Dicle University Journal of Engineering (DUJE)
web: http://dergipark.gov.tr/dumf

Araştırma Makalesi / Research Article

Sıra-Bağımlı Hazırlık Zamanlı Genel Montaj Hattı Dengeleme Problemlerinin Çözümü İçin Bir Diferansiyel Gelişim Algoritması

A Differential Evolution Algorithm for The Solution of General Assembly Line Balancing Problems with Sequence-Dependent Setup Times

Şehmus Aslan[1*], Mehmet Aytekin[2]

[1] Mardin Artuklu Üniversitesi, İşletme Bölümü, Mardin, sehmusaslan@artuklu.edu.tr, ORCID: 0000-0003-1886-3421
[2] Gaziantep Üniversitesi, İşletme Bölümü, Gaziantep, aytekin@gantep.edu.tr, ORCID: 0000-0001-5464-0677

MAKALE BİLGİLERİ

Makale geçmişi:

Geliş: 26 Şubat 2020
Düzeltme: 21 Nisan 2020
Kabul: 27 Nisan 2020

Anahtar kelimeler:

Montaj hattı dengeleme, sıra-bağımlı hazırlık zamanlı, diferansiyel gelişim algoritması

ÖZET

Basit Montaj Hattı Dengeleme Problemleri (BMHDP) ile ilgili literatürde birçok çalışma yapılmıştır. Ancak BMHDP'de bulunan kısıtlardan dolayı yapılan akademik çalışmalar ve endüstrideki uygulamalar arasında büyük bir boşluk bulunmaktaydı. Bu boşluğun kapatılması için Genel Montaj Hattı Dengeleme Problemleri (GMHDP) adı altında daha çok endüstrinin pratik sorunlarını çözmeye yönelik çalışmalar başlamıştır. Otomotiv ve elektronik sektöründe sıkça rastlanan sıra-bağımlı hazırlık zamanları, daha önce yapılan Montaj Hattı Dengeleme (MHD) çalışmalarında ele alınmamıştır. Daha önceleri, MHD çalışmalarında hazırlık zamanları, istasyon zamanlarına eklenerek problemler çözülüyordu. Bu yaklaşım sorunu çözmede yetersiz kaldığı için sıra-bağımlı hazırlık zamanlı GMHDP çalışmaları ortaya çıkmıştır. Sıra-bağımlı hazırlık zamanlı GMHDP, NP-zor yapıda ve çok karmaşık problemler olduğundan çözümü için lineer programlama ve dal-sınır algoritması gibi belirli (deterministik) yöntemler, makul zamanlarda çözüm üretememektedir. Bu çalışmada ise problemlerin çözümünde bir metasezgisel yöntem olan yeni bir Diferansiyel Gelişim Algoritması (DGA) geliştirilmiştir. Geliştirilen DGA'nın performansı literatürdeki test problemleri üzerinde denenmiş ve literatürde daha önce geliştirilmiş sezgisel yöntemlerden daha iyi sonuçlar vermiştir.

Doi: 10.24012/dumf.694846

ARTICLE INFO

Article history:

Received: 26 February 2020
Revised: 21 April 2020
Accepted: 27 April 2020

Keywords:

Assembly line balancing, sequence-dependent setup times, differential evolution algorithm

ABSTRACT

Many studies have been conducted in the literature on Simple Assembly Line Balancing Problems (SALBP). However, there was a huge gap between academic studies and industry practices due to the limitations found in SALBP. In order to close this gap, studies under the field of the General Assembly Line Balancing Problems (GALBP) have been started to solve the practical problems of the industry. The sequence-dependent setup times, which are common in the automotive and electronics sectors, have not been addressed in previous Assembly Line Balancing (ALB) studies. Setup times were added to station times to solve the problems in previous ALB studies. As this approach is insufficient to solve the problem, the sequence-dependent setup times GALBP studies have emerged. Because the sequence-dependent setup times GALBP is NP-hard and very complex problems, they can not be solved in reasonable time by deterministic methods such as linear programming and branch and bound algorithm. In this study, a new Differential Evolution Algorithm (DEA), which is a metaheuristic, has been developed to solve these problems. The performance of the developed DGA was tested on the test problems in the literature and gave better results than the previously developed heuristic methods in the literature.

* Sorumlu yazar / Correspondence
Şehmus ASLAN
✉ sehmusaslan@artuklu.edu.tr

Please cite this article in press as S. Aslan,M. Aytekin, "Sıra-Bağımlı Hazırlık Zamanlı Genel Montaj Hattı Dengeleme Problemlerinin Çözümü İçin Bir Diferansiyel Gelişim Algoritması", DUJE, vol. 11, no.3, pp. 1103-1118, September 2020.

Giriş

Sürekli üretim sistemlerinde, üretimin birimler halinde gerçekleştirildiği ve kitle talebinin olduğu durumlarda, yüksek üretim hızıyla talebi karşılamanın en makul yolu montaj hatlarının yapılandırılmasıdır. Üretim sistemlerinde önemli bir birim olan montaj hatları, bünyesinde bir veya daha çok işlem yapılan ve birbiri ardına dizilmiş olan iş istasyonlarından oluşmaktadır. Her bir istasyonda, çevrim zamanı denilen sınırlı bir zaman dilimi içerisinde bir veya daha fazla montaj işlemi yapılır. Montaj Hattı Dengeleme (MHD), montaj işleminin yapılabilmesi için gerekli işlerin, bu işlerin süreleri ve işler arasındaki öncelik ilişkileri verildiğinde, bir performans ölçütü en iyilenecek şekilde, sıralı iş istasyonlarına atanması şeklinde tanımlanabilir [1]. MHD'nin ana amaçlardan biri, her iş istasyonuna eşit miktarda iş dağıtımı yapabilmektir. Başka bir deyişle, toplam iş yükünü iş istasyonları arasında mümkün olduğu kadar eşit bir şekilde bölebilmektir. Bir Basit Montaj Hattı Dengeleme Problemi (BMHDP)'nde temel olarak aşağıdaki kısıtlar kullanılmaktadır:

- Bütün işler iş istasyonlarına atanmalıdır,
- Bir görev sadece bir iş istasyonuna atanabilir,
- İş istasyonuna atanan görevlerin toplam süreleri çevrim süresini geçemez,
- Görevler arasındaki teknolojik ilişkilerden kaynaklanan öncül-ardıl ilişkileri bozulamaz.

Amaç fonksiyonu göz önüne alındığında MHDP, 4 farklı gruba ayrılabilir [2]:

1. MHDP tip-1: Çevrim zamanı verilir, istasyon sayısı minimize edilmeye çalışılır.
2. MHDP tip-2: İstasyon sayısı verilir, çevrim zamanı minimize edilmeye çalışılır.
3. MHDP tip-E: Hat verimliliğini maksimize etmek için, istasyon sayısı ve çevrim zamanı aynı anda minimize edilmeye çalışılır.
4. MHDP tip-F: Verilen istasyon sayısı ve çevrim zamanı için problemin uygun (feasible) olup olmadığına bakılır. Uygun çözümler bulunmaya çalışılır.

BMHDP, montaj hattı dengeleme problemlerinin en basit ve orijinal halidir. Bu problem tipine paralel istasyonlar, bölgeleme kısıtları, kaynak kısıtları gibi bazı kısıtlamalar veya faktörler eklenirse problem Genel Montaj Hattı Dengeleme Problemleri (GMHDP) olarak adlandırılır. Literatürde GMHDP, U-tipi, paralel istasyonlar, karışık modelli vb. çeşitleri mevcuttur.

Literatür Araştırması

MHD literatürüne bakıldığında BMHDP ile ilgili çok sayıda çalışma yapıldığı görülmektedir. BMHDP'de kullanılan kısıtlar gerçek hayattaki montaj hatlarını yansıtmaktan uzak kalmasına sebep oluyordu. Bu nedenle de akademik çalışmalar ile pratikteki uygulamalar arasındaki boşluğun doldurulması için bu çalışmalarda, daha gerçekçi yaklaşımlar sunan GMHDP altında yapılmaya başlandı [2]. Daha detaylı bilgi için, Becker ve Scholl'a [2] bakılabilir. Andres vd. [3] literatürde başka bir boşluğu doldurmak için sıra-bağımlı hazırlık zamanlı GMHDP kavramını ortaya atmışlardır. Literatürde ilk defa bahsedilen bu kavram aslında pratikte sık rastlanan bir durumdur. Bir çok endüstriyel montaj hattında hazırlık zamanları olmasına rağmen, işlem (proses) zamanına göre az oranda olduğu için gözardı edilmektedir. Ancak; otomotiv, elektronik ve diğer bazı robotik üretim hatlarında işlem zamanları kısa olduğu için hazırlık zamanlarının işlem zamanına göre oranı yüksek olabilmektedir. Dolayısıyla daha verimli bir montaj hattı dengelemesi için bu hazırlık zamanlarının göz önünde bulundurulması gerekmektedir.

GMHDP ile ilgili çalışmalara bakıldığında; Andres vd. [3], çalışmalarında sıra-bağımlı hazırlık zamanlı tip-1 GMHDP için bir tamsayılı doğrusal programlama modeli

sundukları ve çözüm için de 8 farklı sezgisel ve GRASP algoritması geliştirdikleri görülmektedir. Özcan ve Toklu [4] sıra-bağımlı hazırlık zamanlı çift taraflı GMHDP için karma tamsayılı bir doğrusal programlama modeli sunmuşlardır. Ayrıca modelin çözümü için de COMSOAL tabanlı bir sezgisel geliştirmişlerdir. Yolmeh ve Kianfar [5], sıra-bağımlı hazırlık zamanlı tip-2 GMHDP için hibrid bir genetik algoritma geliştirmişlerdir. Buldukları hibrid algoritmayı literatürdeki diğer algoritmalar ile karşılaştırmışlar ve daha iyi sonuçlar bulmuşlardır. Seyed vd. [6], sıra-bağımlı hazırlık zamanlı tip-2 GMDHP'nin çözümü için bir tavlama benzetimi algoritması geliştirmişler ve parametre optimizasyonu için de Taguchi yöntemini kullanmışlardır. Scholl vd. [7] çalışmalarında, Andres vd. yaptıkları çalışmaya ek olarak hem geri (backward), hem de ileri (forward) hazırlık zamanlarını kullanmışlardır. Ayrıca yeni bir tamsayılı model kurmuşlar ve model çözümü için de çeşitli sezgisel algoritmalar önermişlerdir. Bu sezgisel algoritmalar geçmiş sezgisel çözümlerden daha iyi sonuçlar elde etmesine rağmen hazırlık zamanlarının oransal olarak büyük olduğu test problemlerinde çok iyi sonuçlar bulamamışlardır. Hamta vd. [8], sıra-bağımlı hazırlık zamanlı tip-2 GMDHP için çok amaçlı bir matematiksel model geliştirmişlerdir. Bu modelin amaçları çevrim süresini, malzeme maliyetlerini ve istasyonlar arasındaki düzgünlük indeksini minimize etmektir. Problem NP-zor olduğu için büyük problemler için matematiksel modellerle makul çözümler bulunması çok zor olmaktadır. Bu nedenle, çözüm için değişken komşu arama sezgiseli ile parçacık sürü meta-sezgiseliyle hibridize edilen bir algoritma sunmuşlardır. Literatürdeki diğer yöntemlerle karşılaştırıldığında geliştirilen algoritma hem optimal çözüme yakın çözümler bulma kabiliyeti, hem de çözüme ulaşma süresi bakımından daha iyi sonuçlar vermiştir. Akpınar vd. [9], karışık modelli hazırlık zamanlı GMHDP için karma tamsayılı doğrusal programlama modeli geliştirmiş ve problemin karmaşık olmasından dolayı çözüm için karınca algoritması ve genetik algoritmayı hibridize ederek bir çözüm algoritması geliştirmişlerdir. Diri vd. [10], stokastik sıra-bağımlı hazırlık zamanlı GMHDP için doğrusal olmayan karma tamsayılı programlama modeli sunmuşlardır. Janardhanan vd. [11], sıra-bağımlı hazırlık zamanlı robotik tip-2 GMHDP için karma tamsayılı doğrusal programlama modeli geliştirmişlerdir. Problem yapısı NP-zor olduğundan dolayı çözüm için göç eden kuşlar metasezgiselini kullanmışlardır. Söz konusu metasezgiseli literatürdeki diğer meta-sezgiseller ile karşılaştırmışlardır. Bulunan çözümlerde göç eden kuşlar algoritmasının diğerlerine göre daha iyi sonuçlar verdiği gözlenmiştir.

Gutjahr ve Nemhauser [12], basit montaj hattı dengeleme problemlerinin bile NP-zor yapıda olduğunu ispatlamışlardır . Dolayısıyla çok daha karmaşık olan sıra-bağımlı hazırlık zamanlı GMHDP'nin de NP-zor olduğunu söyleyebiliriz. NP-zor yapıdaki problemlerin doğrusal programlama, dinamik programlama ve dal-sınır algoritmaları gibi kesin yöntemlerle makul zamanlarda en iyi çözümlerinin bulunması büyük problemler için imkansızdır. Bu nedenle bu tip problemlerin çözümü için sezgisel veya metasezgisel algoritmalar kullanılmaktadır. Bu çalışmada da çözüm için yeni bir Diferansiyel Gelişim meta-sezgisel algoritması geliştirilmiştir. Diferansiyel Gelişim Algoritması (DGA) ilk defa Storn ve Price [13] tarafından geliştirilmiş ve sürekli eniyileme problemlerin çözümünde kullanılmıştır. DGA günümüze gelinceye kadar birçok eniyileme problemlerin çözümünde kullanılmıştır [14-18]. DGA aynı zamanda birçok MHDP'nin çözümünde de kullanılmıştır. Nearchou [19], ilk defa BMHDP için DGA yöntemini kullanmıştır. Çalışmada, literatürdeki test problemleri üzerinden diğer evrimsel algoritmalar ile karşılaştırılan DGA üstünlük sağlamıştır. Nearchou [20], çok amaçlı BMHDP için DGA geliştirmiştir. Bu çalışmada esas amaç istasyon çevrim zamanını minimize etmek, ikincil amaçlar olarak de geç kalma dengelemesini ve düzgünlük indeksini minimize etmektir. Çalışmada Kim vd. [21] tarafından geliştirilen genetik algoritmayla karşılaştırılan DGA, daha iyi sonuçlar vermiştir. Nourmohammadi ve Zandieh'in [22], çok amaçlı BMHDP'nin

çözümü için geliştirdikleri DGA'nın amaçları; çevrim zamanını ve düzgünlük indeksini minimize etmektir. Bir sonraki popülasyonda bireyleri seçerken Pareto baskınlık kavramı ve değerlendirme mekanizması için de TOPSİS yönteminden faydalanmışlardır. Taguchi yöntemi DGA parametrelerinin optimizasyonu için kullanılmıştır. Önerilen DGA'da iki adet aday vektör oluşmaktadır. Bunlardan biri çekinik diğeri ise baskın vektördür. Mozdgir vd. [23], çalışmalarında BMHDP'nin ana amaç fonksiyonu olarak işgücü düzgünlük indeksini k için DGA geliştirmişlerdir. DGA parametrelerin optimizasyonu için Taguchi yöntemini kullanmışlardır. Vincent ve Ponnambalam [24], esnek montaj hatlarını dengelemek için iki aşamalı DGA geliştirmişlerdir. Model literatürde geliştirilen iki aşamalı genetik algoritma ile test problemleri üzerinden karşılaştırılmış ve daha iyi sonuçlar verdiği gözlenmiştir. Pitakaso ve Sethanan [25], çalışmalarında tekstil endüstrisindeki montaj hattı dengeleme uygulamalarında her görev için önceden belirlenmiş bir makine atanması gerektiğini belirtmişlerdir. Bu kısıt altında tip-1 BMHDP için bir matematiksel model geliştirmişler ve çözüm için de DGA sunmuşlardır. Önerilen DGA'da farklı bir çaprazlama operatörü kullanılmıştır. Nilakantan vd. [26], robotik montaj hatlarının dengelenmesi için iki amaçlı bir DGA önermişlerdir. Bu amaçlar çevrim zamanının ve hat maliyetinin minimize edilmesidir. Bu çalışmada hem düz hem de U-tipi robotik montaj hatlarının dengelenmesi hedeflenmiştir. Literatürdeki 30 veriseti üzerinde yapılan testlerde önerilen DGA'nın etkinliği ispatlanmıştır. Zhang vd. [27], tip-2 BMHDP için bir DGA geliştirmişlerdir. Bu DGA'da klasik DGA'nın aksine çözüm gösterimleri gerçel sayılar olarak değil de tam sayı olarak gösterilmiştir. Ayrıca geliştirilen yeni bir çaprazlama operatörü herhangi bir tamir mekanizması gerektirmeden vektörler üzerinde işlem yapmaktadır. Aynı şekilde çözüm vektörleri kesikli yapıda olduğu için yeni bir mutasyon operatörü de geliştirilmiştir. Nearchou ve Omirou [28], hem tip-1 hem de tip-2 BMHDP için yeni bir DGA geliştirmişlerdir. Çözüm vektörleri öncelik tabanlı kodlama ve rastgele kodlama şeklinde kodlanmıştır. Literatürdeki test problemleri üzerinden farklı metasezgisellerle karşılaştırılan DGA'nın rastgele kodlamalı olan çözüm yöntemi üstünlük sağlamıştır. Gansterer ve Hartl [29], tek ve çift taraflı montaj hattı dengeleme problemlerinin çeşitli kısıtlar altında çözülmesi için genetik algoritma, tabu arama ve DGA metasezgisellerini kullanmışlardır. Tekstil sektöründen gerçek verilerin kullanıldığı uygulamada 52 görevli örneklere kadar her üç metasezgisel de eniyi çözümlere ulaşabilmiştir. Ancak daha büyük problemlerde genetik algoritmanın, hem tabu arama metasezgiselinden hem de DGA'dan daha iyi sonuçlar verdiği gözlenmiştir. Feriştah [30], düz ve U-tipi montaj hatları için bir DGA geliştirmiştir. Önerilen algoritma literatürdeki diğer algoritmalar ile karşılaştılmış ve daha iyi sonuçlar verdiği gözlenmiştir. Önerilen algoritmanın üstünlüğü kromozom yapısından kaynaklanmaktadır. Kromozom yapısı öncelik ilişkilerine öncelik vermekte olup, herhangi bir tamir operasyonuna gerek duymamaktadır.

Andres vd.'nin [3] ortaya koydukları sıra-bağımlı hazırlık zamanlı MHDP çok karmaşık bir GMHDP'dir. Bu problem türünde istasyonlar arasında dengeleme yapılırken aynı zamanda istasyon içindeki görevler arasında da çizelgeleme yapılması gerekir. Dolayısıyla, sıra-bağımlı hazırlık zamanlı GMHDP, hem bir dengeleme hem de bir çizelgeleme problemidir. Daha önce sezgisellerle çözülmeye çalışılan sıra-bağımlı hazırlık zamanlı GMHDP tip-1 problemi, daha sonra da bir kaç çalışmada GMHDP tip-2 problemi olarak metasezgisellerle çözülmeye çalışılmıştır. Ancak, sıra-bağımlı hazırlık zamanlı GMHDP tip-1 problemleri literatürde ilk defa bu çalışmada bir metasezgisel olan DGA ile çözülmüştür. Yukarıda da söylendiği gibi sürekli problemlere başarıyla uygulanan DGA'nın, çözüm kümesi ayrık olan NP-zor problemlere uygulaması literatürde çok fazla rastlanılmamaktadır. Bu çalışmada DGA yönteminin seçilmesinde diğer bir neden, diğer evrimsel algoritmalara göre DGA'nın daha basit ve etkili bir yöntem olmasıdır. DGA'da kullanılan operatör sayısı diğer evrimsel

algoritmalara göre daha azdır. Dolayısıyla diğer evrimsel algoritmalara göre uygulanması da daha kolay olmaktadır.

Çalışmanın izleyen bölümünde problemin daha iyi anlaşılabilmesi için problem tanımı detaylı bir şekilde anlatılmıştır. Daha sonra çalışmada kullanılan yöntem DGA açıklanmıştır. Ardından bu problem çeşidini çözmek için geliştirilen DGA'nın geliştirilme aşamaları detaylı bir şekilde açıklanmıştır. Sonrasında geliştirilen DGA, test problemleri üzerinden literatürdeki sezgisellerle karşılaştırılmış ve deney sonuçları açıklanmıştır. Son bölümde ise sonuçlar ve gelecek çalışmalar için öneriler verilmiştir.

Problem Tanımı

BMHDP'de, görevler arasında hazırlık zamanları varsa ve görev sırasına bağımlı olarak değişiyorsa bu tarz problemlere sıra-bağımlı hazırlık zamanlı GMHDP denmektedir. Andres vd.'den [3] önce hazırlık zamanları dikkate alınmamıştır. Fakat robotik hatlar, otomotiv ve elektronik montaj hatları gibi hazırlık zamanlarının yüksek oranda olduğu hatlarda hazırlık zamanlarınının gözardı edilmesi verimli bir dengeleme yapılmasını engel olmaktadır. Andres vd.'ne [3] göre robotik hatlarda bir görevden diğerine geçildiği zaman bazı aletlerin değiştirilmesi için bir hazırlık zamanına ihtiyaç duyulmaktadır. Becker ve Scholl [31], otomotiv montaj hatlarında araç gövdesi üzerinde çalışan işçiler için yürüme zamanlarının hesaba katılması gerektiğini savunmuşlardır. Ayrıca yürüme zamanlarının öncül-ardıl olan işe göre değişebildiğini söylemişlerdir.

Bu problem şu şekilde tanımlanabilir: Bir i görevi herhangi bir j görevinin öncülü ise ve aynı istasyona atandıysa görevler arasında $\tau_{i,j}$ kadar bir hazırlık zamanı gerekebilir. Bu süreye ileri (forward) hazırlık zamanı denilmekte ve istasyonun toplam zamanına eklenmesi gerekmektedir. Scholl vd. [7], buna ek olarak geri (backward) hazırlık zamanını eklemiştir. Eğer bir istasyonda p son görev olarak yer alıyor ve i de birinci görev olarak yer alıyorsa, iki görev arasında $\mu_{p,i}$, kadar bir geri hazırlık zamanı gerekebilir. Dolayısıyla bu geri hazırlık zamanının da istasyon süresine eklenmesi gerekir.

Problemin daha iyi anlaşılması açısından şöyle bir örnek verilebilir: 3 görevden oluşan ve aralarında öncül-ardıl ilişkisi olmayan A,B,C görevleri aynı istasyonda yer almaktadır. Görev zamanları: τ_A=15, τ_B=12, τ_C=10 olsun. İleri ve geri hazırlık zamanları: $\tau_{A,B}$=3, $\tau_{A,C}$=4, $\tau_{B,C}$=5, $\tau_{B,A}$=2, $\mu_{C,A}$=2, $\mu_{B,A}$=1, $\mu_{B,C}$=2, $\mu_{C,B}$=1 olduğunu varsayalım. Tablo 1'de iki farklı şekilde sıralanan görevlerin toplam istasyon zamanları verilmiştir.

Tablo 1. A, B ve C görevlerinin iki farklı şekilde sıralanışı

Sıralama	Dikkate Alınacak Zamanlar	Toplam İstasyon Zamanı
A-B-C	15+3+12+5+10+2	47
B-A-C	12+2+15+4+10+1	44

Aynı istasyonda yer alan bu üç görevin iki farklı dizilişi ile aralarında 3 birimlik zaman farkı oluşmuştur. Yukarıdaki örnekte çevrim zamanının 45 olduğunu varsayalım. Bu durumda A-B-C sıralaması uygun bir çözüm olmaz; ancak B-A-C uygun bir çözüm olmaktadır. Dolayısıyla montaj hatları dengelenirken Andres vd.'ne [3] göre hem görevlerin istasyonlara atanması gerekir hem de atanan görevlerin istasyon içinde de çizelgelenmesi gerekmektedir.

Diferansiyel Gelişim Algoritması (DGA)

Diferansiyel Gelişim Algoritması (DGA), Storn ve Price [13] tarafından geliştirilen popülasyon temelli sezgisel bir eniyileme yöntemidir. DGA, her ne kadar sürekli ve küresel eniyileme problemlerinin çözümü için ortaya çıkmış olsa da Nearchou [32] tarafından ilk defa permütasyonel ve kesikli olan makina yerleşim probleminde başarıyla uygulanmıştır.

DGA işleyiş yapısı ve kullandığı operatörler bakımından Genetik Algoritmaya (GA) benzemektedir. Aralarındaki temel farklardan biri, GA arama işlemini ağırlıklı olarak

çaprazlama işlemi üzerinden yaparken, DGA ise arama işlemini daha çok mutasyon işlemi üzerinden yapmaktadır. Dolayısıyla DGA'da iyi bir arama yapabilmek için mutasyon işleminin etkin çalışması gerekir.

DGA diğer meta-sezgisellerden farklı olarak seçilim aşamasında en iyi bireylerin yaşamasına izin verir. Ayrıca her bir iterasyonda yeni nesiller oluşturmak yerine, ilk nesilde oluşturulan bireyler iterasyonlar boyunca varlıklarına devam etmekte ancak kendinden daha iyi bireyler oluştuğu zaman onunla yer değiştirirler. Böylece mevcut durumda kötü bir sonuç verse bile, küresel en iyiye ulaşabilecek çözümlerin devamı ve bireylerin çözüm uzayında dağıtıklığı sağlanarak erken yakınsamanın önüne geçmektedir.

Klasik DGA, temel olarak başlangıç popülasyonunun oluşturulması, mutasyon, çaprazlama ve seçilim olmak üzere dört temel aşamadan oluşmaktadır [13].

Başlangıç Popülasyonunun Oluşturulması

DGA'nın ilk aşamasında N_p adet D boyutlu parametreleri gerçel sayı olan vektör $x_{i,G}$ oluşturulur [13].

$$X_{i,G}\ ,\ \ i = 1,2,3,\dots\dots,N_p$$

$$G = 0,1,2,\dots\dots,G_{max} \qquad (1)$$

$$S = (X_{1G}, X_{2G}, X_{3G},\dots\dots,X_{N_pG})$$

Burada *i*, popülasyon indisini; G, nesil sayısını ve S ise başlangıç popülasyonunu göstermektedir.

Mutasyon

Her bir hedef vektör $X_{i,G}\ ,\ \ i = 1,2,3,\dots\dots,N_p$ için bir mutant vektör v_{iG+1} aşağıdaki gibi oluşturulur [13].

$$v_{iG+1} = x_{r1,G} + F(x_{r2,G} - x_{r3,G}) \qquad (2)$$

Burada $r_1, r_2, r_3 \epsilon (1,2,3,\dots\dots,N_p)$ birbirinden farklı tamsayılardır. Burada r_1, r_2, r_3 tamsayıları hedef vektör olan $X_{i,G}$'nin i indisinden farklı olduğu için popülasyon sayısı N_p, dört veya daha büyük olması gerekir. $F\epsilon[0,2]$ ise $(x_{r2,G} - x_{r3,G})$ vektörlerin farkını artıran veya azaltan kullanıcı tarafından belirlenen gerçel bir katsayıdır.

Çaprazlama

Mutasyona uğramış vektörlerin çeşitlendirmesini artırmak için çaprazlama operatörü kullanılır. Bu işlem sonunda aday vektör $u_{k,iG+1}$ aşağıdaki gibi oluşturulur [13].

$$u_{k,iG+1} = \begin{cases} v_{k,iG+1}\ eğer\ rasg \le CR\ ya\ da \\ x_{k,iG}\ eğer\ rasg > CR \end{cases} \qquad (3)$$

$$k = RasgTams(1,D), i \in [1,N_p]$$

Burada rasg, [0,1] aralığında eş olasılıkla üretilmiş bir gerçel sayı;, $CR \in [0,1]$, kullanıcı tarafından belirlenen çaprazlama olasılığı ve RasgTams(1,D) ise [1,D] aralığında rasgele tamsayı üreten bir fonksiyondur. Bu fonksiyon en azından bir parametrenin mutant vektörden alınmasını garantilemektedir.

Seçilim

Seçilim operatörünün amacı bir sonraki nesil olan G+1 neslini oluşturmaktır. Bir sonraki nesile aktarılan vektöre x_{iG+1} aşağıda denklem (4)' teki gibi karar verilir [13].

$$x_{iG+1} = \begin{cases} x_{iG} & eğer\ uyg(x_{iG}) \le uyg(u_{iG+1}) \\ u_{iG+1} & aksi\ takdirde \\ & i \in [1,N_p] \end{cases} \qquad (4)$$

Seçilim işleminde hedef ve aday çözüm vektörlerinin uygunluk değerlerine bakılır hangisi daha küçükse o vektör bir sonraki nesle aktarılır.

Geliştirilen Diferansiyel Gelişim Algoritması (GDGA)

Bu bölümde sıra-bağımlı hazırlık zamanlı GMHDP için geliştirilen DGA açıklanmıştır. GDGA 6 bölümden oluşmaktadır. Bunlar:

1. Başlangıç popülasyonun oluşturulması
2. Mutasyon
3. Çaprazlama
4. Tamir Operatörü

5. Uygunluk Fonksiyonu
6. Seçilim

Başlangıç Populasyonunun Oluşturulması

Sıra-bağımlı hazırlık zamanlı GMHDP, kesikli ve permütasyonel olduğu için GDGA'da çözüm vektörleri gerçel sayılardan değil kesikli sayılardan oluşmaktadır. Herbir çözüm vektörü D (görev sayısı) büyüklüğünde vektörden oluşmaktadır. Ayrıca başlangıç popülasyonu sayısı N_p kadar olmaktadır. Burada populasyon sayısı N_p kullanıcı tarafından belirlenmektedir. N_p sayısı büyüdükçe modelin en iyi çözümü bulma yeteneği artmaktadır. Ancak aynı zamanda çözüm bulma süresi de artmaktadır. Başlangıç popülasyonundaki her bir çözüm vektörü, 1'den D'ye kadar ve birbirinden farklı tamsayılardan oluşan D büyüklüğünde bir vektördür. Örneğin, montaj hattının 8 görevden oluştuğunu ve popülasyon sayısının 20 olarak belirlendiğini varsayalım. Popülasyon sayısı 20 olduğu için başlangıç popülasyonu 20 adet 8 elemanlı elemanları birbirinden farklı çözüm vektörlerinden oluşur.

Mutasyon

Sıra-bağımlı hazırlık zamanlı GMHDP'ye klasik DGA mutasyon operatörü uygulanamaz. Çünkü GDGA'da klasik DGA'nın aksine başlangıç populasyondaki çözüm vektörleri sürekli değil kesikli sayılardan oluşmaktadır. Dolayısıyla GDGA'da mutasyon operatörü için yeni bir bakış açısı gerekmekteydi. Bu yüzden bu çalışmada mutasyon operasyonu için yer değiştirme (swap) işlemi uygulanmıştır.

Bu çalışmadaki mutasyon operasyonu için geliştirilen yer değiştirme (swap) algoritması için üç birbirinden farklı vektör S_{1i}, S_{2i}, S_{3i} seçilmektedir. Burada klasik DGA'daki gibi iki vektörün birbirinden farkı bulunmamaktadır. Vektörlerin birbirinden farkının bulunmamasının sebebi, negatif değerlerin çıkmasını engellemektir. Çünkü negatif sonuçların çıkması çözüm vektörünü uygunsuz bir çözüm yapar. Örnek olarak, Tablo 2'de verilen vektörlerin $S_{2i} - S_{3i}$ işleminin sonucu [0 1 0 -1 0 0 2 -2] çıkacaktır. Negatif bir görev sayısı olmadığı için bu işlem sonucu uygunsuz (infeasible) olacaktır. Bu yüzden GDGA'da mutasyon işlemi için bir yer değiştirme prosedürü uygulanmıştır. Bu prosedüre göre seçilen üç vektörden ikisinin her bir elemanın birbirine eşit olup olmadığına bakılır. Aynı ise 0, değilse 1 değeri verilir (denklem 5). Ayrıca GDGA'da F katsayısı kullanılmamaktadır. Sadece değerlerin birbirine eşit olup olmadığına bakıldığı için F katsayısının kullanımı gereksiz olmaktadır.

$$x_i \;\; x_i = S_{2i} - S_{3i} \quad \forall i \in D$$
$$Y_i \;\; Y_i = \begin{cases} if(x_i == 0 & 0 \\ aksi\ takdirde & 1 \end{cases} \qquad (5)$$

Daha sonra vektörler üzerinde aşağıda Tablo 2'de gösterilen yer değiştirme işlemi uygulanır.

Tablo 2. Yer değiştirme (swap) işlemi

				S_{1i}				
İndis	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Değer	1	2	3	4	5	6	7	8
				S_{2i}				
İndis	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Değer	1	3	4	2	5	7	8	6
				S_{3i}				
İndis	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Değer	1	2	4	3	5	7	6	8
				Y_i				
İndis	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Değer	0	1	0	1	0	1	0	1

Tablo 2'de görüldüğü üzere Y_i vektörü S_{2i} ve S_{3i} vektörlerinin elemanlarının karşılaştırılmasıyla 0 ve 1 sayılarından oluşmaktadır. S_{2i} ve S_{3i} 'ün 1., 3., 5. ve 7. elemanları aynı görevlerden oluştuğu için Y_i vektöründe bunlara karşılık 0 değeri verilmiştir. Y_i vektörü oluşturulduktan sonra S_{1i} vektörü mutasyon işlemine tabi tutulacaktır. Buradaki

mutasyon işlemi ise klasik DGA'daki mutasyon işleminden farklıdır. Buradaki mutasyon işleminde Y_i vektöründeki 0 olan indislerin S_{1i} vektöründeki karşılıkları ardışık olarak yer değiştirirler. Örneğin, 1. ve 3. indisler Y_i vektöründe 0 olduğu için S_{1i} vektöründe 1. ve 3. indisler yer değiştirirler. Aynı şekilde 5. ve 7. indislerindeki görevler de karşılıklı yer değiştirirler. Oluşan yeni mutant vektör aşağıda Tablo 3'deki gibidir.

İndis	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Değer	3	2	1	4	6	7	5	8

Tablo 3. Mutant vektör(v_{iG+1})

Çaprazlama

Çaprazlama işlemi klasik DGA'daki gibi yapılır. Aday vektör $u_{k,iG+1}$ aşağıdaki gibi oluşturulmaktadır.

$$u_{k,iG+1}=\begin{cases} v_{k,iG+1}\ eğer\ rasg \le CR\ ya\ da\ k = RasgTams(1,D) \\ x_{k,iG}\ eğer\ rasg > CR \end{cases} \quad (6)$$

$k \in [1, D], i \in [1, N_p]$

Burada rasg, [0,1] aralığında eş olasılıkla üretilmiş bir gerçel sayı; $CR \in [0,1]$, kullanıcı tarafından belirlenen çaprazlama olasılığı ve RasgTams (1,D) ise [1,D] aralığında rasgele tamsayı üreten bir fonksiyondur. Bu fonksiyon en azından bir parametrenin mutant vektörden alınmasını garantilemektedir.

Tablo 4. Çaprazlama işlemi

Rastgele Değer	0,3	0,7	0,1	0,8	0,9	0,4	0,45	0,6
Mutant Vektör ($v_{k,iG+1}$)	3	2	1	4	6	7	5	8
Hedef Vektör ($x_{k,iG}$)	1	2	3	4	5	7	6	8
Çaprazlama Sonucu Oluşan Vektör ($u_{k,iG+1}$)	3	2	1	4	5	7	5	8

Yukarıdaki örnekte CR (çaprazlama oranı) değeri 0,5 olarak alınmıştır. Bu değerin altında çıkan indislerin değerleri mutant vektörden diğerleri hedef vektördeki değerlerden alınmıştır. Aynı şekilde bu çalışmadaki GDGA'da da CR değeri 0,5 olarak alınmıştır. Çaprazlama operatürünün sözde kod yapısı aşağıda Algoritma 1'de verildiği gibidir.

Algoritma 1. Çaprazlama operatörü sözde kod yapısı

Başla
for (k=1; k≤D; k++)
rasg, [0,1] aralığında eş olasılıkla üretilmiş bir gerçel sayı
  if($rasg \le$CR)
    $v_{k,iG+1}$ görevinin u_{iG+1} vektöründe olup olmadığını kontrol et, varsa (c=1) yoksa (c=0)
    if(c==0)
      $u_{k,iG+1} = v_{k,iG+1}$
    endif
  else
    $x_{k,iG}$ görevinin u_{iG+1} vektöründe olduğunu kontrol et, varsa (c=1) yoksa (c=0)
    if(c==0)
      $u_{k,iG+1} = x_{k,iG}$

endif
endif
endfor
Boş kalan indisleri belirle
Kalan görevleri küçükten büyüğe doğru boş kalan indislere ata.
Bitir

Tamir Operatörü

Başlangıç popülasyonunu oluşturan çözüm vektörleri rastgele oluşturulduğu için, bazı vektörler öncül-ardıl ilişkisine uymayabilir. Bu nedenle başlangıç popülasyonunun her bir elemanı için tamir operatörü uygulanır. Ayrıca, mutasyon operatöründe rastgele işlemlere maruz kalan vektörler çaprazlama sonucunda aday vektör olarak seçilebilirler. Dolayısıyla, aday vektörlerin arasında da uygunsuz çözümler çıkabilir. Bu nedenle, aday vektörler de bir tamir sürecinden geçmektedirler. Tamir algoritmasının sözde kod yapısı aşağıda Algoritma 2'de verildiği gibidir.

Algoritma 2. Tamir algoritması sözde kod yapısı
Başla
for (m=1; m≤D; m++)
k=m
while (öncül-ardıl ilişkileri sağlanıncaya kadar)
do
$X_{k,iG}$ görevi ile kendisinden sonra gelen görevlerin öncül ardıl ilişkisini kontrol et, sağlıyorsa $(t_k = 1)$ veya sağlamıyorsa $(t_k = 0)$
if $(t_k == 0)$
Bir sonraki görevi seç, $(k = k + 1)$
else
$X_{k,iG}$ görevi ile $X_{m,iG}$ görevinin yerini değiştir, $X_{k,iG} = X_{m,iG}$
endif
endwhile
endfor
Bitir

Görevlerin İstasyonlara Atanması: Uygunluk Değerinin Hesaplanması

Çözüm vektörlerinin uygunluk değerlerinin hesaplanması için, görevlerin istasyonlara atanması gerekir. Bu çalışmadaki GDGA'nın amacı verilen GMHDP için istasyon sayısının minimize edilmesidir. Dolayısıyla çalışmanın uygunluk değeri istasyon sayısıdır. Tamir operasyonundan çıkan hedef ve aday vektörlerdeki görevler, vektörün fenotipindeki sırayla tek tek istasyonlara atanırlar. Bu atamalar aşağıda Denklem 7 [7] kullanılarak görev ve hazırlık sürelerinin toplamına bakılır. Eğer bu süre çevrim zamanını geçerse o istasyon kapanır ve yeni bir istasyon açılır. Böylece çözüm vektöründeki bütün görevler istasyonlara atanıncaya kadar bu işlem devam eder. En sonunda açılan istasyon sayısı çözüm vektörünün uygunluk değeri olur.

$$t_i + \tau_{i,j} + t_j + \tau_{j,h} + t_h + \mu_{h,i} \leq c \qquad (7)$$

Denklem 7'deki t_i: i. görevin işlem zamanı, $\tau_{i,j}$: i ve j görevleri arasındaki ileri hazırlık zamanı, $\mu_{h,i}$: h ve i görevleri arasındaki geri hazırlık zamanı ve c: Çevrim zamanını göstermektedir.

Seçilim

Klasik DGA'da seçilim prosedürü denklem 8'de verilmiştir:

$$x_{iG+1} = \begin{cases} x_{iG} & eğer\ f(x_{iG}) \leq f(u_{iG+1}) \\ u_{iG+1} & aksi\ takdirde \end{cases} \quad i \in [1, N_p] \qquad (8)$$

Seçilim işleminde hedef ve aday çözüm vektörlerinin uygunluk değerlerine bakılır hangisi daha küçükse o vektör bir sonraki nesle aktarılır. Bu çalışmadaki GDGA'nın seçilim işlemi, klasik DGA'nın seçilim işlemiyle aynıdır.

GDGA, tümüyle aşağıda Tablo 5'deki gibi özetlenebilir.

Tablo 5. Geliştirilmiş Diferansiyel Gelişim Algoritması prosedürü

Başla

N_p sayısı kadar rastgele hedef vektör *i* oluştur (*i*=1…N_p)

while(maksimum jenerasyon sayısına ulaşıncaya kadar)

for (i=1; i≤N_p;i++)

do

Algoritma 2'de verilen tamir işlemlerini hedef vektör *i* üzerinde uygula

Vektör *i*'ye mutasyon işlemlerini uygula

Vektör *i*'ye Algoritma 1'de verilen çaprazlama işlemlerini uygula

Aday vektör *i*'ye Algoritma 2' de verilen tamir işlemlerini uygula

Hedef vektör *i* ve aday vektör *i* üzerinde verilen Denklem 8'de verilen seçilim işlemlerini uygula

endfor

endwhile

Bitir

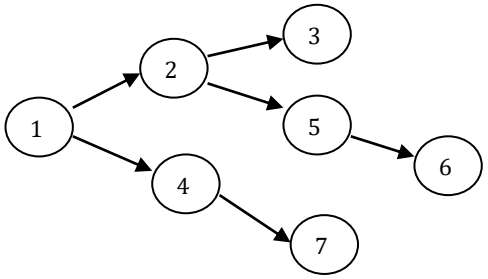
Geliştirilen Diferansiyel Gelişim Algoritması İçin Sayısal Örnek

Bu bölümde GDGA'nın nasıl çalıştığını daha iyi anlayabilmek için literatür problemlerinden Mertens'in 7 görevli ve çevrim zamanı 18 birim olan problemi ele alınacaktır. Problemde verilen görevlerin işlem zamanları Tablo 6'daki gibidir.

Tablo 6. Mertens örneği işlem zamanları

Görevler	**1**	**2**	**3**	**4**	**5**	**6**	**7**
İşlem Zamanları	1	5	4	3	5	6	5

Mertens örneğinin öncül-ardıl ilişkileri Şekil 1'deki gibidir.



Şekil 1. Mertens örneği öncelik diyagramı

Sıra-bağımlı ve hazırlık zamanlı GMHDP'de görevler arasında hem ileri, hem de geri hazırlık zamanları olmaktadır. Mertens örneğinin Scholl vd. [7] tarafından eklenen ileri ve geri hazırlık zamanları aşağıda Tablo 7 ve Tablo 8'deki gibidir

Tablo 7. Mertens örneği ileri hazırlık zamanları

	1	**2**	**3**	**4**	**5**	**6**	**7**
1	0	1	0	0	0	0	0
2	0	0	2	1	1	0	1
3	0	0	0	1	1	0	1
4	0	1	1	0	0	1	0
5	0	0	1	0	0	1	0
6	0	0	0	1	0	0	1
7	0	1	1	0	0	1	0

Tablo 8. Mertens örneği geri hazırlık zamanları

	1	**2**	**3**	**4**	**5**	**6**	**7**
1	3	0	0	0	0	0	0
2	4	3	0	4	0	0	4
3	2	1	3	2	2	3	2
4	3	2	4	3	3	4	0
5	3	2	4	3	3	0	3
6	2	1	3	2	2	3	2
7	3	2	4	3	3	4	3

GDGA'nın ilk aşamasında başlangıç popülasyonu N_p oluşturulur. Bu örnekte başlangıç popülasyonunun kullanıcı tarafından N_p=8 olarak belirlendiğini varsayalım. Görev sayısı D=7 olduğu için, 8 adet 7 elemanlı çözüm vektörleri Tablo 9'daki gibi rassal oluşturulmuştur.

Tablo 9. Mertens örneği başlangıç populasyonu

X_{11}	X_{21}	X_{31}	X_{41}	X_{51}	X_{61}	X_{71}	X_{81}
7	5	5	2	3	2	1	6
6	4	1	1	2	5	3	4
3	1	4	6	6	6	5	5
1	7	7	4	5	7	6	7
4	6	3	5	4	3	7	2
5	3	2	7	7	4	4	3
2	2	6	3	1	1	2	1

İlk nesil oluşturulduktan sonra her bir çözüm vektörü için aşağıda yapılan işlemler teker teker yapılır. Bu örnekte bu işlemler sadece bir çözüm vektörü üzerinde gösterilecektir. Populasyonun ilk çözüm vektörü X_{11} olduğu için hedef vektör de X_{11} olacaktır. GDGA'nın ilk aşamasında X_{11} hedef vektörü uygun(feasible) bir çözüm olmadığı için üzerinde tamir operasyonu olan Algoritma 2 uygulanacaktır. Tamir aşamalarından geçen X_{11} hedef vektörü Tablo 10'daki gibi son halini alacaktır.

Tablo 10. X_{11} hedef vektörü tamir işlemi

X_{11} Hedef Vektörü	1.İşlem	2.İşlem	3. İşlem	4.İşlem	5.İşlem	X_{11} Hedef Vektörünün Son Hali
7	1	1	1	1	1	1
6	6	4	4	4	4	4
3	3	3	7	7	7	7
1	7	7	3	2	2	2
4	4	6	6	6	5	5
5	5	5	5	5	6	6
2	2	2	2	3	3	3

Hedef vektöre tamir işlemi uygulandıktan sonra mutasyon aşamasına geçilmektedir. Mutasyon işlemi için hedef vektörden ve birbirinden farklı 3 çözüm vektörü rassal seçilmektedir. Bu seçilen vektörlerin $S_{11} = X_{21}, S_{21} = X_{51}, S_{31}=X_{61}$ olduğunu varsayalım. Burada öncelikle denklem 5'te verilen işlemler uygulanır ve yer değiştirme(swap) noktalarını veren Y_i vektörü Tablo 11'deki gibi oluşturulur.

Tablo 11. Y_i yer değiştirme vektörü

X_{51}	X_{61}	Y_i
3	2	1
2	5	1
6	6	0
5	7	1
4	3	1
7	4	1
1	1	0

Yer değiştirme Y_i vektörü oluşturulduktan sonra $S_{11} = X_{21}$ vektöründeki Y_i vektörünün 0'a karşılık gelen indisleri yerdeğiştirilir. Bu durumda X_{21} vektörünün 3. ve 7. indisleri yer değiştirilir. Böylece mutant v_{12} vektörü Tablo 12'deki gibi oluşturulur.

Tablo 12. Mutasyon işlemi

				X_{21}			
İndis	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Değer	5	4	1	7	6	3	2
				⇩			
				v_{12}			
İndis	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Değer	5	4	2	7	6	3	1

Mutasyon işleminden sonra hedef vektör üzerinde çaprazlama işlemi uygulanır. Aday vektör u_{12} Algoritma 1'deki gibi oluşturulup Tablo13'te verilmiştir.

Tablo 13. Çaprazlama işlemi

Rastgele Değer	**0,6**	**0,5**	**0,1**	**0,8**	**0,9**	**0,4**	**0,6**
Mutant Vektör (v_{12})	5	4	2	7	6	3	1
Hedef Vektör (x_{11})	1	4	7	2	5	6	3
Çaprazlama Sonucu Oluşan Aday Vektör (u_{12})	1	4	2	-	6	3	-

Bu örnekte CR(çaprazlama oranı) değeri 0,5 alınmıştır. Aday vektörün 4. ve 7. indislerine gelen görevler daha önce vektörde bulunduğu için kalan 5 ve 7 görevleri kalan boş yerlere küçükten büyüğe doğru atanırlar. Çaprazlama sonucu oluşan aday vektörün son hali u_{12} Tablo 14'teki gibidir.

Tablo 14. Aday vektör u_{12} son hali

	u_{12}						
İndis	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Değer	1	4	2	5	6	3	7

Çaprazlama sonucu oluşan aday vektör öncül-ardıl ilişkisine uyduğu için tekrar bir tamir işlemi uygulamaya gerek kalmamıştır. GDGA'nın son aşamasında seçilim işlemi bulunmaktadır. Denklem 8'de verilen seçilim işlemine göre hedef vektör ve aday vektörün uygunlukları karşılaştırılacaktır. Bu çalışma bir minimizasyon problemi olduğu için uygunluk fonksiyonu küçük olan; yani istasyon sayısı küçük olan vektör bir sonraki nesle aktarılacaktır. Uygunluk fonksiyonunun hesaplanması için öncelikle görevlerin istasyonlara atanması gerekir. Görevlerin istasyonlara atanması için Denklem 7 kullanılmıştır. Hedef vektör X_{11} ve aday vektörün u_{12} görevlerinin istasyon ataması Tablo 15 ve Tablo 16'daki gibidir:

Tablo 15. Hedef vektör X_{11} istasyon ataması

İstasyon 1			İstasyon 2		İstasyon 3	
1	4	7	2	5	6	3

Tablo 16. Aday vektör u_{12} istasyon ataması

İstasyon 1				İstasyon 2		İstasyon 3
1	4	2	5	6	3	7

Burada her bir istasyondaki görevlerin ve ileri ve geri hazırlık zamanlarının toplamı çevrim süresi olan 18 birimi geçmemesi gerekir. Örneğin hedef vektörün X_{11} İstasyon 1'i şu şekilde hesaplanmıştır:

$t_1 + \tau_{1,4} + t_4 + \tau_{4,7} + t_7 + \mu_{7,1} \leq 18$

$\rightarrow$1+0+3+0+5+3$\leq$ 18 $\rightarrow$ 12$\leq$ 18

Diğer istasyonlar da aynı şekilde hesaplanmıştır. Seçilim işlemi için hem hedef vektörün hem de aday vektörün uygunluklarına bakılır. Uygunluk değeri vektörlerin istasyon sayılarına eşittir. Hedef vektörün uygunluk değeri $f(x_{11})$=3 ve aday vektörün uygunluk değeri $f(u_{12}) = 3$. Denklem 8'deki seçilim işlemine göre uygunluk değerleri birbirine eşitse bir sonraki nesle hedef vektör X_{11} taşınır. Dolayısıyla X_{12}= X_{11} olacaktır.

Deneysel Sonuçlar

GDGA, MATLAB programında kodlanmış ve test problemleri Intel Core i5, 2.4 GHz, 6 GB RAM özelliklerine sahip PC kullanılarak test edilmiştir. Test problemleri http://www.assembly-line-balancing.de adresinden alınmıştır. Literatürde geçen ve en iyi (optimum) çözümü bilinen BMHDP'ye Scholl vd. [7] tarafından ileri (forward) ve geri (backward) hazırlık zamanları eklenerek sıra-bağımlı hazırlık zamanlı GMHDP için 269 adet test problemi oluşturulmuştur Bu çalışmada, deneysel çalışma için bu test problemleri kullanılacaktır. GDGA'nın parametreleri olan başlangıç populasyonu N_p ve çaprazlama oranı CR'nin farklı değerleri için GDGA çalıştırılmıştır. Bulunan sonuçlara göre N_p =50 ve CR=0,5 değerlerinde algoritmanın makul zamanda en iyi sonuçlar verdiği gözlenmiştir. Yolmeh ve Kianfar [5] ve Pitakaso ve Sethanan [25], test problemlerini, geliştirdikleri algoritma ile 5 kez çözmüş ve buldukları en iyi sonuçları literatürdeki diğer sonuçlarla karşılaştırmışlardır. Bu çalışmada da benzer şekilde her bir test problemi için GDGA, 5 kez çalıştırılmış ve bulunan en iyi sonuçlar kaydedilmiştir. Toplam 269 adet test problemi olduğu için toplamda 1345 adet deney yapılmıştır.

Scholl vd. [7]., hazırlık zamanlarını oluşturmak için $\alpha.t_{ort}$ sayısını üst sınır seçmiş ve rassal ileri ve geri hazırlık zamanları oluşturmuşlardır. Burada t_{ort}, ortalama işlem zamanını temsil etmekte ve α ise 0,25, 0,50, 0,75, 1 sayılarından birini almaktadır. Bu durumda ileri ve geri hazırlık zamanları ortalama işlem süresinin 12,5, 25, 37,5, 50%'sine kadar değer alabilmektedir. Dolayısıyla 4 farklı α seviyesinde 4 farklı deney seti oluşturulmuştur. Bu çalışmada problem setlerinin en zor seviyesi

olan α=1 seviyesindeki problem seti kullanılarak GDGA çalıştırılmıştır.

Test problemleri daha önce de söylendiği gibi literatürde çalışılan BMHDP'ye, Scholl vd. tarafından ileri ve geri hazırlık zamanları eklenerek oluşturulmuştur. BMHDP'nin en iyi çözümleri bilinmesine rağmen, oluşturulan bu test problemleri çok daha karmaşık olduğu için en iyi çözümleri bilinmemektedir [7]. Dolayısıyla bu sorunun üstesinden gelebilmek için ve Andres vd. [3]., buldukları sonuçların performanslarını belirleyebilmek için bir alt sınır (lower bound) metodu kullanmışlardır. Andres vd. [3], tarafından kullanılan alt sınır metodu daha sonra Scholl vd. tarafından geliştirilmiştir [7]. Bu çalışmada da Scholl vd. tarafından geliştirilen alt sınır denklemi (9) kullanılmıştır.

$$LB = min\{m \geq LB1 \mid t_{top} + \tau_{top}^{n-m} + \mu_{top}^{m} \leq TC(m)\}$$
$$\text{LB1}=\left\lceil \frac{t_{top}}{c} \right\rceil \quad , \quad \text{TC(m)=m.c} \tag{9}$$

Denklem 9'daki LB: İstasyon sayısının alt sınırı, t_{top}: Bütün görevlerin toplam işlem zamanı, n: Toplam görev sayısı, m: Bilinen eniyi istasyon sayısı, τ_{top}^{n-m}: n-m sayıdaki görevin arasındaki toplam ileri (forward) hazırlık zamanı, μ_{top}^{m}: m sayıdaki görevin arasındaki toplam geri (backward) hazırlık zamanı ve c: Çevrim zamanını göstermektedir.

Bu çalışmadaki GDGA, Scholl vd. tarafından sıra-bağımlı hazırlık zamanlı GMHDP çözümü için geliştirilen ve en iyi sonuçları bulan 4 farklı sezgisel olan FBRI10, FBCRI10, FBCRI100 ve FBLS10 ile karşılaştırılmıştır [7]. Bulunan sonuçlar Tablo 17'de verilmiştir.

Tablo 17. GDGA ve diğer sezgisellerin karşılaştırması (α=1 veri seti için)

	FBRI 10	FBCRI 10	FBCRI 100	FBLS 10	GDGA
Rel.LB (%)	31,19	31,19	30,27	35,39	29,87
#Opt	8	8	8	4	20
CPU (sn)	1,86	1,86	18,85	6,32	54

Tablo 17'de verilen Rel.LB (%): Bulunan sonucun istasyon alt sınır sayısından sapma yüzdesi, #Opt: Bulunan en iyi çözüm sayısı, CPU (sn): Algoritmanın her bir problem için saniye cinsinden çalışma süresidir.

Tablo 17'ye bakıldığında bu çalışmadaki GDGA'nın diğer sezgisellere göre daha iyi sonuçlar verdiği görülmektedir. GDGA'nın 269 test problemi içinde bulduğu çözümlerin alt sınırdan sapma yüzdesi %29,87 ile diğer algoritmalardan daha iyi sonuçlar bulduğunu göstermektedir. Bununla birlikte diğer algoritmalar 269 test problemi içinde 8 adet en iyi sonuca ulaşmışken, GDGA 20 adet en iyi sonuca ulaşmıştır. Burada bulunan en iyi çözüm sayısının düşük olmasının nedeni bu problem türünün çok karmaşık olması ve problemlerin en iyi çözümleri bilinmediği için bulunan çözümler alt sınırlar ile karşılaştırılmasıdır. Dolayısıyla test problemlerinin en iyi çözümleri bilinseydi, algoritmanın bulduğu en iyi çözümlerin daha fazla olduğu ve alt sınırdan sapma yüzdesinin daha da azaldığı görülebilirdi. GDGA'nın her bir test problemi için bilgisayardaki ortalama çalışma süresi 54 saniye olmuştur ve diğer algoritmalardan daha yüksek çıkmıştır. Ancak bu süre algoritmanın bulduğu sonuçlar itibariyle için tolere edilebilir düzeydedir.

Sonuç ve Öneriler

Bu çalışmada sıra-bağımlı hazırlık zamanlı GMHDP tip-1'in çözümü için yeni bir DGA geliştirilmiştir. Bu çalışmada çözülmeye çalışılan sıra-bağımlı hazırlık zamanlı GMHDP, NP-zor olan BMHDP'den çok daha karmaşık yapıya sahiptir. Bu tür NP-zor problemler için doğrusal programlama, dinamik programlama ve dal-sınır algoritmaları gibi kesin yöntemlerle makul zamanlarda optimum çözümlerinin bulunması büyük problemler için imkansızdır. Dolayısıyla bu tür problemlerin çözümleri için literatürde sezgisel veya metasezgisel yöntemler çok sık kullanılmaktadır. Bu çalışmada önerilen GDGA, klasik DGA'dan farklı olarak kesikli bir yapıda olan sıra-bağımlı hazırlık zamanlı GMHDP'ye uyarlanmıştır. Sürekli problemlerin çözümünde çok iyi sonuçlar veren DGA, bu çalışmada da çok iyi sonuçlar vermiştir.

Bu çalışmadaki sonuçlar; literatürde daha önce geliştirilen sezgisellerden çözüm kalitesi bakımından daha iyi sonuçlar vermiştir, ancak süre açısından daha iyi sonuçlar verememiştir. Ama istenilen, makul zamanda makul çözümler

olduğu için GDGA'nın bu isteği fazlasıyla karşıladığı söylenebilir.

Bu çalışmada GDGA'nın parametrelerinin eniyilenmesi için kapsamlı bir deney tasarımı yapılmamıştır. İleriki çalışmalarda parametre eniyilenmesi için bir deney tasarımı yapılabilir. Ayrıca, sıra-bağımlı hazırlık zamanlı GMHDP'nin daha karmaşık ve gerçekçi hali olan sıra-bağımlı hazırlık zamanlı U-tipi, çift taraflı veya paralel istasyonlu gibi çeşitleri için de DGA'lar veya başka metasezgiseller geliştirilebilir.

Kaynaklar

[1] Ağpak, K., Gökçen, H., Saray, N.N. Özel, S., (2002). Stokastik Görev Zamanlı Tek Modelli U Tipi Montaj Hattı Dengeleme Problemleri İçin Bir Sezgisel, *Gazi Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi*, **17**, 4, 115-124.

[2] Becker, C., Scholl, A., (2006). A survey on problems and methods in generalized assembly line balancing, *European Journal of Operational Research*, **168**, 3, 694-715.

[3] Andres, C., Miralles, C., Pastor, R., (2008). Balancing and Scheduling Tasks in Assembly Lines with Sequence-Dependent Setup Times, *European Journal of Operational Research*, **187**, 3, 1212-1223.

[4] Özcan, U., Toklu, B.2010. Balancing Two-Sided Assembly Lines with Sequence-Dependent Setup Times, *International Journal of Production Research*, **48**, 18, 5363-5383.

[5] Yolmeh, A., Kianfar, F., (2011). An Efficient Hybrid Genetic Algorithm to Solve Assembly Line Balancing Problem with Sequence-Dependent Setup Times, *Computers & Industrial Engineering*, **62**, 4, 936–945.

[6] Seyed-Alagheband, S., Ghomi, S.M.T.F., Zandieh, M., (2011). A simulated annealing algorithm for balancing theassembly line type II problem with sequence-dependent setup times between tasks, *International Journal of Production Research*, **49**, 805-825.

[7] Scholl, A., Boysen, N., Fliedner, M., (2013). The Assembly Line Balancing and Scheduling Problem with Sequence-Dependent Setup Times: Problem Extension, Model Formulation and Efficient Heuristics, *OR Spectrum*, **35**, 1, 291-321.

[8] Hamta, N., Ghomi, S.M.T.F., Jolai, F., Shirazi, M. A., (2013). A Hybrid PSO Algorithm for a Multi-Objective Assembly Line Balancing Problem with Flexible Operation Times, Sequence-Dependent Setup Times and Learning Effect, *International Journal of Production Economics*, **141**, 1, 99-111.

[9] Akpınar, Ş., Bayhan, G.M., Baykasoğlu, A., (2013). Hybridizing Ant Colony Optimization via Genetic Algorithm for Mixed-Model Assembly Line Balancing Problem with Sequence Dependent Setup Times between Tasks, *Applied Soft Computing*, **13**, 1, 574-589.

[10] Diri, Z., Mete, S., Çil, Z.A., Ağpak, K., (2015). Stokastik Sıra-Bağımlı Hazırlık Zamanlı Montaj Hattı Dengeleme Problemi, *Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi*, **21**, 4, 152-157.

[11] Janardhanan, M.N., Li, Z., Bocewicz, G., Banaszak, Z, Nielsen, P., (2018). Metaheuristic algorithms for balancing robotic assembly lines with sequence-dependent robot setup times, *Applied Mathematical Modelling*.

[12] Gutjahr, A.L., Nemhauser, G.L., (1964). An algorithm for the line balancing problem, *Management Science*, **11**, 2, 308–315.

[13] Storn, R., Price, K., (1997). Differential evolution-A simple and efficient heuristic for global optimization over continues spaces, *Journal Global Optimization*, **11**, 241–354.

[14] Cheng, S.L., Hwang, C., (2001). Optimal approximation of linear systems by a differential evolution algorithm, *IEEE Transactions on Systems, Man, and Cybernetics-Part A, Systems and Humans*, **31**, 6, 698–707.

[15] Ali, M.M., Torn, A., (2004). Population set-based global optimization algorithms: Some modifications and numerical studies, *Computers and Operations Research*, **31**, 1703–1725.

[16] Kaelo, P., Ali, M.M., (2005). A numerical study of some modified differential evolution algorithms, *European Journal of Operational Research*, **169**, 1176–84.

[17] Sun, J., Zhang, Q., Tsang. E. P. K., (2005). DE/EDA: A new evolutionary algorithm for global optimization, *Information Sciences*, 169, 3, 249–262.

[18] Montes, M. E., Miranda-Varela, M. E., del Carmen Gomez-Ramon, R., (2010). Differential evolution in constrained numerical optimization: An empirical study, *Information Sciences*, **180**, 22, 4223–4262.

[19] Nearchou, A. C., (2007). Balancing large assembly lines by a new heuristic based on differential evolution method, *International Journal of Advanced Manufacturing Technology*, **34**, 1016–1029.

[20] Nearchou, A. C. (2008). Multi-objective balancing of assembly lines by population heuristics, *International Journal of Production Research*, **46**, 8, 2275–2297.

[21] Kim, Y. K., Kim, Y. J., Kim. Y., (1996). Genetic algorithms for assembly line balancing with various objectives, *Computers Industrial Engineering*, **30**, 3, 397–409.

[22] Nourmohammadi, A., Zandieh, M., (2011). Assembly line balancing by a new multiobjective differential evolution algorithm based on TOPSIS, *International Journal of Production Research,* **49**, 10, 2833–2855.

[23] Mozdgir, A., Mahdavi, I., Seyedi Badeleh, I., Solimanpur, M., (2013). Using the Taguchi method to optimize the differential evolution algorithm parameters for minimizing the workload smoothness index in simple assembly line balancing, *Mathematical and Computer Modelling*, **57**, 1–2, 137–151.

[24] Vincent, L. W. H., Ponnambalam, S. G., (2013). A differential evolution-based algorithm to schedule flexible assembly lines, *IEEE Transactıons On Automatıon Scıence And Engıneerıng,* **10**, 4, 1161-1165.

[25] Pitakaso, P., Sethanan, K., (2015). Differential Modified differential evolution algorithm for simple assembly line balancing with a limit on the number of machine types, *Engineering Optimization*.

[26] Nilakantan, J. M., Nielsen, I., Ponnambalam, S. G., Venkataramanaiah, S., (2016). Differential evolution algorithm for solving RALB problem using cost- and time-based models, *The International Journal of Advanced Manufacturing Technology*, **89**, 1, 311-332.

[27] Zhang, H., Yan, Q., Liu, Y., Jiang, Z., (2016). An integer-coded differential evolution algorithm for simple assembly line balancing problem of type 2, *Assembly Automation*, **36**, 3.

[28] Nearchou, A.C., Omirou, S.L., (2017). Assembly Line Balancing Using Differential Evolution Models, *Cybernetics and Systems*.

[29] Gangsterer, M., Hartl, R. F., (2017). One- and two-sided assembly line balancing problems with real-world constraints, *International Journal of Production Research*, 3025-3042.

[30] Özçelik, F., (2018). Basit düz ve U-tipi montaj hattı dengeleme problemleri için diferansiyel evrim algoritması, *Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi*, **24**, 1.

[31] Becker, C., Scholl, A., (2009). Balancing assembly lines with variable parallel workplaces: Problem definition and effective solution procedure. European Journal of Operational Research, **199**, 359–374.

[32] Nearchou, A. C., (2006). Meta-heuristics from Nature for the Loop Layout Design Problem, *International Journal of Production Economics*, **101**, 2, 312–328.