=x'.

Adım 5: Aynı sıcaklıktaki adım sayısı tamamlanmadıysa Adım 3'e git.

Adım 6: Sıcaklığı α'ya göre güncelle: T=T* α.

Adım 7: Eğer sıcaklık 1'in altına düştüyse sıcaklığı ilk sıcaklığa yükselt.

Adım 8: Sonlandırma kriteri sağlanmıyorsa Adım 3'e git.

Algoritmada belirtilen sonlandırma kriteri olarak belirli bir iterasyon sayısı, iki iterasyon arası çözümdeki değişmenin belirli bir değerden az olması kullanılabilir.

Tavlama benzetimi yönteminde soğuma farklı şekillerde yapılabilir. Bunlardan en bilinenleri doğrusal soğuma ve üstel soğumadır. Doğrusal soğumada sıcaklık her adımda aynı oranda azalır. Üstel soğumada ise sıcaklık üstel olarak azalmaktadır. Yüksek sıcaklıklarda daha az adım gerçekleştirirken, düşük sıcaklıklarda daha fazla adım gerçekleştirmektedir.

Tavlama benzetiminde dikkat edilmesi gereken en önemli kriterlerden biri başlangıç sıcaklığının belirlenmesidir. Başlangıç sıcaklığı çok yüksek seçildiğinde, çözüm iyi veya kötü olsa da kabul edilecektir. Yüksek sıcaklıklarda çok fazla durmanın faydası olmayacaktır. Başlangıç sıcaklığı düşük seçildiğinde ise yerel minimumlara takılmalar olmaktadır. Bu yüzden başlangıç sıcaklığı seçimi oldukça önemlidir. İdeal olarak önerilen yöntem sistemi kötü çözümleri belirli oranda kabul eden bir sıcaklığa getirip yavaş soğuma yapmaktadır. Bu işlem metallerde yapılan fiziksel işleme de oldukça benzemektedir. Metaller sıvı hale geçinceye kadar ısıtılmakta, sonrasında yavaş soğuma işlemi yapılmaktadır (Kendall 2016).

Dikkat edilmesi gereken bir diğer kriter ise soğuma katsayısıdır. Eğer doğrusal soğuma yapılıyorsa her adımda sıcaklığın ne kadar azalacağına iyi karar verilmelidir. Eğer üstel soğuma yapılıyorsa $0 < \alpha < 1$ arasında olmalıdır. Burada α büyük seçilirse soğuma çok yavaş olmakta, çok küçük seçilirse de soğuma çok hızlı olmaktadır. Deneyler α'nın 0,8 ile 0,99 arasında seçildiğinde daha iyi sonuçlar

verdiğini göstermektedir (Kendall 2016). Ancak α değeri büyüdükçe iterasyon sayısının da artacağı unutulmamalıdır.

Tavlama benzetiminde son sıcaklık ta önemlidir. Genelde sıcaklık 0 dereceye düşene kadar soğutma işlemi yapılmaktadır. Ancak bu işlem yöntemin oldukça uzun çalışmasına sebep olabilir. Uygulamada, sıcaklığın 0'a ulaşması gerekli değildir. Çünkü sıcaklık 0'a yaklaştıkça kötü çözümleri kabul etme olasılığı neredeyse 0'dır. Ayrıca üstel soğuma için sıcaklık 0'a gelemeyeceği de unutulmamalıdır.

2.4.3.2 Genetik Algoritma

Genetik algoritma evrim teorisindeki doğal seçilimin modellenmesiyle oluşturulan bir meta-sezgisel optimizasyon yöntemidir. Popülasyondaki bireylerin çaprazlanıp güçlü olanların nesillerini devam ettirmesi modellenmiştir. İlk olarak Holland (1975) tarafından önerilmiştir. Son zamanlarda değişik kombinatoryal optimizasyon problemleri için birçok araştırmacı genetik algoritma türevlerini kullanmaktadır (Çela 1998).

KAP'ta popülasyon rastgele belirlenen başlangıç çözümleriyle oluşturulur. Bu çözümlere başlangıç popülasyonu denilmektedir. Popülasyondaki her bir çözüme de birey denilmektedir. Çözümlerin maliyet değerlerine göre hangi çözümlerin güçlü olduğu belirlenir. Bireylerden çiftler seçilerek çaprazlama denilen işleme göre yeni bireyler oluşturulur. Oluşturulan bu yeni bireylere de çocuk denilmektedir. Popülasyondan kötü çözümlerin bir kısmı atılır. Daha sonra bu işlem durma kriteri sağlanana kadar tekrarlanır. Algoritmanın daha iyi sonuçlar verebilmesi için mutasyon ve göç gibi özellikler de eklenebilir.

Algoritmanın adımları aşağıdaki gibi özetlenebilir:

Adım 1: Popülasyondan çiftler halinde bireyler seç.

Adım 2: Çaprazlama işlemi uygulayarak yeni birler oluştur.

Adım 3: Olasılıksal olarak bazı çocuklara mutasyon işlemi uygula.

Adım 4: Olasılıksal olarak popülasyonda olmayan yeni bireyler oluştur.

Adım 5: Popülasyona eklenecek ve çıkarılacak bireyler seç (Burkard ve diğ. 2009).

2.4.3.3 Tabu Araması

Tabu araması metodu ilk olarak Glover (1989) tarafından önerilmiştir. Tabu araması yerel minimumlara takılmamak için oldukça basit ve sade bir kuralı vardır: bir çözümün komşularında daha iyi çözüm olmadığında daha önceden denenmemiş kötüleştirici bir çözümü seçer. Tüm denenmiş çözümleri tutmak ve karşılaştırmak mümkün olmadığından Glover her çözüm için bazı özelliklerin tutulduğu tabu listesi önermiştir. Çoğu durumda listede yeni çözüm oluşturmak için yapılan hareket tutulmaktadır. KAP'ta ikili yer değiştirme yöntemi kullanılıyorsa yeri değiştirilen iki tesis tabu listesinde tutulmaktadır. Bu şekilde sonraki adımlarda tekrardan aynı iki tesisin yeri değiştirilecek olursa bu işlem listede olduğu için engellenmektedir (Burkard ve diğ. 2009). Tabu listesinde de bazı iyi hamlelerin listede olduğundan engellenmesi durumuyla başa çıkmak için de bazı hamleler listeden çıkarılmaktadır. Bunun için de tabu listesinin belirli bir uzunluğu vardır ve listeye ilk girenin ilk çıktığı yaklaşım kullanılır.

2.4.3.4 Dağınık Arama

Dağınık arama Glover (1977) tarafından önerilen tamsayı doğrusal programlama problemleri için önerilen bir yöntemdir. Genetik algoritmadaki popülasyon konseptini kullanır. Ancak üç farklı yönden genetik algoritmadan farklılaşır. Bunlardan ilki yüksek kaliteli çözümleri tutarken popülasyondaki çeşitliliği kaybetmemek için kümeleme teknikleri kullanır. İkinci farklılığı genetik algoritmada yeni çözüm oluşturmak için popülasyondaki iki çözüm kullanılırken, dağınık aramada ise böyle bir limit yoktur. Üçüncü farklılığı ise popülasyon boyutu genetik algoritmaya göre oldukça azdır (Burkard ve diğ. 2009).

2.4.3.5 Açgözlü Rassallaştırılmış Uyarlamalı Arama Yordamı

Açgözlü rassallaştırılmış uyarlamalı arama yordamı (GRASP), her aşamada yaklaşık bir çözümün bulunduğu rastgele ve iteratif bir yöntemdir. Son çözüm tüm iterasyonlar içindeki en iyi çözümdür. Her aşamada, ilk çözüm rassal açgözlü fonksiyon yardımıyla bulunur. Sonraki çözümler, önceki çözümlere yerel arama algoritması uygulanarak daha iyi bir çözüm elde edilir. Tüm iterasyonların sonunda bulunan çözüm elde edilen en iyi çözümdür. Yöntemin yerel minimuma takılmama garantisi yoktur. Bu sebeple uygun veri yapısı ve uygun uyarlamalar daha verimli bir yerel arama sağlar (Loiola ve diğ. 2007).

2.4.3.6 Karınca Kolonisi Optimizasyonu

Karınca kolonisi algoritması, karıncaların yuvaları ile yiyecek olan yerler arasındaki en kısa yolu bulma yeteneklerinden esinlenilerek geliştirilmiştir. Alternatif yolların bulunduğu durumlarda karıncalar, başlangıçta bu yollara aynı oranda dağılırlarken, belli bir süre sonunda en kısa olan yol üzerinde yoğunlaşmaktadır. Zaman ilerledikçe tüm karıncalar en kısa yolu kullanmaktadır. Bunu diğer karıncaların geçişleri sırasunda salgıladıkları feromon vasıtasıyla yapmaktadırlar. Temel kural, feromon miktarının yoğun olan yolun kullanılma olasılığının daha fazla olmasıdır. Görme yetenekleri çok gelişmemiş olan karıncalar yol seçimini feromon izlerine göre yapmaktadırlar.

Kısa olan yolda feromon miktarı uzun yollara göre daha fazla birikmektedir. Kısa yoldan geçiş daha hızlı olacağından, birim zamanda geçen karınca sayısı uzun yola göre daha fazla olacaktır. Dolayısıyla herhangi iki konum arasındaki yol üzerinde bulunan feromon miktarı yolun uzunluğu arttıkça azalmaktadır (Timur ve Söyler 2006).

3. TEZ KAPSAMINDA KULLANILAN TEKNOLOJİLER

Karesel atama problemi çözümünde matris işlemleri çok fazla yapıldığından çalışmada MATLAB tercih edilmiştir. Ayrıca MATLAB Paralel Hesaplama Araç Kutusu kullanılarak paralel işlemler yapmak oldukça kolaydır. Tezde kullanılan kilit teknoloji Paralel hesaplama Araç Kutusu olduğundan aşağıda detaylıca tanıtılmıştır.

3.1 MATLAB Paralel Hesaplama Araç Kutusu

MATLAB ile paralel programlama MATLAB Paralel Hesaplama Araç Kutusu aracılığıyla yapılmaktadır. MATLAB paralel programlama bize yoğun veri ve hesaplamanın olduğu problemleri çok çekirdekli işlemcileri, GİB ve bilgisayar kümelerini kullanarak çözmeye olanak tanır (Mathworks 2016).

Araç kutusu uygulamaları yerel makinedeki işçiler (MATLAB hesaplayıcıları) üzerinde çalıştırarak çok çekirdekli bilgisayarların tam gücünü kullanmamızı sağlar. Kodda herhangi bir değişiklik yapmadan, aynı uygulama bilgisayar kümeleri ve grid hesaplama servisleri üzerinde de çalıştırılabilir.

Temel Özellikleri:

- Paralel for döngüleri (parfor) vardır. Paralel for döngüsüyle döngüdeki işlemler işlemcilere dağıtılır.
- CUDA özelliği olan NVIDIA GİB'leri destekler. CUDA platformu grafik işlem birimi üzerinde paralel hesaplama yapmaya yarar. Hesaplamalar işlemciye göre çok daha hızlı olmaktadır.
- Çok çekirdekli işlemcileri tam kullanmaya olanak sağlar. Paralel olmayan kodlar tek çekirdek üzerinde çalışmaktadır. Araç kutusu ile tüm çekirdekler aktif olarak kullanılabilir.
- Bilgisayar kümelerini ve grid hesaplama servislerini destekler.
- Paralel uygulamalar etkileşimli ve yığın olarak çalıştırılabilir. Etkileşimli çalıştırıldığında değişkenlerdeki değerlere ulaşılabilir,

değişiklik yapılabilir; yığın ise kodun çalışması bitene kadar uygulama ile ilgili hiçbir işlem yapılamaz.

- Büyük veriler için dağıtık dizi ve tek program-çok veri özelliği vardır. Veri tek diziye sığmadığı durumlarda dağıtık diziler kullanılır. Tek program-çok veri ise aynı uygulamanın farklı veriler üzerinde çalışıp sonucu hesaplamasını sağlar.

3.1.1 MATLAB Paralelleştirme Ortamları

MATLAB'da pmode ve paralel havuz olmak üzere iki farklı paralelleştirme ortamı vardır.

3.1.1.1 pmode Paralelleştirmesi

Birden çok işçi üzerinde etkileşimli olarak çalışmaya olanak sağlar. Girilen komut tüm işçiler üzerinde eş zamanlı çalışır. Her işçinin kendine ait değişkenleri ve çalışma alanı vardır.

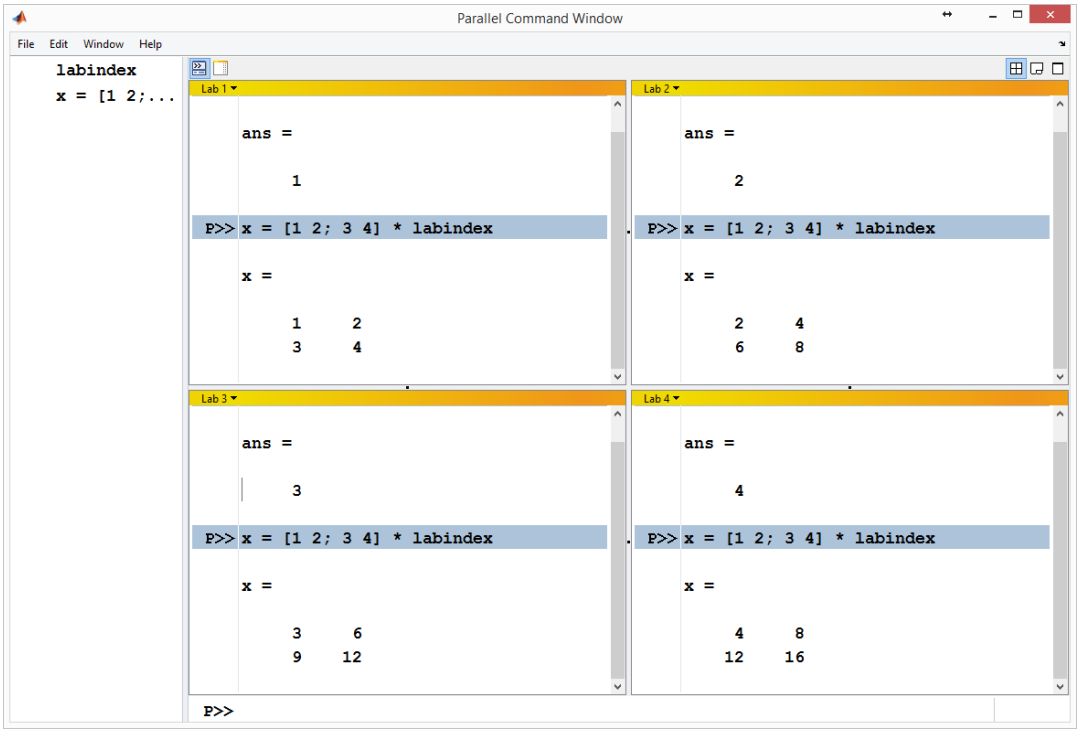
Bir işçi kendi komutunu bitirdiğinde boş durumuna geçerek diğer işçilerin komutlarını bitirmesini bekler. Bütün işçilerin kodu bittiğinde bir sonraki komut bütün işçiler üzerinde çalıştırılmaya başlanır.

pmode paralelleştirmesinde her işçinin değişkenlerini, sonuçlarını görmemize olanak sağlanır. Ancak seri ve paralel işler arasında serbestçe geçişe izin vermez. pmode başlatıldığında mutlaka paralel işlemler çalıştırılır. pmode'dan çıkıldığında çalışan görevler, işçilere ait bilgiler ve veriler kaybedilir. Yeniden pmode başlatıldığında hiçbir verinin olmadığı komutun hiçbir komutun çalıştırılmadığı temiz durumdan başlar.

pmode'da grafik çizdirme gibi işlemler yapılamaz. Grafik çizdirilmek istendiği durumlarda *lab2client* komutu ile pmode çalışma alanındaki değişkenler paralel olmayan çalışma alanına aktarılır ve çizim orada yapılır.

"*pmode start local n*" komutuyla çalıştırılır. Burada n kaç tane paralel işçi çalıştırılmak istendiğini belirtir. n bilgisayardaki fiziksel çekirdek sayısından fazla olamaz.

pmode'da tek bir komut girişi olup girilen kod tüm işçilerde çalıştığından işçilerin farklı işlemler yapması için dağıtık diziler ve *numlab*, *labindex* gibi komutlar kullanılır. Dağıtık dizide dizinin sütunları eşit parçalara ayrılır. *getLocalPart(dagitikDizi)* komutuyla dizinin o işçi için ayrılan parçasına ulaşılır. *numlab* komutu paralel modda kaç tane işçinin çalıştığı sayısını döndürür. *labindex* komutunda ise çalışılan işçinin numarası döner. Bu komutlar yardımıyla her işçinin farklı veriler üzerinde çalışması sağlanır. Şekil 3.1'de pmode için örnek bir paralel programlama penceresi verilmiştir. Her işçide x matrisi olmasına rağmen her işçinin x matrisinde farklı değerler vardır.



Şekil 3.1: pmode paralelleştirmesi ekran görüntüsü

3.1.2 Paralel Havuz

Paralel havuz, bir MATLAB hesaplama işçileri kümesidir. Varsayılan olarak, parfor gibi bir paralel hesaplama kodu gördüğünde otomatik olarak paralel havuz oluşturulur.

Paralel havuzdaki işçiler etkileşimli olarak kullanılabilir ve işin çalışması sırasında birbirleriyle haberleşebilirler. Havuzdaki işçiler paralel programlamada etkileşimli olarak kullanım için rezerve edildiklerinden, diğer istemciler tarafından kullanılamaz. Havuzdaki işçiler bir zaman diliminde sadece tek bir istemci tarafından kullanılabilir (Mathworks 2016).

pmode'dan farkı her işçi için bir çalışma alanı olmayışıdır. Ancak pmode oturumu sonlandığında değişkenleri kaybolmasına rağmen, paralel havuzda değişkenler erişilebilirdir. Ayrıca, istenildiği zaman paralel oturum başlatılıp sonlandırılması, çalıştırılan kodun bir kısmının seri, bir kısmının paralel çalıştırılması gibi avantajları vardır.

Paralel havuzda çalışan en temel üç kod parfor, spmd ve parfeval'dir.

3.1.2.1 parfor

for döngüsünün paralel halidir. Paralelleştirme işlemi otomatik olarak yapılır. Her işçi iterasyonların belirli bir kısmını yapar. parfor kullanırken dikkat edilmesi gereken en önemli nokta iterasyonların birbirinden bağımsız olmasıdır. Bir iterasyonun sonucu diğerini etkilediği durumlarda parfor kullanılamamaktadır.

parfor'un iki yaygın kullanımı Şekil 3.2 ve Şekil 3.3'te verilmiştir. Şekil 3.2'de verilen kodda kümülatif toplama işlemi yapılmaktadır. Bir iterasyondaki sonuç diğerini etkilememektedir. Şekil 3.3'te verilen kodda ise sonuçlar vektörde iterasyona göre uygun indekste saklanmakta ve aynı şekilde bir iterasyonun sonucu diğerini etkilememektedir. Bu işleme birbirine bağlama denilmektedir. İşlem paralel olduğu için önce hangisinin hesaplanacağı belirli değildir. Vektörün 7. elemanının

sonucu 3. elemanından önce hesaplanabilir. Ancak hangisinin önce hesaplandığı sonuç açısından bir fark oluşturmamaktadır.

```
x = 0;
parfor i = 1:10
    x = x + i;
end
x
x =
          55
```

Şekil 3.2: parfor kümülatif toplam kodu

```
x = [];
parfor i = 1:8
     x = [x, i];
end
x
x =
         1  2  3   4   5   6   7   8   9   10
```

Şekil 3.3: parfor birbirine bağlama kodu

parfor komutunun kullanılabilmesi için işlemin (3.1) eşitliğinde verilen koşul sağlanmalıdır. Toplama ve çarpma işlemi bu koşu sağlarken, çıkarma ve bölme işlemi bu koşulu sağlamamaktadır.

$$x \diamond (y \diamond z) = (x \diamond y) \diamond z \tag{3.1}$$

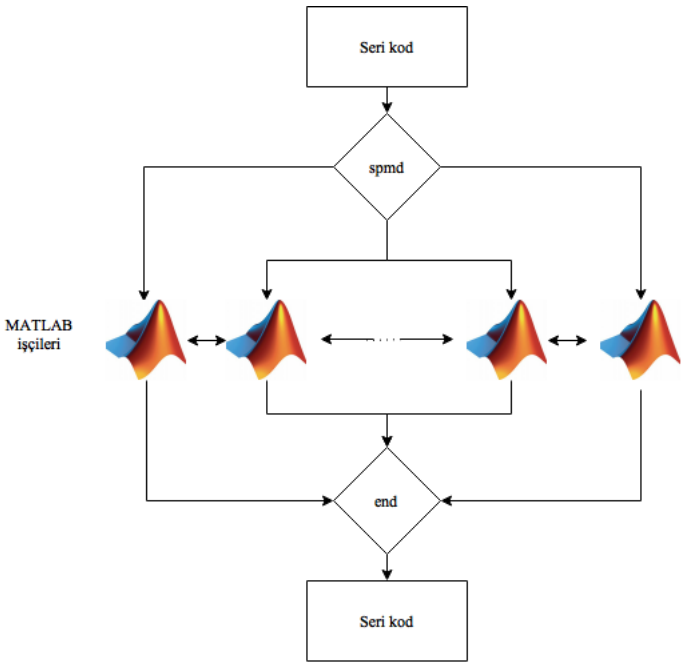
Şekil 3.4'te ise parfor kullanımına uygun olmayan kod verilmiştir. Fibonacci serisinin hesaplanmasında bir önceki iterasyonun sonucu bir sonrakinin hesaplanmasında kullanıldığı için Fibonacci serisi parfor kullanımı için uygun değildir.

```
f = zeros(1,8)
f(1) = 1;
f(2) = 2;
parfor n =3:8
     f(n) = f(n-1) + f(n-2);
end
```

Şekil 3.4: parfor kullanılamayan kod

3.1.2.2 spmd

spmd tek program çoklu veri (single program multiple data) demektir. spmd ifadesi bir kod bloğunun birden çok işçi üzerinde eş zamanlı çalışmasına olanak sağlar. Seri ve paralel programlama arasında geçiş yapmak oldukça kolaydır. spmd bloğu içerisindeki kod paralel çalışırken, blok dışarısındaki kodlar seri çalışır. Çalışma mantığı Şekil 3.5'te gösterilmiştir.



Şekil 3.5: spmd kod bloğu

spmd için gerekli tipik uygulamalar birden çok veri üzerinde eş zamanlı çalışmanın gerektiği, işçiler arası haberleşme ve senkronizasyonun gerektiği durumlardır. Kullanıldığı en yaygın durumlar:

- Programın çalışmasının çok zaman alması: spmd çözümleri eş zamanlı çalıştırarak süreyi kısaltır.
- Büyük veri setleri üzerinde çalışan algoritmalar: spmd veriyi birden çok işlemciye paylaştırarak hesaplamayı yapar.

spmd parametresiz çalıştırıldığında havuzdaki tüm işçilerde çalıştırılır. Sadece belirli sayıda işçi üzerinde çalıştırılmak istenirse parametre olarak işçi sayısı verilir (Mathworks 2016).

spmd bloğunda işçilerin birbirleriyle haberleşmeleri için belirli komutlar vardır. Bunlar:

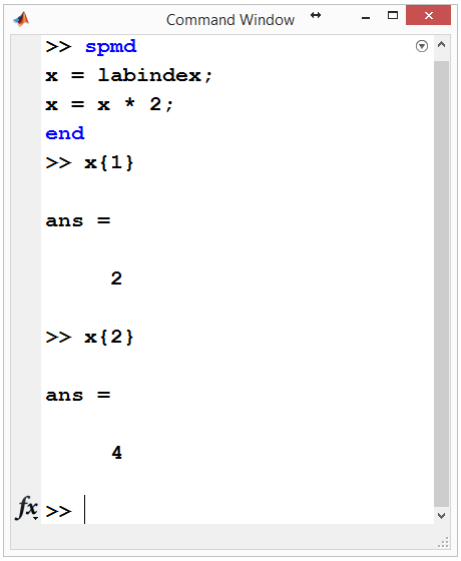
- *labBarrier*: Blok içerisinde tüm işçilerin bu komutun olduğu yere ulaştığından emin olmamızı sağlar. Önce ulaşan işçileri bekletir.
- *labBroadcast*: Bir işçiden diğer tüm işçilere veri göndermek için kullanılır. Gönderen işçi *labBroadcast(kaynakIsci,veri)* seklinde gönderirken diğer işçiler için de *veri = labBroadcast(kaynakIsci)* şeklinde kullanılır.
- *labSend*: Bir işçiden başka bir işçiye veri göndermek için kullanılır. *labSend(veri, hedefIsci)* şeklinde veri gönderme işi yapılır.
- *labReceive*: Başka bir işçi tarafından gönderilen veriyi almak için kullanılır. Kullanımı *labReceive*(*kaynakIsci*) şeklindedir. *labReceive* komutu kullanıldığında eğer belirtilen kaynak işçi tarafından herhangi bir veri gönderilmemişse o işçi için kilitlenme meydana gelir ve program hata verir.
- *labSendReceive*: Eş zamanlı veri göndermek ve almak için kullanılır. Kullanımı *alinanVeri = labSendReceive(hedefIsci, kaynakIsci, gonderilenVeri)* şeklindedir.
- *labProbe*: Diğer işçinin veri gönderip göndermediğini, alınmayı bekleyen veri olup olmadığını kontrol etmek için kullanılır. Gönderilen bir veri olmadığında *labReceive* komutu kilitlenmeye sebep olduğundan faydalı bir komuttur. Kullanımı *bekleyenVeriVarmi = labProbe(kaynakIsci)* şeklindedir.

spmd bloğunda her işçinin kendine ait değişkenleri vardır. Bu yapı kompozit nesnelerle sağlanmaktadır. Kompozit nesneler MATLAB'da işçilerin verilerine doğrudan ulaşmaya olanak sağlar. Bu değişkenler en yaygın olarak spmd ifadelerinde kullanılır. Kompozitler dizilere benzer. İki farklı şekilde kompozit nesne oluşturulabilir:

- İstemcide "*Composite"* fonksiyonu kullanılarak oluşturulur. Oluşturulan bu değişken işçilerde kullanılabilir.

- spmd ifadesi içinde tanımlanan tüm değişkenler kompozit yapıdadır. Blok içerisinde normal değişken gibi, spmd bloğundan sonra da istemci tarafından kompozit nesne olarak erişilebilirdir (Mathworks, 2016).

Şekil 3.6'da örnek bir kompozit nesne oluşturulmuştur. spmd bloğu içinde oluşturulan x değişkeni kompozit yapıdadır. Blok dışında 2 numaralı işçinin x değerine ulaşılmak istenildiğinden x{2} yazılarak ulaşılabilir.



Şekil 3.6: Kompozit nesne kullanımı

3.1.2.3 parfeval

parfeval komutu MATLAB'da arka planda çalışan asenkron işlemler için kullanılır. İçerisine parametre olarak bir fonksiyon, geriye kaç farklı değer döndüreceği ve fonksiyonun çalışması için gerekli olan parametreleri alır. parfeval çalışırken arka planda çalıştığı için MATLAB komut satırında başka işlemler yapmaya izin verir. Eğer başka işlem yapılmak istenmiyor, parfeval'in tamamlanmasının beklenilmesi isteniyorsa *fetchOutputs* komutu kullanılabilir. Bu komut tüm asenkron işlemler bitene kadar başka işlem yaptırmamaktadır. Eğer tüm işlemlerin değil de sadece bir tanesinin bitmesi beklenmek istenirse de *fetchNext* komutu kullanılabilir.

4. TAVLAMA BENZETİMİNİN PARALELLEŞTİRİLMESİ

Tavlama benzetiminin paralelleştirilmesi sıralı bir algoritma olduğundan tekrarlı yapısı sebebiyle oldukça zordur (Chen ve diğ. 1998). Onbaşoğlu ve Özdamar (2001) ve Ferreiro ve diğ. (2013) yaptıkları çalışmalarda sürekli problemler için tavlama benzetimi paralelleştirme yöntemlerini detaylıca incelemişlerdir. Kombinatoryal problemler için de tavlama benzetimi paralelleştirilmesi aşağıdaki yöntemler kullanılarak yapılabilir.

4.1 Maliyet Fonksiyonunun Paralelleştirilmesi

Bu yöntemde maliyet fonksiyonunun hesaplanması işlemi çekirdekler arasında paylaştırılabilir. Maliyet fonksiyonunun hesaplanmasının çok karmaşık olduğu durumlarda faydalıdır. Ancak her maliyet fonksiyonu hesaplaması parçalanabilir yapıda değildir. Maliyet fonksiyonunun parçalanamadığı durumlarda kullanılamaz. Ayrıca hesaplamanın çok basit olduğu durumlarda paralelleştirme işlemi, hesaplamadan daha fazla sürmektedir. Bu durumda kullanmak fayda yerine zarar sağlamaktadır.

4.2 Arama Uzayı Paylaştırma

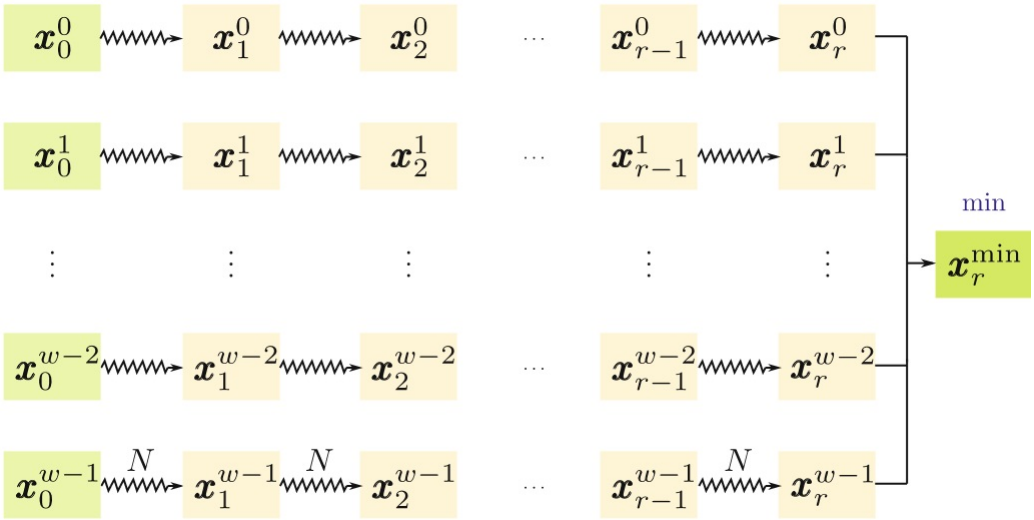
Arama uzayının alt uzaylara bölünüp her işlemcinin kendi alt uzayında en iyi değerini aradığı, tüm işlemcilerin arama işlemini bitirdiğinde bulunan sonuçlardan en küçüğünün seçildiği yöntemdir. Arama uzayının çok büyük olmadığı durumlarda faydalıdır. En küçük KAP boyutu 12'dir. Boyutu n=12 olan bir KAP için 12! = 479001600 farklı çözüm vardır. Tüm çözümlere 12 işlemcili bir bilgisayarda bu yöntem kullanılarak paralelleştirildiğinde bile her işçinin denemesi 39916800 farklı çözüm vardır.

4.3 Çoklu Markov Zinciri Yöntemi

Tavlama benzetimini paralelleştirmede kullanılabilecek en uygun yöntemdir. Çoklu Markov zincirleri birbiriyle haberleşmenin olmadığı asenkron şekilde çalıştırılır ve belirli periyotlarla veya süreç sonunda zincirler arası haberleşme yapılarak bulunan en iyi sonuç alınır. Haberleşme türüne göre senkron ve asenkron olmak üzere iki kategoriye ayrılır.

4.3.1 Asenkron Markov Zinciri

Tavlama benzetimini paralelleştirmek için en kolay yöntemdir. Günümüz bilgisayarlarındaki çok çekirdekli işlemcilerden faydalanılarak her çekirdekte eş zamanlı tavlama benzetimi çalıştırılır. Her çekirdek işlemini tamamladıktan sonra, bulunan sonuçlardan en iyi seçilir. Asenkron Markov zinciri yönteminde her çekirdek aynı başlangıç çözümüyle veya rastgele üretilen farklı başlangıç çözümleriyle başlatılabilir. Şekil 4.1'de asenkron Markov zincirinin çalışma mantığı verilmiştir.



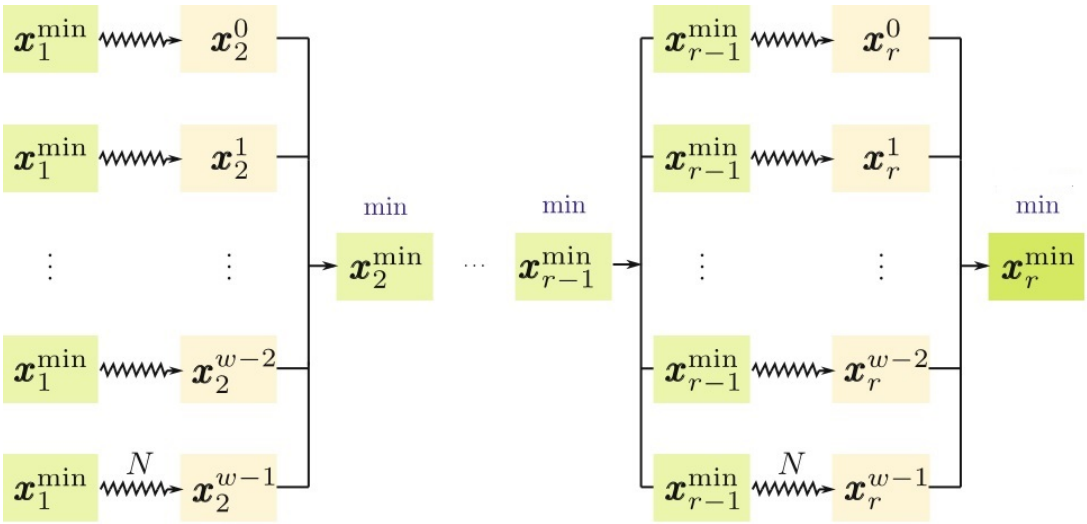
Şekil 4.1: Asenkron Markov zinciri yöntemi

4.3.2 Senkron Markov Zinciri

Senkron Markov zinciri yönteminde her bir işlemci çekirdeğinde farklı bir başlangıç çözümüyle tavlama benzetimi algoritması çalıştırılır. Her çekirdek bir sonraki sıcaklık değerine ulaşıncaya kadar çekirdekler birbirinden bağımsız olarak işlemlerini yapar. Hedef noktaya ulaşınca haberleşme sağlanarak, sonuçlardan en iyi olanı bulunur ve her çekirdek bulunan en iyi çözümü yeni başlangıç çözümü olarak kullanıp işlemi tekrarlar.

Senkron Markov zinciri yöntemi genetik algoritma ile tavlama benzetimi algoritmasının birleşimi olarak değerlendirilebilir. Tavlama benzetimindeki her bir zincir genetik algoritmadaki bireylere karşılık gelir (Ferreiro ve diğ. 2013). En iyi sonucun alınıp yeni başlangıç çözümü olarak kabul edilmesi de gen havuzundaki en iyi bireyin neslini devam ettirmesi olarak yorumlanabilir.

Şekil 4.2'de her sıcaklık değişiminden sonra bulunan en iyi sonucun alınıp yeni başlangıç çözümü kabul edilmesi ve en iyi çözümün aranması işlemi gösterilmiştir. Ancak sıcaklık değişmelerindeki haberleşmelerin de bir maliyeti vardır. Bu yüzden her sıcaklık değişimi yerine belirli sayıda sıcaklık değişiminden sonra haberleşmenin yapılması haberleşme maliyeti azaltılması yönünden verimli olabilir. Ancak kaç sıcaklık değişiminde bir haberleşme yapılırsa daha iyi sonuç elde edileceği de ayrı bir parametre olarak karşımıza çıkmaktadır.



Şekil 4.2: Senkron Markov zinciri yöntemi

5. UYGULAMA SONUÇLARI

Çalışmada tavlama benzetimi yöntemi bir önceki bölümde belirtilen Paralelleştirme yöntemleriyle paralelleştirilmiş, yöntemlerin hata oranı 0.01'den az olması için gereken süre ve iterasyon sayılarındaki değişimler incelenmiştir. Hata oranı (5.1) eşitliği kullanılarak belirlenmiştir. Denklemde *C* bulunan çözüm değeri ve C_{best} problem için bilinen en iyi değerdir.

$$Q = \frac{C - C_{best}}{C_{best}} \tag{5.1}$$

5.1 Test Ortamı

Bu çalışmadaki deneyler iki adet altışar fiziksel çekirdekli Intel Xeon E5-2620 2.0 GHz işlemcisi olan 32 GB ram bulunan bir iş istasyonu üzerinde çalıştırılmıştır. Dolayısıyla sistemde toplam 12 fiziksel çekirdek bulunmaktadır. MATLAB sistemimizde maksimum 12 işçiye kadar izin verdiğinden paralel denemelerde 12 işçi kullanılmıştır.

5.2 Parametrelerin Belirlenmesi

Soğuma yöntemi olarak üstel soğuma, soğuma katsayısı olarak α=0,9, aynı sıcaklıktaki deneme sayısı olarak problem boyutunun yarısı (n/2) ve maksimum iterasyon sayısı olarak problemin 3000 katı belirlendiğinde iyi sonuçlar alınmıştır. Sabit bir başlangıç sıcaklığı belirlendiğinde bazı problemler iyi sonuç verirken, bazılarında maksimum iterasyona ulaştığı halde sonuç alınamadığı için her problem için kendisine uygun bir başlangıç sıcaklığı belirlenmiştir. Sıcaklık 1'in altına düştüğünde de yeniden ısıtma işlemi uygulanmıştır.

5.3 Kullanılan Problemler

Çalışmada kullanılan problemler KAP örneklerinin bulunduğu ve araştırmacıların geliştirdikleri yöntemleri bu örneklerle test ettiği bir kütüphane olan, Burkard ve diğ. (1997) tarafından oluşturulan QAPLIB sayfasından alınmıştır. Sayfada problemlerle ilgili detaylı bilgiler bulunmaktadır.

Bu çalışmada 3 farklı problem grubundan 36 tane örnek kullanılmıştır. Kullanılan problemler ve temel özellikleri aşağıdaki gibidir:

- Nugent: Nugent ve diğ. (1968) tarafından önerilen, boyutları 12 ile 30 arasında değişen 15 örnek bulunmaktadır. Uzaklık matrisi Manhattan uzaklıklarının dikdörtgensel ızgara üzerine yerleştirilmesi ile oluşturulmuştur. Örnek boyutu n = {14,16,17,18,21,22,24,25} olan örnekler daha büyük örneklerin belirli satır ve sütunlarının silinmesi ile elde edilmiştir.
- Skorin: Skorin ve Kapov (1990) tarafından önerilen, boyutları 42 ile 100 arasında değişen 13 örnek bulunmaktadır. Mesafeler diktörtgenseldir ve uzaklıklar sözde rastgele sayılardır.
- Lipa: Li ve Pardalos (1992) tarafından önerilen, boyutları 20 ile 90 arasında değişen 16 örnek bulunmaktadır. Bilinen optimum değerlerden asimetrik örnekler oluşturulmuştur. Bu çalışmada örneklerden 8 tanesi kullanılmıştır.

5.4 Karşılaştırma İçin Kullanılan Yöntemler

Çalışmada kullanılan yöntemler seri bir şekilde çalışan tavlama benzetimi, seri tavlama benzetimi algoritmasında maliyet fonksiyonunun paralel hesaplandığı maliyet fonksiyonu paralelleştirmesi, asenkron hesaplamalı tavlama benzetimi ve belirli aralıklarla haberleşmenin yapıldığı senkron hesaplamalı tavlama benzetimi yöntemleridir.

5.4.1 Maliyet Fonksiyonunun Paralelleştirilmesi

KAP'ta maliyet fonksiyonu MATLAB ortamında paralel for döngüsünün paralel hali olan parfor komutuyla paralelleştirilmiş ve etkileri incelenmiştir. Şekil 5.1'de seri ve Şekil 5.2'de paralel maliyet hesabı kodları verilmiştir.

```
for i = 1 : size(distance,1)
    for j = 1 : size(distance,1)
      cost = cost + distance(i, j) * flow(assign(i), assign(j));
    end
end
```

Şekil 5.1: Seri maliyet hesabı kodu

Paralel maliyet hesabındaki dış döngüde problemin boyutu kadar döngü adımı işçilere MATLAB tarafından otomatik olarak paylaştırılmaktadır.

```
parfor i = 1 : size(distance,1)
    for j = 1 : size(distance,1)
      cost = cost + distance(i, j) * flow(assign(i), assign(j));
    end
end
```

Şekil 5.2: Paralel maliyet hesabı kodu

5.4.2 Asenkron Hesaplamalı Paralel Tavlama Benzetimi

Asenkron hesaplamalı paralel tavlama benzetimi yönteminde her işlemcide ayrı bir tavlama benzetimi algoritması çalıştırılmıştır. Paralelleştirme spmd paralelleştirmesiyle yapıldığında işçilerden bir tanesi istenilen sonucu bulursa kendi arama işlemini sonlandırmakta, diğer işçilerin arama işlemlerini bitirmesini beklemektedir. Sonuç bulunduğunda diğerlerinin arama işlemi sonlandırılamamaktadır. Ancak parfeval kullanıldığında işçilerden bir tanesi istenilen sonuca ulaşırsa diğer işçilerdeki işlemler iptal edilebilmektedir. Bundan dolayı bu yöntemde parfeval komutu yardımıyla paralelleştirme yapılmıştır.

5.4.3 Senkron Hesaplamalı Tavlama Benzetimi Yöntemi

Senkron hesaplamalı tavlama benzetimi yönteminde spmd paralelleştirmesi kullanılarak işçiler buldukları çözümleri 1. işçiye göndermekte, 1. işçi çözümlerden

en iyisini bulup diğer işçilere bu en iyi çözümü göndermektedir. Her işçi bu çözümü ara başlangıç çözümü olarak alıp bu çözümü kullanarak tavlama benzetimi işlemlerini yapmaya devam etmektedir. Yöntemde haberleşmenin maliyeti olduğundan çok sık haberleşme yapmak yerine 100 iterasyonda bir haberleşme yapıldığındaki etkileri incelenmiştir.

Bu yöntemde MATLAB ortamında işçiler arası haberleşmeler spmd bloğunda yapılabilmektedir. İşçilerden birisinin istenilen çözümü bulduğunda kendisini sonlandırması 1. işçinin hesaplamasını bitiren işçiden veri alması gerektiğinden kilitlenmeye sebep olmaktadır. Bu yüzden sonlandırma işlemi haberleşme sırasında işçilerden birinde istenilen çözüme ulaşıp ulaşmadığına bakılarak yapılmaktadır. Haberleşme kısmına gelmeden çözüm bulunsa da çalışmaya devam edilmektedir.

Senkron hesaplamalı tavlama benzetimi yönteminde hep en iyi çözüm üzerinden gitmenin iyi olmayabileceği düşünülerek, genetik algoritma benzeri evrimsel senkron tavlama benzetimi yöntemi önerilmiştir.

5.4.3.1 Evrimsel Senkron Hesaplamalı Tavlama Benzetimi Yöntemi

Evrimsel senkron hesaplamalı tavlama benzetimi yönteminde senkron yöntemden farklı olarak haberleşmede işçilerin bulduğu çözümler sıralanmakta; en iyi 6 çözüm, kötülerden rastgele 2 tanesi ve rastgele üretilen 4 başlangıç çözümü ara çözüm olarak kullanılıp algoritma çalıştırılmaya devam etmektedir. Yöntemde başlangıç çözümü olarak kullanılacak çözümlerin seçimi evrim teorisini andırdığından evrimsel olarak isimlendirilmiştir. Birleştirme işlemi yapan işçi için algoritma aşağıdaki gibi özetlenebilir:

Adım 1: Başlangıç sıcaklığı T , soğuma katsayısı α belirle.

Adım 2: Rastgele bir başlangıç çözümü x oluştur.

Adım 3: x çözümünde iki tesisin yerini değiştirerek yeni çözüm x' oluştur.

Adım 4: Eğer yeni çözüm daha iyiyse yeni çözümü kabul et: x=x'. Eğer yeni çözüm daha kötüyse 0 ile 1 arasında rastgele bir sayı belirle. Belirlenen sayı (2.3) eşitliğinden küçükse yeni çözümü kabul et: x=x'.

Adım 5: Aynı sıcaklıktaki adım sayısı tamamlanmadıysa Adım 3'e git.

Adım 6: Sıcaklığı α'ya göre güncelle: T=T* α.

Adım 7: Eğer sıcaklık 1'in altına düştüyse sıcaklığı ilk sıcaklığa yükselt.

Adım 8: Eğer iterasyon sayısı birleştirme sıklığındaysa diğer işçilerin buldukları çözümleri al.

Adım 8.1: Çözümleri sırala, kötü çözümlerden 2 tanesini seç ve sıralamada 7 ve 8'e yerleştir.

Adım 8.2: Rastgele 4 çözüm oluştur ve sıralamada 8-12 arasına yerleştir.

Adım 8.3: Sıralı çözümleri her işçiye bir çözüm gidecek şekilde gönder.

Adım 8.4: İstenilen çözümün bulunup bulunmadığı bilgisini gönder.

Adım 9: Eğer istenen çözüm bulunmuşsa veya sonlandırma kriterine ulaşılmışsa algoritmayı sonlandır. Sonlandırma kriteri sağlanmıyorsa Adım 3'e git.

5.5 Karşılaştırmalar

Çalışmada öncelikle maliyet fonksiyonunun paralleştirilmesinin hesaplama süresine etkileri incelenmiştir. Bunun için Lipa problemlerinde maliyet fonksiyonu seri ve paralel olarak çalıştırılmış, Tablo 5.1'de 100 defa maliyet hesabı hesabı için

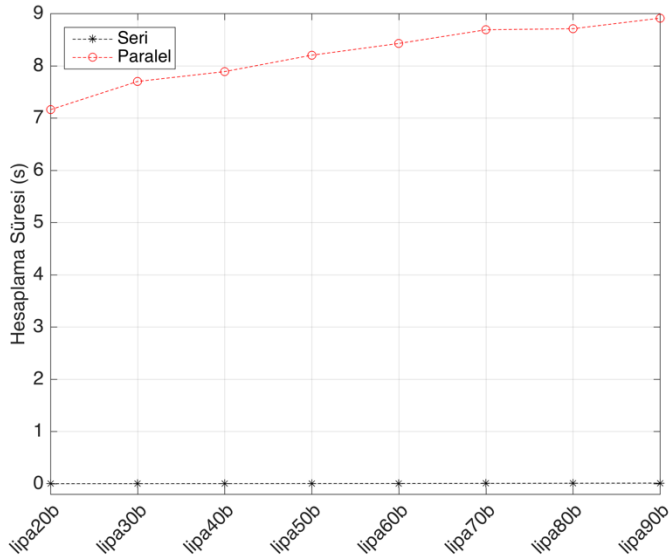
gerekli süre verilmiştir. Bu test yapılırken 1. konuma 1. tesis, 2. konuma 2. tesis, …, n. konuma n. tesis gelecek şekilde sıralı bir atama kullanılmıştır.

Tablo 5.1'de görüldüğü gibi Lipa problemleri için maliyet fonksiyonunun paralelleştirilmesi maliyeti, hesaplama maliyetine göre oldukça fazladır. Diğer örnekler için de maliyet fonksiyonu paralelleştirildiğinde benzer sonuçlar alınmıştır.

Tablo 5.1: Lipa problemleri 100 defa seri ve paralel maliyet hesabı

Problem Adı	**Seri Süre (s)**	**Paralel Süre (s)**
lipa20b	0,0011	7,1601
lipa30b	0,0019	7,6996
lipa40b	0,0028	7,8882
lipa50b	0,0041	8,2019
lipa60b	0,0057	8,4266
lipa70b	0,0074	8,6888
lipa80b	0,0096	8,7087
lipa90b	0,0123	8,9102
Ortalama	**0,0056**	**8,2105**

Ayrıca Şekil 5.3 incelendiğinde problem boyutu arttıkça seri maliyet fonksiyonunun hesaplama süresi artmasına rağmen, paralel maliyet hesabına göre artış oldukça düşüktür ve grafikte belli olmayacak kadar küçüktür.



Şekil 5.3: Lipa problemleri 100 defa seri ve paralel maliyet hesabı süreleri

Haberleşmenin 100 iterasyonda yapıldığı ve en iyi sonucun ara başlangıç çözümü kabul edildiği senkron hesaplamalı yöntemde Lipa problemleri için seri

yöntemle karşılaştırılması sonuçları Tablo 5.2'de verilmiştir. Ara başlangıç çözümlerinde hep bulunan iyi çözümden gidildiğinde yerel minimumlara takılmaların arttığı; hedeflenen hata değerine ulaşılamadığı, maksimum iterasyon sayısına ulaşılıp kodun sonlandığı gözlemlenmiştir. Diğer problemler için de benzer sonuç alınmıştır.

Tablo 5.2: Lipa problemleri seri ve senkron yöntem karşılaştırma

	Seri		**Senkron Hesaplamalı**	
Problem	Süre (s)	Hata (Q)	Süre (s)	Hata (Q)
lipa20b	0,28	0,0000	18,69	0,0645
lipa30b	1,28	0,0000	66,75	0,1000
lipa40b	4,63	0,0000	161,33	0,1261
lipa50b	12,32	0,0000	363,49	0,1489
lipa60b	34,69	0,0000	664,37	0,1634
lipa70b	82,91	0,0000	1112,16	0,1687
lipa80b	265,67	0,0009	1774,62	0,1751
lipa90b	443,12	0,0008	2679,61	0,1768
Ortalama	**105,61**	**0,0002**	**855,13**	**0,1404**

Maliyet hesabı paralelleştirmesi ve senkron hesaplamalı paralelleştirme yönteminden iyi sonuçlar alınamayınca bundan sonraki kısımda seri, asenkron hesaplamalı ve evrimsel hesaplamalı senkron yöntemi arasında yapılmıştır.

Nugent problemleri için seri, asenkron hesaplamalı ve haberleşmenin 100 iterasyonda bir yapıldığı evrimsel senkron hesaplamalı yöntemleri için karşılaştırma değerleri Tablo 5.3'te verilmiştir. Ortalama sonuçlara bakıldığında asenkron hesaplamalı yöntem seri yönteme göre süre ve iterasyon olarak daha iyi sonuçlar vermiştir. Asenkron hesaplamalı yöntem süre olarak ortalama %46,24 ve iterasyon olarak %72,88 iyileşme sağlamıştır. Hesaplama süresi olarak evrimsel senkron hesaplamalı yöntemi ise problem boyutu 17'den küçükler için daha kötü sonuçlar vermiştir. Ancak iterasyon sayısındaki iyileşmelere bakıldığında da en iyi sonuçlar evrimsel senkron hesaplamalı yöntemde alınmıştır. Evrimsel senkron hesaplamalı yöntemde ortalama olarak sürede %14,68 ve iterasyon sayısında %84,94 iyileşme sağlamıştır.

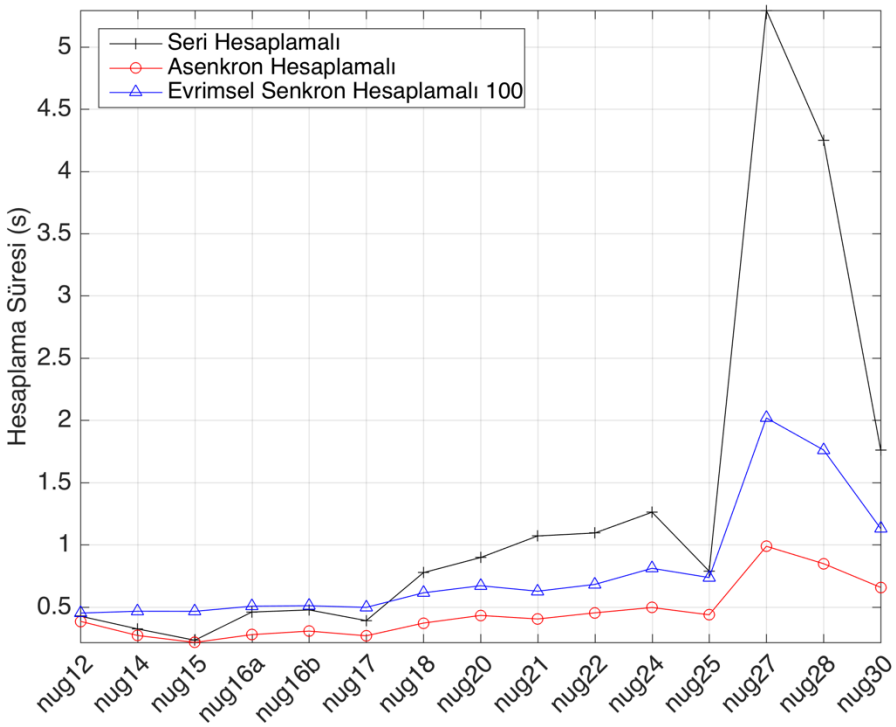
Nugent problemlerinde Şekil 5.4'te verilen hesaplama süreleri grafiği incelendiğinde asenkron hesaplamalı yöntemin en iyi sonuçları verdiği

görülmektedir. Özellikle grafikte nug27, nug28 ve nug30 için seri yöntemde hesaplama süreleri diğer problemlere göre oldukça fazladır. Ancak asenkron hesaplamalı yöntemde de hesaplama süresi artmasına rağmen bu artış seri yöntemle kıyaslandığında oldukça azdır. Evrimsel senkron hesaplamalı yöntemde de şekilde görüldüğü gibi nug18 problemi ve daha büyük problemler için daha iyi sonuçlar vermiştir.

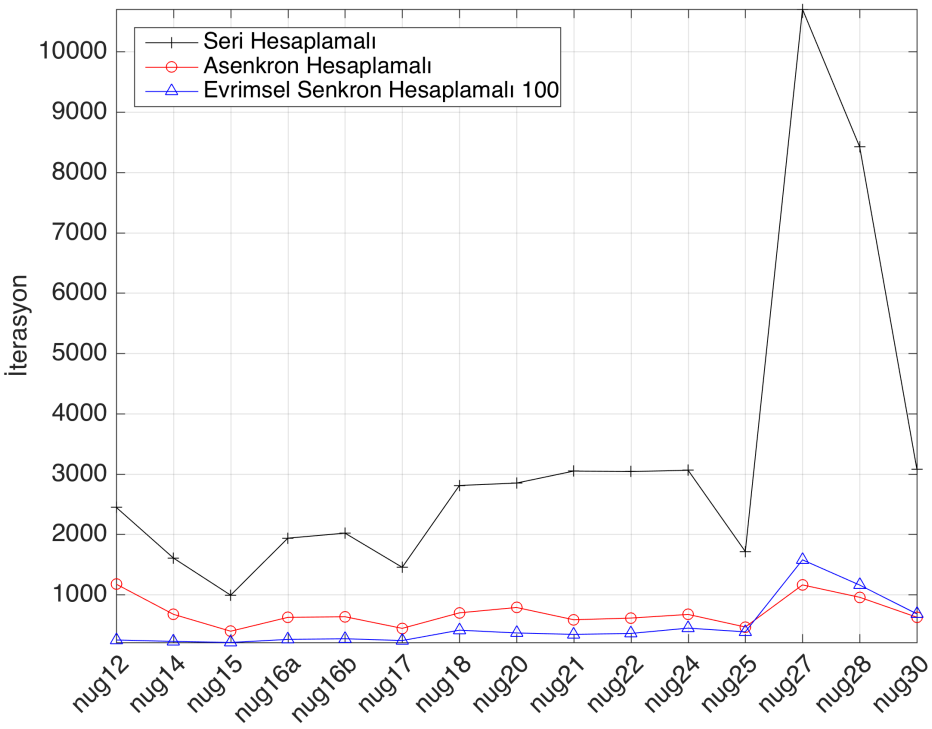
Tablo 5.3: Nugent problemleri hesaplama süreleri ve iterasyon sayıları

	Seri		**Asenkron Hesaplamalı**				**Evrimsel Senkron Hesaplamalı**			
Problem	Süre (s)	İterasyon	Süre (s)	Süre İyileşme (%)	İterayon	İterasyon İyileşme (%)	Süre (s)	Süre İyileşme (%)	İterayon	İterasyon İyileşme (%)
nug12	0,43	2446	0,39	9,76	1174	52,02	0,45	-6,07	248	89,86
nug14	0,32	1606	0,27	16,33	673	58,07	0,46	-41,54	225	85,99
nug15	0,23	991	0,22	7,48	394	60,28	0,45	-92,96	207	79,11
nug16a	0,46	1936	0,28	39,44	623	67,81	0,48	-5,01	258	86,68
nug16b	0,48	2019	0,31	35,63	634	68,61	0,50	-5,30	269	86,68
nug17	0,39	1453	0,27	31,18	441	69,65	0,49	-24,55	239	83,55
nug18	0,78	2809	0,37	52,18	698	75,14	0,59	23,78	411	85,37
nug20	0,90	2850	0,43	51,78	787	72,38	0,59	34,58	365	87,19
nug21	1,07	3049	0,40	62,25	584	80,85	0,58	45,60	341	88,81
nug22	1,10	3040	0,45	58,56	612	79,88	0,62	43,56	357	88,26
nug24	1,26	3062	0,50	60,66	672	78,05	0,68	45,84	445	85,47
nug25	0,79	1715	0,44	44,43	464	72,96	0,63	20,48	383	77,66
nug27	5,29	10703	0,99	81,33	1161	89,15	1,57	70,36	1577	85,27
nug28	4,25	8423	0,85	80,02	956	88,65	1,32	69,04	1159	86,24
nug30	1,76	3081	0,66	62,64	625	79,71	1,02	42,31	681	77,90
Ort.	**1,30**	**3279**	**0,45**	**46,24**	**700**	**72,88**	**0,70**	**14,68**	**478**	**84,94**

Nugent problemlerinde Şekil 5.5'te verilen iterasyon sayıları grafiği incelendiğinde asenkron hesaplamalı ve evrimsel senkron hesaplamalı yöntemlerin iterasyon sayılarını büyük oranda düşürdükleri gözlenmektedir. nug25 ve daha küçük problemler için en iyi iterasyon sayıları evrimsel senkron hesaplamalı yöntemde alınmıştır. nug27, nug28 ve nug30 problemlerinde ise en iyi sonuçlar asenkron hesaplamalı yöntemde alınmıştır.



Şekil 5.4: Nugent problemleri hesaplama süreleri



Şekil 5.5: Nugent problemleri iterasyon sayıları

Evrimsel senkron hesaplamalı yöntemin Şekil 5.4'te görüldüğü gibi nug17 ve daha küçük problemlerde süre olarak kötü sonuç vermesine rağmen iterasyon

sayıları olarak daha iyi olmasını evrimsel senkron hesaplamalı yöntemdeki haberleşme ve sonuçların sıralanması maliyetlerine bağlanabilir.

Nugent problemlerine genel olarak bakıldığında, asenkron hesaplamalı ve evrimsel senkron hesaplamalı yöntemlerin seri yöntemlere göre daha iyi sonuç verdiği, evrimsel senkron hesaplamalı yöntemin boyutu 17'den küçük problemlerde haberleşme maliyetinden dolayı süreyi artırdığı görülmüştür. Amacın istenilen kalitede sonucu en hızlı almak olduğundan Nugent problemleri için en iyi sonuçlar asenkron hesaplamalı yöntemde elde edilmiştir.

Lipa problemleri için yöntemlerin karşılaştırma değerleri Tablo 5.4'te verilmiştir. Tabloya göre ortalama sonuçlara bakıldığında Nugent problemlerinde olduğu gibi asenkron hesaplamalı yöntem seri yönteme göre süre ve iterasyon olarak daha iyi sonuçlar vermiştir. Asenkron yöntem süre olarak ortalama %70,00 ve iterasyon olarak %80,90 iyileşme sağlamıştır. Evrimsel senkron hesaplamalı yöntemi ise hesaplama süresi olarak seri yönteme göre sadece lipa20b probleminde kötü sonuç vermiştir. Evrimsel senkron hesaplamalı yöntemi ortalama olarak sürede %50,08 ve iterasyon sayısında %80,87 iyileşme sağlamıştır.

Lipa problemlerinde Şekil 5.6'da verilen hesaplama süreleri grafiği incelendiğinde asenkron hesaplamalı yöntemin en iyi sonuçları verdiği görülmektedir. Problemin boyutu arttıkça seri yöntemde hesaplama süreleri diğer problemlere göre oldukça fazladır. Ancak paralel yöntemlerde bu artış oranı daha azdır.

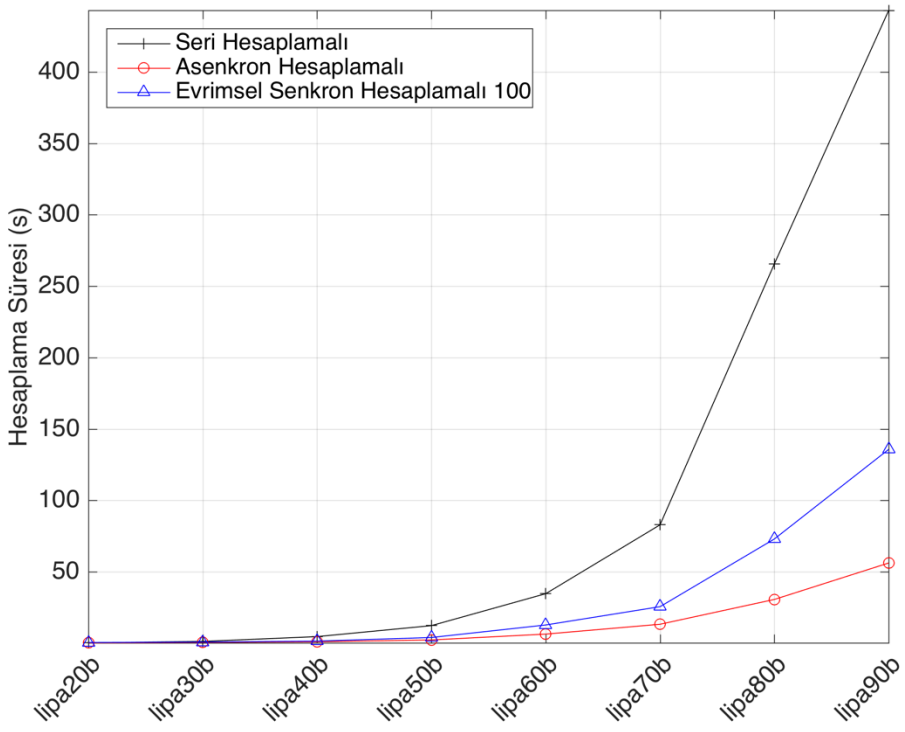
Tablo 5.4: Lipa problemleri hesaplama süreleri ve iterasyon sayıları

	Seri		**Asenkron Hesaplamalı**				**Evrimsel Senkron Hesaplamalı**			
Problem	Süre (s)	İterasyon	Süre (s)	Süre İyileşme (%)	İterayon	İterasyon İyileşme (%)	Süre (s)	Süre İyileşme (%)	İterayon	İterasyon İyileşme (%)
lipa20b	0,28	876	0,27	4,18	431	50,83	0,49	-76,60	227	74,10
lipa30b	1,28	2177	0,57	55,65	507	76,69	0,74	42,23	401	81,58
lipa40b	4,63	4670	1,04	77,57	676	85,52	1,45	68,79	745	84,05
lipa50b	12,32	7881	2,30	81,32	1083	86,26	3,43	72,15	1327	83,16
lipa60b	34,69	14385	6,40	81,54	2310	83,94	8,95	74,19	2460	82,90
lipa70b	82,91	23154	13,28	83,98	3326	85,63	22,41	72,97	4327	81,31
lipa80b	265,67	51750	30,65	88,46	5378	89,61	68,67	74,15	10060	80,56
lipa90b	443,12	62888	56,19	87,32	7101	88,71	120,69	72,76	13009	79,31
Ort.	**105,61**	**20973**	**13,84**	**70,00**	**2602**	**80,90**	**28,35**	**50,08**	**4070**	**80,87**

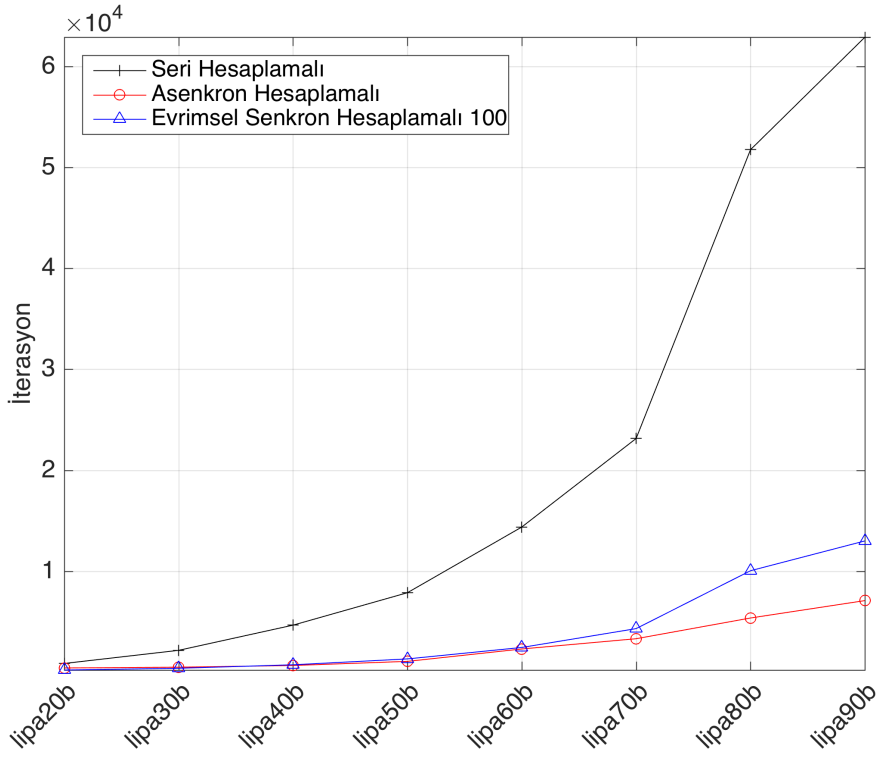
Lipa problemlerinde Şekil 5.7'de verilen iterasyon sayıları grafiği incelendiğinde paralel yöntemler iterasyon sayılarını büyük oranda düşürmüşlerdir. Paralel yöntemler lipa60b ve daha küçük problemlerde birbirine oldukça yakın sonuçlar vermelerine rağmen, problem boyutu arttıkça asenkron hesaplamalı yöntem daha iyi sonuçlar vermiştir.

Lipa problemlerinde paralel yöntemlerin iterasyon sayıları birbirine aşırı yakın olmasına rağmen hesaplama sürelerinde asenkron hesaplamalı yöntemin daha iyi sonuçlar vermesi sebebi olarak yine haberleşme maliyeti verilebilir.

Lipa problemlerine genel olarak bakıldığında paralel yöntemler daha iyi sonuç vermiştir. Sadece evrimsel senkron hesaplamalı yöntem lipa20b problemi için süre olarak daha kötü sonuç vermiş, ancak iterasyon olarak daha az iterasyonda sonuca ulaşmıştır. Süre ve iterasyon sonuçları birlikte değerlendirildiğinde Lipa problemleri için en iyi sonuçlar asenkron hesaplamalı tavlama benzetimi yönteminde alınmıştır.



Şekil 5.6: Lipa problemleri hesaplama süreleri



Şekil 5.7: Lipa problemleri iterasyon sayıları

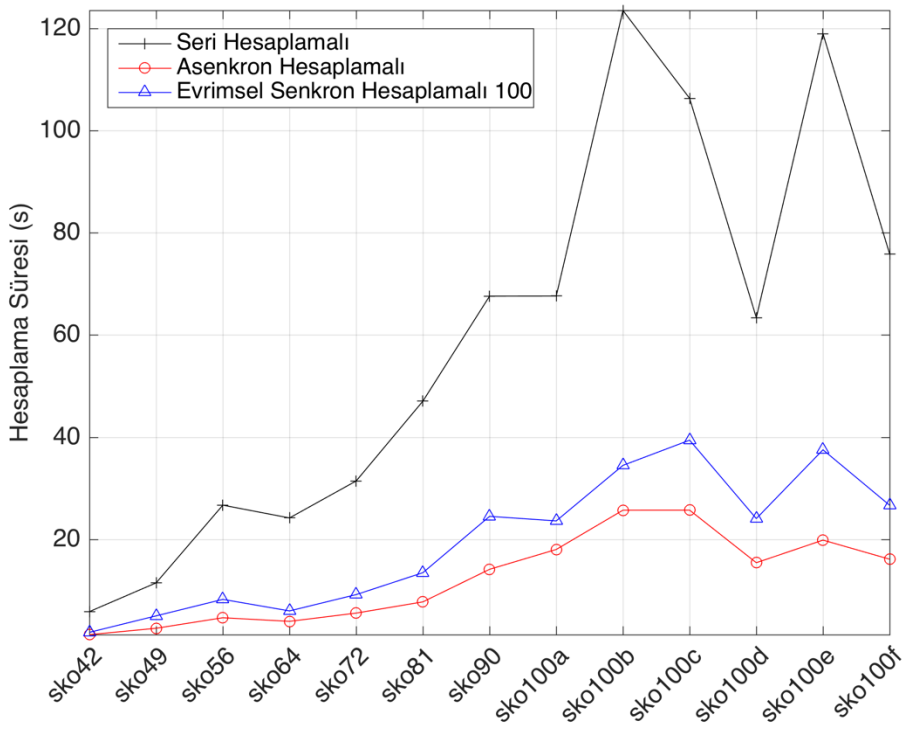
Skorin problemleri için yöntemlerin karşılaştırma değerleri Tablo 5.5'te verilmiştir. Tabloya göre ortalama sonuçlara bakıldığında asenkron hesaplamalı yöntem seri yönteme göre tüm problemlerde süre ve iterasyon olarak daha iyi sonuçlar vermiştir. Asenkron hesaplamalı yöntem süre olarak ortalama %79,39 ve iterasyon olarak %82,13 iyileşme sağlamıştır. Evrimsel senkron hesaplamalı yöntem de ortalama olarak sürede %71,66 ve iterasyon sayısında %80,51 iyileşme sağlamıştır.

Skorin problemlerinde Şekil 5.8'de verilen hesaplama süreleri grafiği incelendiğinde asenkron hesaplamalı yöntemin tüm problemler için en iyi sonuçları verdiği görülmektedir.

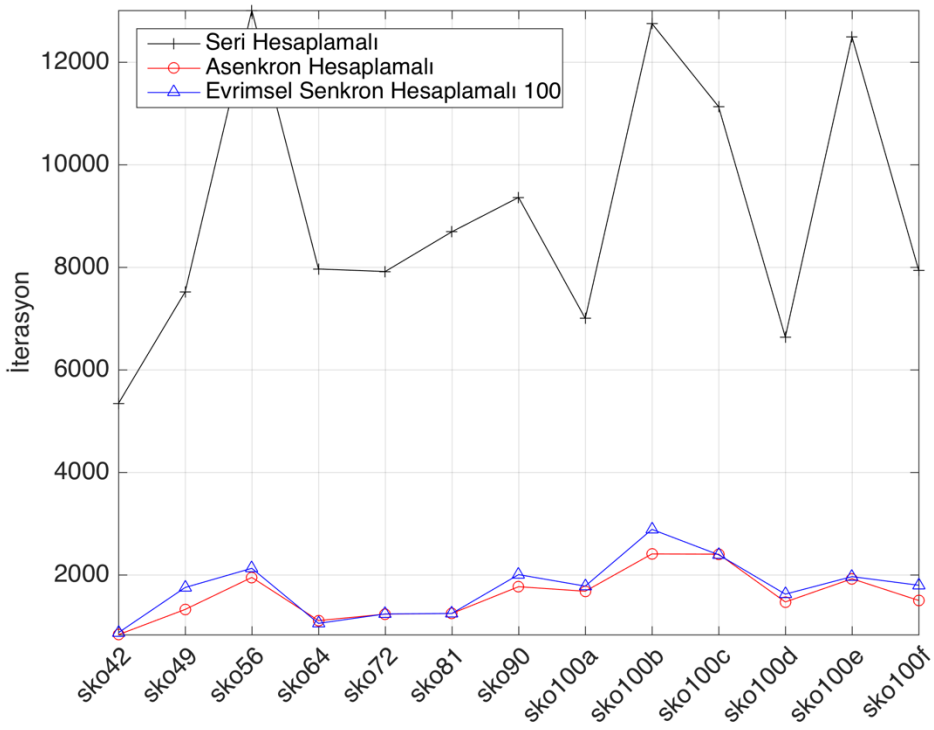
Skorin problemlerinde Şekil 5.9'da verilen iterasyon sayıları grafiği incelendiğinde paralel yöntemler iterasyon sayılarını büyük oranda düşürmüşlerdir. Genel olarak paralel yöntemlerin iterasyon sayıları birbirine yakın olmakla birlikte en iyi sonuçlar asenkron hesaplamalı yöntemde alınmıştır.

Tablo 5.5: Skorin problemleri hesaplama süreleri ve iterasyon sayıları

	Seri		Asenkron Hesaplamalı				Evrimsel Senkron Hesaplamalı			
Problem	Süre (s)	İterasyon	Süre (s)	Süre İyileşme (%)	İterayon	İterasyon İyileşme (%)	Süre (s)	Süre İyileşme (%)	İterayon	İterasyon İyileşme (%)
sko42	5,85	5344	1,33	77,21	836	84,35	1,90	67,59	875	83,63
sko49	11,48	7520	2,58	77,49	1328	82,34	4,23	63,12	1756	76,65
sko56	26,69	13008	4,65	82,59	1952	84,99	6,55	75,46	2130	83,63
sko64	24,19	7966	3,94	83,72	1111	86,05	4,71	80,51	1056	86,74
sko72	31,43	7917	5,56	82,32	1237	84,38	7,13	77,32	1240	84,34
sko81	47,05	8692	7,75	83,53	1250	85,62	9,44	79,93	1250	85,62
sko90	67,62	9361	14,13	79,10	1771	81,08	19,37	71,35	2007	78,56
sko100a	67,63	7002	18,02	73,36	1683	75,97	22,65	66,52	1783	74,54
sko100b	123,56	12752	25,70	79,20	2410	81,10	39,29	68,20	2889	77,34
sko100c	106,34	11130	25,71	75,82	2406	78,38	32,91	69,05	2394	78,49
sko100d	63,41	6641	15,42	75,68	1473	77,81	20,77	67,24	1626	75,51
sko100e	119,00	12492	19,84	83,32	1926	84,58	26,66	77,59	1968	84,25
sko100f	75,84	7942	16,10	78,77	1503	81,07	24,50	67,70	1800	77,34
Ort.	**59,24**	**9059**	**12,36**	**79,39**	**1607**	**82,13**	**16,93**	**71,66**	**1752**	**80,51**



Şekil 5.8: Skorin problemleri hesaplama süreleri



Şekil 5.9: Skorin problemleri iterasyon sayıları

5.5.1 Evrimsel Senkron Hesaplamalı Yöntemde Haberleşme Sıklığının Hesaplama Süresine Etkisi

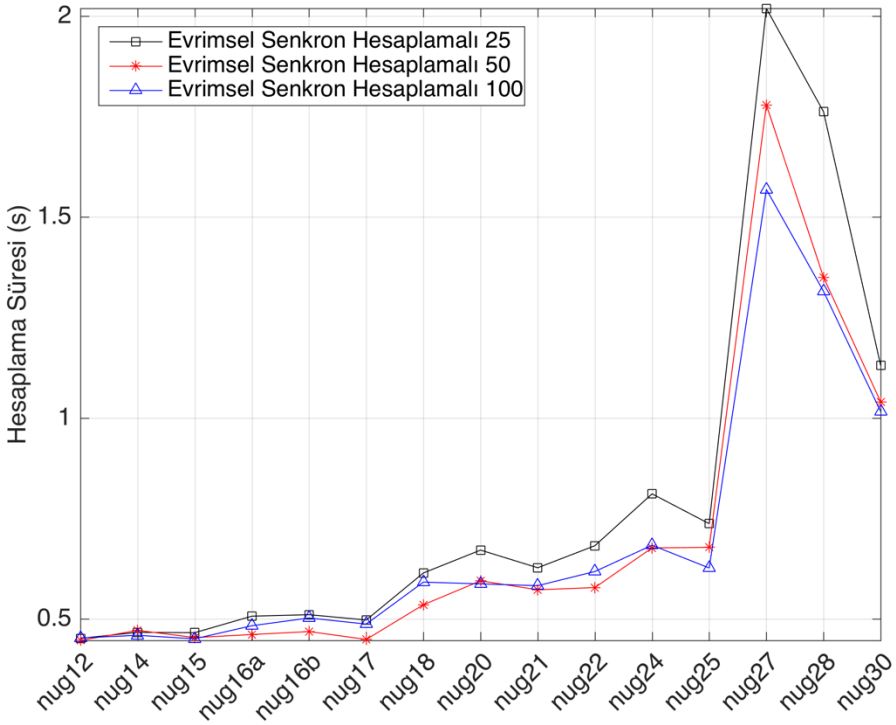
Çalışmanın son kısmında da evrimsel senkron hesaplamalı yöntemde haberleşme sıklığı artırıldığında hesaplama süresinin nasıl etkilendiği incelenmiştir. Önceki bölümdeki karşılaştırmalarda evrimsel senkron hesaplamalı yöntemde 100 iterasyonda bir haberleşme yapılmıştır. Bu kısımda ise karşılaştırmalarda 50 ve 25 iterasyonda bir haberleşme yapıldığındaki sonuçlar incelenmiş, 100 iterasyonda bir haberleşmeye göre karşılaştırması verilmiştir. Birleşme sıklığı 100'den fazla olduğunda bazı problemlerde yaklaşık 200 - 400 iterasyonda sonuca ulaşıldığı için, haberleşme çok seyrek yapılırsa asenkron yöntemden farkı olmayacağından 100'den daha seyrek haberleşme denenmemiştir.

Tablo 5.6'da Nugent problemleri için evrimsel senkron hesaplamalı yöntemde 50 ve 25 iterasyonda bir haberleşme yapıldığında 100 iterasyona göre iyileşme değerleri verilmiştir. Ortalama olarak hesaplama sürelerinde 50 iterasyonda bir haberleşme yapıldığında 100 iterasyondaki haberleşmeye göre %0,53 iyileşme ve

25 iterasyonda bir haberleşme yapıldığında %10,66 kötüleşme gözlemlenmiştir. Şekil 5.10'da verilen evrimsel senkron hesaplamalı yöntemlerin hesaplama süreleri grafiğine bakıldığında, 50 iterasyonda bir haberleşme genel olarak daha iyi sonuç vermiş; ancak nug25, nug27, nug28 ve nug30 problemlerinde en iyi sonuçlar haberleşme 100 iterasyonda bir yapıldığında alınmıştır.

Tablo 5.6: Nugent problemleri evrimsel senkron hesaplamalı yöntemde haberleşme sıklıkları

	Evrimsel Senkron Hesaplamalı 100		**Evrimsel Senkron Hesaplamalı 50**				**Evrimsel Senkron Hesaplamalı 25**			
Problem	Süre (s)	İterasyon	Süre (s)	Süre İyileşme (%)	İterayon	İterasyon İyileşme (%)	Süre (s)	Süre İyileşme (%)	İterayon	İterasyon İyileşme (%)
nug12	0,45	248	0,45	1,37	209	15,73	0,45	0,22	194	21,77
nug14	0,46	225	0,47	-2,68	209	7,33	0,47	-1,61	197	12,67
nug15	0,45	207	0,45	-0,68	177	14,73	0,47	-3,38	183	11,47
nug16a	0,48	258	0,46	4,54	220	14,73	0,51	-4,92	247	4,46
nug16b	0,50	269	0,47	6,72	281	-4,46	0,51	-1,65	252	6,23
nug17	0,49	239	0,45	7,85	217	9,41	0,50	-2,08	232	3,03
nug18	0,59	411	0,54	9,53	411	0,00	0,61	-3,84	401	2,49
nug20	0,59	365	0,59	-1,30	425	-16,44	0,67	-14,32	438	-19,93
nug21	0,58	341	0,57	1,76	348	-1,91	0,63	-7,64	368	-7,77
nug22	0,62	357	0,58	6,45	336	6,02	0,68	-10,39	403	-12,82
nug24	0,68	445	0,68	0,97	436	2,02	0,81	-18,70	519	-16,63
nug25	0,63	383	0,68	-8,20	432	-12,79	0,74	-17,59	435	-13,58
nug27	1,57	1577	1,78	-13,41	1756	-11,32	2,02	-28,72	1762	-11,75
nug28	1,32	1159	1,35	-2,66	1120	3,41	1,76	-33,99	1362	-17,54
nug30	1,02	681	1,04	-2,36	660	3,16	1,13	-11,27	661	2,90
Ort.	**0,70**	**478**	**0,70**	**0,53**	**482**	**1,98**	**0,80**	**-10,66**	**510**	**-2,33**

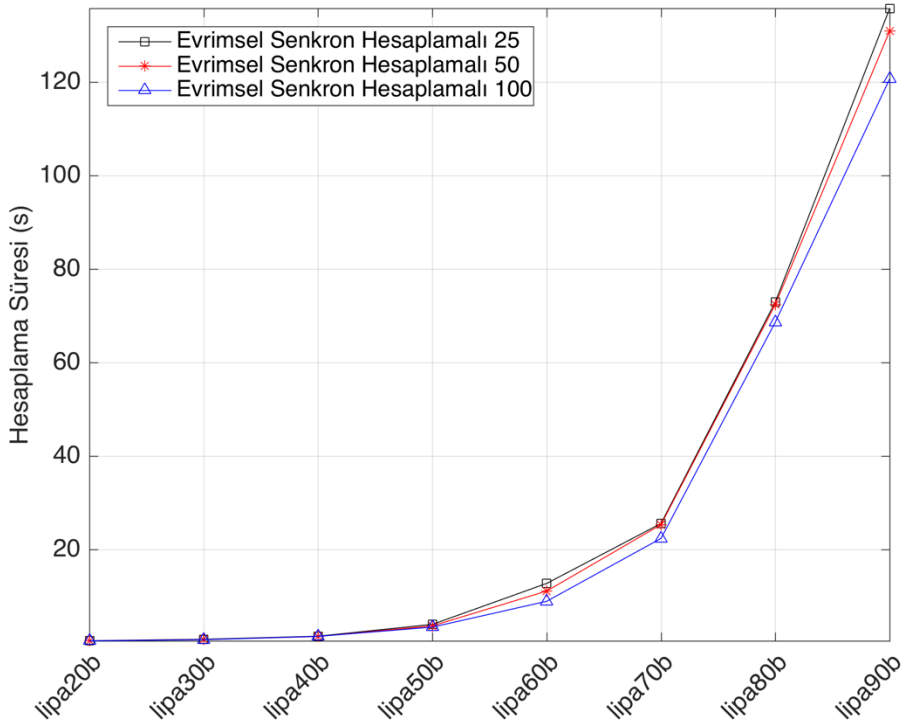


Şekil 5.10: Nugent problemleri evrimsel senkron hesaplamalı yöntemde 25, 50 ve 100 hesaplama süreleri

Lipa problemleri için evrimsel senkron hesaplamalı yöntemde 50 ve 25 iterasyonda bir haberleşme yapıldığında 100 iterasyona göre karşılaştırma değerleri Tablo 5.7'de verilmiştir. Ortalama olarak hesaplama 50 iterasyonda bir haberleşme yapıldığında 100 iterasyona göre %7,41 ve 25 iterasyonda bir haberleşme yapıldığında %14,02 kötüleşme gözlemlenmiştir. Sadece 50 iterasyonda haberleşme yapıldığında lipa20b problemi için daha iyi sonuç vermiş, bunun dışındaki tüm problemlerde 100 iterasyonda bir haberleşme yapıldığında daha iyi sonuçlar alınmıştır. Lipa problemlerinde hesaplama süreleri Şekil 5.11'de görüldüğü gibi haberleşme sıklığı arttıkça süre de genel olarak artmıştır.

Tablo 5.7: Lipa problemleri evrimsel senkron hesaplamalı yöntemde haberleşme sıklıkları

	Evrimsel Senkron Hesaplamalı 100		Evrimsel Senkron Hesaplamalı 50				Evrimsel Senkron Hesaplamalı 25			
Problem	Süre	İterasyon	Süre	Süre İyileşme (%)	İterayon	İterasyon İyileşme (%)	Süre	Süre İyileşme (%)	İterayon	İterasyon İyileşme (%)
lipa20b	0,49	227	0,49	0,14	195	14,10	0,51	-4,70	192	15,53
lipa30b	0,74	401	0,76	-2,38	388	3,24	0,83	-12,09	411	-2,56
lipa40b	1,45	745	1,42	2,12	682	8,46	1,48	-2,64	758	-1,74
lipa50b	3,43	1327	3,69	-7,63	1338	-0,79	4,01	-16,97	1612	-21,46
lipa60b	8,95	2460	11,14	-24,43	2800	-13,82	12,78	-42,71	3444	-40,01
lipa70b	22,41	4327	25,35	-13,12	4975	-14,98	25,60	-14,24	4804	-11,01
lipa80b	68,67	10060	72,45	-5,50	9803	2,56	73,00	-6,31	9742	3,17
lipa90b	120,69	13009	130,93	-8,49	13099	-0,69	135,75	-12,48	12887	0,94
Ort.	**28,35**	**4070**	**30,78**	**-7,41**	**4160**	**-0,24**	**31,75**	**-14,02**	**4231**	**-7,14**



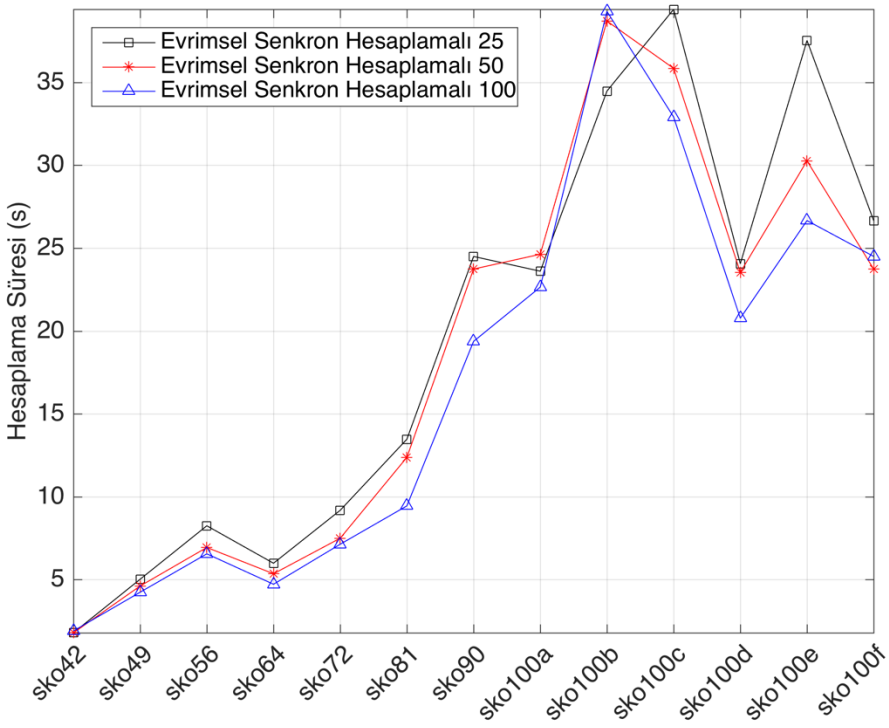
Şekil 5.11: Lipa problemleri evrimsel senkron hesaplamalı yöntemde 25, 50 ve 100 hesaplama süreleri

Skorin problemleri için evrimsel senkron hesaplamalı yöntemde 50 ve 25 iterasyonda bir haberleşme yapıldığında 100 iterasyona göre karşılaştırma değerleri Tablo 5.8'de verilmiştir. Ortalama olarak hesaplama sürelerinde 50 iterasyonda bir haberleşme yapıldığında 100 iterasyona göre %9,29 ve 25 iterasyonda bir haberleşme yapıldığında %18,49 kötüleşme gözlemlenmiştir. Şekil 5.12'de verilen hesaplama süreleri grafiğine bakıldığında bazı problemlerde 25, bazı problemlerde 50

iterasyonda bir haberleşme yapıldığında daha iyi sonuçlar alınmasına rağmen genele bakıldığında haberleşme sıklığı arttığında hesaplama süresi artmıştır.

Tablo 5.8: Skorin problemleri evrimsel senkron hesaplamalı yöntemde haberleşme sıklıkları

	Evrimsel Senkron Hesaplamalı 100		**Evrimsel Senkron Hesaplamalı 50**				**Evrimsel Senkron Hesaplamalı 25**			
Problem	Süre	İterasyon	Süre	Süre İyileşme (%)	İterayon	İterasyon İyileşme (%)	Süre	Süre İyileşme (%)	İterayon	İterasyon İyileşme (%)
sko42	1,90	875	1,78	5,92	884	-0,97	1,79	5,77	873	0,29
sko49	4,23	1756	4,61	-8,84	1808	-2,96	5,01	-18,49	1787	-1,74
sko56	6,55	2130	6,93	-5,86	2014	5,45	8,24	-25,85	2149	-0,87
sko64	4,71	1056	5,35	-13,42	1123	-6,34	5,98	-26,90	1203	-13,90
sko72	7,13	1240	7,49	-5,09	1246	-0,48	9,18	-28,74	1498	-20,81
sko81	9,44	1250	12,37	-31,01	1582	-26,52	13,47	-42,61	1670	-33,60
sko90	19,37	2007	23,74	-22,54	2319	-15,52	24,50	-26,46	2336	-16,38
sko100a	22,65	1783	24,63	-8,76	1807	-1,32	23,61	-4,24	1763	1,14
sko100b	39,29	2889	38,71	1,48	2759	4,52	34,48	12,24	2664	7,79
sko100c	32,91	2394	35,85	-8,93	2748	-14,77	39,41	-19,75	2854	-19,23
sko100d	20,77	1626	23,53	-13,31	1630	-0,22	24,06	-15,84	1681	-3,38
sko100e	26,66	1968	30,27	-13,52	2152	-9,35	37,53	-40,75	2615	-32,86
sko100f	24,50	1800	23,74	3,12	1793	0,42	26,66	-8,81	1900	-5,54
Ort.	**16,93**	**1752**	**18,38**	**-9,29**	**1835**	**-5,24**	**19,53**	**-18,50**	**1922**	**-10,70**



Şekil 5.12: Skorin problemleri evrimsel senkron hesaplamalı yöntemde 25, 50 ve 100 hesaplama süreleri

6. SONUÇLAR VE ÖNERİLER

Bu tez çalışması kapsamında KAP için tavlama benzetimi yöntemi MATLAB ortamında farklı paralelleştirme yöntemleri kullanılarak hesaplama süresi ve gereken iterasyonun seri çalışan tavlama benzetimine göre karşılaştırması yapılmıştır.

Çalışmada maliyet fonksiyonunun paralelleştirilmesinin hesaplama süresini azaltmadığı, aksine artırdığı gözlemlenmiştir. Bunun sebebi maliyet fonksiyonunun paralelleştirilmesi maliyetinin hesaplama maliyetinden fazla olmasıdır.

Belirli aralıklarla haberleşmenin yapıldığı ve en iyi sonucun ara başlangıç çözümü olarak kullanıldığı paralel senkron hesaplamalı yöntemin yerel minimumlara takılmayı artırdığı, istenilen hata oranına ulaşmada başarısız olduğu gözlemlenmiştir. Senkron hesaplamalı yöntem seri çalışan tavlama benzetimi algoritmasından daha kötü sonuçlar vermiştir.

Haberleşmenin yapılmadığı asenkron hesaplamalı yöntemde ve belirli aralıklarla haberleşme yapılıp çözümlerden en iyi 6 tanesi, kalan 6 çözümden 2 tanesi ve rastgele 4 yeni çözümün ara başlangıç çözümü olarak alındığı evrimsel senkron hesaplamalı yöntemlerinin seri yönteme göre hesaplama süreleri ve iterasyon sayıları bakımından oldukça iyi sonuçlar elde edilmiştir.

Asenkron hesaplamalı yöntem ile evrimsel senkron hesaplamalı yöntemi kıyaslandığında ise asenkron hesaplamalı yöntem genel olarak daha iyi sonuçlar vermiştir. Evrimsel senkron hesaplamalı yöntemde haberleşme ve ara başlangıç çözümü olarak kullanılacak çözümlerin sıralanması ek maliyet getirdiğinden hesaplama süreleri daha fazla çıkmıştır.

Evrimsel senkron hesaplamalı yönteminde 25, 50 ve 100 iterasyonda bir haberleşme yapıldığında genel olarak 100 iterasyonda bir haberleşme yapıldığında daha iyi sonuçlar elde edilmiştir.

Yapılan çalışma GİB kullanılarak ilerletilebilir. Görüntü işlemek için geliştirilen GİB'ler günümüzde paralel çalışan yoğun aritmetik işlemler için de

oldukça sık kullanılmaktadır. GİB'ler işlemciye göre oldukça fazla paralel işlem çalıştırmaya olanak sağlamaktadır. Ayrıca GİB'ler paralel hesaplamaları işlemciye göre çok daha hızlı yaptığından geliştirilen algoritmalar GİB üzerinde de gerçekleştirilebilir.

KAYNAKLAR

Bazaraa, M. S. and Sherali, H. D., "Benders' partitioning scheme applied to a new formulation of the quadratic assignment problem", *Naval Research Logistics Quarterly*, 27(1), 29–41, doi:10.1002/nav.3800270104, (1980).

Bos, J., "Zoning in Forest Management: a Quadratic Assignment Problem Solved by Simulated Annealing", *Journal of Environmental Management*, 37(2), 127–145, doi:10.1006/jema.1993.1010, (1993).

Burkard, R. E., Dell'Amico, M. and Martello, S., *Assignment Problems, Revised Reprint,* Philedelphia: Society for Industrial and Applied Mathematics (SIAM), (2009).

Burkard, R. E., Karisch, S. E. and Rendl, F., "QAPLIB A Quadratic Assignment Problem Library", *Journal of Global Optimization*, 10(4), 391–403, doi:10.1023/A:1008293323270, (1997).

Çela, E., *The Quadratic Assignment Problem: Theory and Algorithms*, Boston, MA: Springer, doi:10.1007/978-1-4757-2787-6, (1998).

Chen, H., Flann, N. S. and Watson, D. W., "Parallel genetic simulated annealing: a massively parallel SIMD algorithm", *IEEE Transactions on Parallel and Distributed Systems*, 9(2), 126–136, doi:10.1109/71.663870, (1998).

Christofides, N. and Benavent, E., "An Exact Algorithm for the Quadratic Assignment Problem on a Tree", *Operations Research*, 37(5), 760–768, doi:10.1287/opre.37.5.760, (1989).

Dickey, J. W. and Hopkins, J. W., "Campus building arrangement using TOPAZ", *Transportation Research*, 6(1), 59–68, (1972).

Duman, E. and Or, I. "The quadratic assignment problem in the context of the printed circuit board assembly process", *Computers & Operations Research*, 34(1), 163–179, doi:10.1016/j.cor.2005.05.004, (2007).

Elshafei, A. N., "Hospital layout as a quadratic assignment problem", *Operational Research Quarterly*, 28(1), 167–179, (1977).

Ferreiro, A. M., García, J. A., López-Salas, J. G. and Vázquez, C., "An efficient implementation of parallel simulated annealing algorithm in GPUs", *Journal of Global Optimization*, 57(3), 863–890, doi:10.1007/s10898-012-9979-z, (2013).

Forsberg, J. H., Delaney, R. M., Zhao, Q., Harakas, G. and Chandran, R., "Analyzing Lanthanide-Induced Shifts in the NMR Spectra of Lanthanide(III) Complexes Derived from 1,4,7,10-Tetrakis(N,N-diethylacetamido)-1,4,7,10-tetraazacyclododecane", *Inorganic Chemistry*, 34(14), 3705–3715, doi:10.1021/ic00118a018, (1995).

Francis, R. L. and White, J. A., *Facility layout and location: an analytical approach*, Prentice-Hall, (1974).

Gamal A. E. A. S., Abeer M. M. and El-Sayed, M. H., "A Comparative Study of Meta-heuristic Algorithms for Solving Quadratic Assignment Problem", *International Journal of Advanced Computer Science and Applications(IJACSA)*, 5(1), 1–6, doi:10.14569/IJACSA.2014.050101, (2014).

Geoffrion, A. M. and Graves, G. W. "Scheduling Parallel Production Lines with Changeover Costs: Practical Application of a Quadratic Assignment/ LP Approach", *Operations Research*, 24(4), 595–610, doi:10.1287/opre.24.4.595, (1976).

Ghandeshtani, K. S., Mollai, N., Neshati, M. M. and Seyedkashi, S. M. H., "New simulated annealing algorithm for quadratic assignment problem", *The Fouth International Conference on Advanced Engineering Computing and Applications in Sciences* (87–92), (2010).

Gilmore, P. C., "Optimal and Suboptimal Algorithms for the Quadratic Assignment Problem", *Journal of the Society for Industrial and Applied Mathematics*, 10(2), 305–313, doi:10.1137/0110022, (1962).

Glover, F., "Heuristics For Integer Programming Usıng Surrogate Constraints", *Decision Sciences*, 8(1), 156–166, doi:10.1111/j.1540-5915.1977.tb01074.x, (1977).

Glover, F., "Tabu Search—Part I", *ORSA Journal on Computing*, 1(3), 190–206, doi:10.1287/ijoc.1.3.190, (1989).

Heffley, D. R., "The Quadratic Assignment Problem: A Note", *Econometrica*, 40(6), 1155, doi:10.2307/1913863, (1972).

Holland, J. H., *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*, University of Michigan Press, (1975).

Hubert, L. J., *Assignment methods in combinatorial data analysis* (C. 73), New York: M. Dekker, (1987).

Kaviani, M. A., Abbasi, M., Rahpeyma, B. and Yusefi, M. M., "A hybrid Tabu search-simulated annealing method to solve quadratic assignment problem", *Decision Science Letters*, 3(3), 391–396, doi:10.5267/j.dsl.2014.2.004, (2014).

Kendall, G., "AI Methods [online]", (12 May 2016), http://www.cs.nott.ac.uk/~pszgxk/aim/notes/simulatedannealing.doc, (2016).

Kirkpatrick, S., Gelatt, C. D. and Vecchi, M. P., "Optimization by Simulated Annealing", *Science*, 220(4598), 671–680, (1983).

Koopmans, T. C. and Beckmann, M., "Assignment Problems and the Location of Economic Activities", *Econometrica*, 25(1), 53, doi:10.2307/1907742, (1957).

Krarup, J. and Pruzan, P., *Computer-aided layout design*, Berlin Heidelberg: Springer, doi:10.1007/BFb0120827, (1978).

Lawler, E. L., "The Quadratic Assignment Problem", *Management Science*, 9(4), 586–599, doi:10.1287/mnsc.9.4.586, (1963).

Li, Y. and Pardalos, P. M., "Generating quadratic assignment test problems with known optimal permutations", *Computational Optimization and Applications*, 1(2), 163–184, doi:10.1007/BF00253805, (1992).

Loiola, E. M., de Abreu, N. M. M., Boaventura-Netto, P. O., Hahn, P. and Querido, T., "A survey for the quadratic assignment problem", *European Journal of Operational Research*, 176(2), 657–690, doi:10.1016/j.ejor.2005.09.032, (2007).

Mathworks Inc., *MATLAB Parallel Computing Toolbox 2016a User Guide*, (2016).

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. and Teller, E., "Equation of state calculations by fast computing machines", *Journal Chemical Physics*, 21(6), 1087–1092, doi:10.1063/1.1699114, (1953).

Nugent, C. E., Vollmann, T. E. and Ruml, J., "An Experimental Comparison of Techniques for the Assignment of Facilities to Locations", *Operations Research*, doi:10.1287/opre.16.1.150, (1968).

Onbaşoğlu, E. and Özdamar, L., "Parallel Simulated Annealing Algorithms in Global Optimization", *Journal of Global Optimization*, 19(1), 27–50, doi:10.1023/A:1008350810199, (2001).

Padberg, M. and Rinaldi, G., "A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems", *SIAM Review*, 33(1), 60–100, doi:10.1137/1033004, (1991).

Paul, G., "Comparative performance of tabu search and simulated annealing heuristics for the quadratic assignment problem", *Operations Research Letters*, 38(6), 577–581. doi:10.1016/j.orl.2010.09.009, (2010).

Paul, G., "A GPU implementation of the Simulated Annealing Heuristic for the Quadratic Assignment Problem", *CoRR*, *abs/1208.2*., http://arxiv.org/abs/1208.2675, (2012).

Pollatschek, M. A., Gershoni, H. and Radday, Y. T., "Optimization Of Typewriter Keyboard By Computer-Simulation", *Angewandte Informatik*, (10), 438–439, (1976).

Rao, S. S., *Engineering Optimization: Theory and Practice* (4th Edition, pp. 702–708), New Jersey:Wiley, (2009).

Saifullah Hussin, M. and Stützle, T., "Tabu search vs. simulated annealing as a function of the size of quadratic assignment problem instances", *Computers and Operations Research*, 43, 286–291, doi:10.1016/j.cor.2013.10.007, (2014).

Skorin-Kapov, J., "Tabu Search Applied to the Quadratic Assignment Problem", *ORSA Journal on Computing*, 2(1), 33–45, doi:10.1287/ijoc.2.1.33, (1990).

Steinberg, L., "The Backboard Wiring Problem: A Placement Algorithm", *SIAM Review*, 3(1), 37–50, doi:10.1137/1003003, (1961).

Tian, P., Ma, J. and Zhang, D. M., "Application of the simulated annealing algorithm to the combinatorial optimization problem with permutation property: An investigation of generation mechanism", *European Journal of Operational Research*, 118(1), 81–94, doi:10.1016/S0377-2217(98)00308-7, (1999).

Timur, K. ve Söyler, H., "Global karınca kolonisi optimizasyonu", *Gazi Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi*, 21(4), 689–698, (2006).

Wang, J. C., "A Multistart Simulated Annealing Algorithm for the Quadratic Assignment Problem", *Third International Conference on Innovations in Bio-Inspired Computing and Applications*, 19–23, doi:10.1109/IBICA.2012.56, (2012).

ÖZGEÇMİŞ

Adı Soyadı	: Selahattin AKKAŞ
Doğum Yeri ve Tarihi	: Osmancık 22/07/1991
Lisans Üniversite	: Pamukkale Üniversitesi
Elektronik posta	: sakkas@pau.edu.tr
İletişim Adresi	: Pamukkale Üniversitesi Bilgisayar Mühendisliği Bölümü 20070 DENİZLİ

Konferans listesi :

• Selahattin AKKAŞ ve Kadir KAVAKLIOĞLU, "Paralelleştirilmiş Tavlama Benzetimi Teknikleri Kullanılarak Karesel Atama Problemlerinin Çözümü", *Otomatik Kontrol Türk Milli Komitesi Ulusal Toplantısı TOK'2015*, 390-395, (2015).