T.C.
HARRAN ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

BEKLEMESİZ AKIŞ TİPİ ÇİZELGELEME PROBLEMLERİ İÇİN HİBRİT GRİ KURT OPTİMİZASYON ALGORİTMASI

Cengiz KINA

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

ŞANLIURFA
2022

İÇİNDEKİLER

Sayfa No

ÖZET i
ABSTRACT ii
TEŞEKKÜR iii
ŞEKİLLER DİZİNİ iv
ÇİZELGELER DİZİNİ v
SİMGELER ve KISALTMALAR DİZİNİ vi
1. GİRİŞ 1
2. ÖNCEKİ ÇALIŞMALAR 4
3. MATERYAL ve YÖNTEM 9
3.1 Beklemesiz Akış Tipi Çizelgeleme 9
3.2 Doğrusal Programlama Yöntemi 13
3.3 Gri Kurt Optimizasyon Algoritması 14
3.3.1 Sosyal hiyerarşi 15
3.3.2 Avı kuşatma 16
3.3.3 Avlanma 17
3.3.4 Ava saldırma 18
3.3.5 Av arama (keşif) 19
3.4 Yerel Arama Algoritması 20
3.5 Hibrit Gri Kurt Optimizasyon Algoritması 23
3.5.1 Başlangıç popülasyonu 24
3.5.2 NEH algoritması 25
3.5.3 Sıralı İşlem Süresi (SİS) 26
3.6 Karşılaştırılan Algoritmalar 28
3.6.1 Uyarlanabilir öğrenme yaklaşımı 28
3.6.2 Genetik sezgisel algoritma 28
3.6.3 Karınca koloni optimizasyon algoritması 28
3.7 Kullanılan Problem Seti 29
4. ARAŞTIRMA BULGULARI ve TARTIŞMA 31
5. SONUÇLAR ve ÖNERİLER 46
KAYNAKLAR 47

ÖZET

Yüksek Lisans Tezi

BEKLEMESİZ AKIŞ TİPİ ÇİZELGELEME PROBLEMLERİ İÇİN HİBRİT GRİ KURT OPTİMİZASYON ALGORİTMASI

Cengiz KINA

Harran Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman : Doç. Dr. İbrahim Berkan AYDİLEK
Yıl: 2022, Sayfa: 50

Günümüzde giderek artan rekabet ortamında siparişlerin zamanında teslim edilmesi ve kaynakların verimli kullanılması ihtiyacı, üretim çizelgelemeyi önemli bir hale gelmiştir. Akış tipi çizelgeleme problemleri gerçek yaşam uygulamalarında yer edinen önemli çizelgeleme türlerinden biridir. Beklemesiz akış tipi çizelgeleme (BATÇ) ilaç sanayi, plastik sanayi, demir çelik sanayi gibi alanlarda bir makinede işlenen bir ürünün bir sonraki işlenecek makineye geçtiğinde makineler arasında herhangi bir beklemenin müsaade edilmediği bir çizelgeleme türüdür. İşler makineler arasında beklediğinde yapısında bozulmalar meydana gelebileceği gibi üretim sürecini olumsuz etkileyip maliyetleri arttırabilmektedir. Makine sayısı iki veya daha az olan problemlerde en iyi çözümü bulmak mümkündür. Ancak makine sayısının fazla olduğu problemlerde en iyi çizelgelemeyi bulmak imkânsız hale gelebilir. Bu tür problemlere literatürde NP-Zor problemler denilmektedir. Kesin yöntemler bu tür problemlerin çözümünde yetersiz kaldığı için metasezgisel yöntemler tercih edilmektedir. Metasezgisel yöntemler kesin sonucu garanti etmemekle birlikte kısa bir süre içirisinde optimum çözüme yakın çözümler sunabildiği için literatürde sıklıkla kullanılmaktadır. Bu çalışmada BATÇ problemlerinin çözümü için birçok çizelgeleme probleminde kullanılan Gri kurt optimizasyon (GKO) algoritması kullanılmıştır. Tez çalışmasında GKO algoritması literatürde iyi bilinen ve BTAÇ için üretilmiş 192 test problemi üzerinde test edilmiştir. Algoritmanın performansını arttırmak amacıyla başlangıç popülasyonuna Nawaz, Enscore, Ham (NEH) ve Sıralı İşlem Süresi (SİS) yöntemlerine göre üretilen çözümler eklenmiştir. Ayrıca algoritmanın sömürü yeteneğini arttırmak için yerel arama algoritması olarak Değişken Komşuluk Arama algoritması ile hibrit hale getirilmiştir. Elde edilen sonuçlar önerilen algoritmanın BATÇ problemleri için oldukça etkili ve verimli olduğunu ortaya koymuştur.

ANAHTAR KELİMELER: Beklemesiz Akış Tipi Çizelgeleme, Gri Kurt Optimizasyon Algoritması, Maksimum Tamamlanma Zamanı, Yerel Arama

ABSTRACT

MSc Thesis

HYBRID GRAY WOLF OPTIMIZATION ALGORITHM FOR NO-WAIT FLOW SHOP SCHEDULING PROBLEMS

Cengiz KINA

Harran University
Graduate School of Natural and Applied Sciences
Department of Computer Engineering

Supervisor : Assoc. Prof. Dr. İbrahim Berkan AYDİLEK
Year: 2022 , Page: 50

These days, in an increasingly competitive environment, the need for timely delivery of orders and efficient use of resources has made production scheduling important. Flow scheduling problems are one of the important scheduling types in real life applications. No-wait flow shop scheduling (NWFSS), is a type of scheduling in which no waiting is allowed between machines when a product processed in one machine passes to the next machine. In fields such as the pharmaceutical industry, plastics industry, iron and steel industry. When jobs waits between the machines, deterioration in its structure may occur, as well as negatively affecting the production process and increasing costs. It is possible to find the best solution for problems with two or less machines. However, it may become impossible to find the best scheduling in problems with a large number of machines. Such problems are called NP-hard problems in the literature. Metaheuristic methods are preferred because exact methods are insufficient in solving such problems. Although metaheuristic methods do not guarantee the exact result, they are frequently used in the literature because they can provide solutions close to the optimum solution in a short time. In this study, Gray Wolf Optimization (GWO) algorithm, which is used in many scheduling problems, is used to solve NWFSS problems. The GWO was tested on 192 test problems that are well known in the literature and produced for NWFSS. In order to increase the performance of the algorithm, solutions produced according to Nawaz, Enscore, Ham (NEH) and Sorted Processing Time (SPT) methods were added to the initial population. In addition, it has been hybridized with the Variable Neighborhood Search algorithm as a local search algorithm to increase the exploitation ability of the algorithm. The results revealed that the proposed algorithm is quite effective and efficient for NWFSS problems.

KEY WORDS: No-Wait Flowshop Scheduling, Gray Wolf Optimization Algorithm, Maximum Completion Time, Local Search

TEŞEKKÜR

Tez çalışmalarım boyunca desteklerini esirgemeyen danışmanım Doç. Dr. İbrahim Berkan AYDİLEK hocama ve TÜBİTAK projesinde birlikte çalıştığımız Dr. Öğr. Üyesi Abdülkadir GÜMÜŞÇÜ, Dr. Öğr. Üyesi İ. Hakan KARAÇİZMELİ, Dr. Öğr. Üyesi Serkan KAYA ve Dr. Öğr. Üyesi Mehmet Emin TENEKECİ, hocalarıma katkılarından dolayı çok teşekkür ederim

Ayrıca tez çalışmalarım sırasında yardımlarını esirgemeyen arkadaşım Veysel TÜRK' e ve ayrıca tüm eğitim hayatım boyunca maddi ve manevi yanımda olan sevgili aileme çok teşekkür ederim.

Cengiz KINA

ŞEKİLLER DİZİNİ

Sayfa No

Şekil 3.1 Üretim çizelgeleme problemlerinin genel olarak sınıflandırılması 9
Şekil 3.2 3 iş 2 makine problemi Cmax değerleri 12
Şekil 3.3 GKO algoritmasında hiyerarşi 15
Şekil 3.4 Gri kurtların 2 ve 3 boyutlu uzayda muhtemel sonraki konumları 17
Şekil 3.5 GKO algoritmasında pozisyon güncellemesi 18
Şekil 3.6 Ava saldırma veya av arama 19
Şekil 3.7 GKO algoritması sözde kodu 20
Şekil 3.8 Yerel arama algoritması çalışma adımları 21
Şekil 3.9 DKA algoritmasında ekleme ve yer değiştirme adımları 22
Şekil 3.10 DKA algoritması sözde kodu 23
Şekil 3.11 HGKO algoritması akış diyagramı 27
Şekil 4.1 Makine sayılarına göre algoritmaların OGS değerlerinin karşılaştırılması 41
Şekil 4.2 İş sayılarına göre algoritmaların OGS değerlerinin karşılaştırılması 42
Şekil 4.3 8 iş 3 makine b problemi iterasyon sayısına göre cmax değerleri 43
Şekil 4.4 50 iş 25 makine b problemi iterasyon sayısına göre cmax değerleri 44
Şekil 4.5 200 iş 10 makine a problemi iterasyon sayısına göre cmax değerleri 45

ÇİZELGELER DİZİNİ

Sayfa No

Çizelge 3.1 3 iş 2 makine problemi hazırlık ve işlem süreleri 12
Çizelge 3.2 Örnek BATÇP Cmax değerleri 2 iş, 3 makine..... 13
Çizelge 4.1 İş sayısı 8 olan problemler için OGS değerleri 32
Çizelge 4.2 İş sayısı 10 olan problemler için OGS değerleri 33
Çizelge 4.3 İş sayısı 12 olan problemler için OGS değerleri 34
Çizelge 4.4 İş sayısı 50 olan problemler için OGS değerleri 35
Çizelge 4.5 İş sayısı 100 olan problemler için OGS değerleri 36
Çizelge 4.6 İş sayısı 150 olan problemler için OGS değerleri 37
Çizelge 4.7 İş sayısı 200 olan problemler için OGS değerleri 38
Çizelge 4.8 İş sayısı 250 olan problemler için OGS değerleri 39
Çizelge 4.9 Önerilen HGKO performansının CPLEX, Gen2, UÖY ve HKKO ile karşılaştırılması 40
Çizelge 4.10 Önerilen HGKO algoritmasının OGS değerlerinin Cplex, Gen2, UÖY, HKKO ile karşılaştırılması 41
Çizelge 4.11 Önerilen HGKO algoritmasının ranking değerlerinin karşılaştırılması 42
Çizelge 4.12 8 iş 3 makine b problemi 10 defa çalıştırma sonuçları..... 43
Çizelge 4.13 50 iş 25 makine b problemi 10 kez çalıştırılma sonuçları..... 44

SİMGELER ve KISALTMALAR DİZİNİ

ATÇP	Akış Tipi Çizelgeleme Problemleri
BATÇ	Beklemesiz Akış Tipi Çizelgeleme
BATÇP	Beklemesiz Akış Tipi Çizelgeleme Problemleri
Cmax	Maksimum Tamamlanma Zamanı
DKA	Değişken Komşuluk Arama
EKİZ	En Küçük İşlem Zamanı
EBİZ	En Büyük İşlem Zamanı
GA	Genetik Algoritma
Gen2	Genetik Sezgisel Algoritma
GKO	Gri Kurt Optimizasyonu
HKKO	Hibrit Karınca Koloni Optimizasyonu
HGKO	Hibrit Gri Kurt Optimizasyon Algoritması
HGLSA	Hibrit Genetik Yerel ve Global Arama Algoritması
MILP	Tamsayılı doğrusal programlama modeli
NEH	Nawaz, Enscore, Ham
NP	Polinom zamanda çözülemeyen problemler
OGS	Ortalama Göreli Sapma
SİS	Sıralı İşlem Süresi
Tmax	Toplam Akış Zamanı
UÖY	Uyarlanabilir Öğrenme Yaklaşımı

1. GİRİŞ

Çizelgeleme, bir üretim sürecinde makinelerdeki işlerin hangi sırada işleneceğinin belirlenmesi olarak tanımlanabilir (Đumić ve Jakobović, 2021). İşlerin tamamının en kısa sürede tamamlanması, makinelerdeki bekleme süresinin en aza indirilmesi, makinelerin toplam çalışma süresinin minimuma indirilmesi veya siparişlerdeki gecikmelerin minimuma indirilmesi gibi amaçlar doğrultusunda farklı amaçlar için farklı iş sıralamaları yapılabilir (Belabid ve ark., 2020; Ren ve ark., 2021). Çizelgeleme istenen amaçlar doğrultusunda, belirli kısıtlar altında ve istenen zamanda, kaynakların işlere atanması ile ilgili karar verme sürecidir (Branda ve ark., 2021; Miyata ve Nagano, 2019). Çizelgeleme sonucunda hangi işin hangi sıra ile hangi makinede yapılacağına karar verilir (Küçükkoç, 2018). Çalışma sahasındaki ve iş gereksinimlerindeki farklılıklar çizelgeleme problemi atölye tipi, akış tipi, hibrit atölye tipi gibi çeşitli türlere ayrılmıştır (Gao ve ark., 2022).

İşletmelerin artan rekabet ortamında kârlılıklarını devam ettirebilmesi için üretim planlanması ve müşteri memnuniyeti önemli bir rol oynamaktadır. Dünyada pazar rekabetinin giderek artması dolayısıyla işletmeler için bireysel müşteri taleplerinin zamanında gerçekleşmesi (örneğin, sipariş edilen bir ürünün zamanında teslim edilmesi vs.) müşteri devamlılığı, işletmenin karlılığı için çok önemlidir. Üretim planlama sorunları yirminci yüzyılın ortalarından itibaren birçok araştırmacının ilgisini çekmiştir (Branda ve ark., 2021). Akış tipi problemi, literatürde kapsamlı olarak tartışılan bir tesis yerleşim planıdır (Branda ve ark., 2021). Literatür kapsamında popülasyon tabanlı metasezgisel yaklaşımlarından Karınca Koloni Optimizasyonu, Genetik Algoritma, Benzetimli Tavlama, Aritmetik Optimizasyon Algoritması, Komşuluk değişimi tabanlı tabu arama algoritması, Parçacık Sürü Optimizasyonu yöntemleri akış tipi problemlerin çözümünde uygulanan yöntemlerdendir (Umam ve ark., 2021; Gümüşçü ve ark., 2021). Son yıllarda optimizasyon problemlerinde başarılı sonuçlar üreten Gri Kurt Optimizasyon (GKO) yöntemi akış tipi çizelgeleme problemlerin çözümünde etkili olabilir (Li ve Feng, 2021). GKO, optimizasyon problemlerinde başarılı çözümler üreten metasezgisel

yöntemlerinden PSO ve GA algoritmaları ile karşılaştırıldığında rekabetçi bir performans ürettiği gözlemlenmiştir (Li ve Feng, 2021). GKO algoritması workshop, bulut teknolojisinde, termodinamik, güç sistemlerinde yaygın olarak kullanılmaktadır (Li ve Feng, 2021).

Üretim çizelgeleme problemlerinde işlerin akış şekline göre sınıflandırılan türlerinden ve NP-hard (makine sayısı 2'den fazla olduğunda NP-hard optimizasyon problemi olarak bilinir) optimizasyon problemlerinden akış tipi çizelgeleme problemleri (ATÇP) günümüzde iletişim, ulaşım, bilgisayar tasarımları gibi birçok sektörde kullanılmaktadır (Abdel-Basset ve ark., 2021; Alawad ve Abed-alguni 2022; Babor ve ark., 2021). Akış tipi çizelgeleme, farklı görevleri yerine getiren m adet makine ve n adet işin olduğu, her işe ait m adet aşamanın bulunduğu, her aşamanın farklı makinelerde işlendiği, tüm işlere ait aşamaların aynı sıra ile yapıldığı problem tipleridir. (Umman ve ark., 2021). Akış tipi çizelgeleme probleminde optimum sonucunu veren ilk çalışma 1954 yılında Johnson tarafından önerilmiştir (Johnson, 1954). Johnson m makine ve n işli ATÇP problemi üzerinde tamamlama süresini minimize etmek için eniyileme algoritmasını önermiştir (Yağmahan ve Yenisey, 2011). ATÇP'de genellikle amaç fonksiyonu tamamlanma zamanını minimize etmeyi hedeflenmektedir. Günümüzde rekabetin artmasıyla birlikte birçok üretim firması maliyetleri düşürerek en kısa zamanda istenen üretimi gerçekleştirmek üzere çalışmalar yapmaktadır. ATÇP'de matematiksel model kullanılarak optimal çözüme ulaşılabilir ancak gerçek hayattaki çoğu problemin boyutu büyük olduğu için matematiksel model yetersiz kalmaktadır. Bundan dolayı optimal çözüme ulaşmak yerine optimal çözüme yakın bir sonuç veren ve kısa sürede çözüm elde edilen metasezgisel yöntemler tercih edilmektedir. Son zamanlarda araştırmacılar metasezgisel yöntemler üzerinde çalışmalar yaparak çeşitli yöntemler geliştirilmiştir.

Beklemesiz akış tipi çizelgeleme problemi ise son yıllarda giderek önemi artan bir çizelgeleme problemidir. Bu problem çeşidinde işler belli bir sıra ile makinelerde işlenirken bir iş bir makineden çıktıktan sonra bekleme olmaksızın diğer makinede işlenmektedir (Keskin ve Engin, 2021). BATÇP kimya, petrol, ilaç, demir çelik sanayi gibi çeşitli alanlarda yaygın olarak kullanılmaktadır (Lin ve ark., 2018). Örneğin, çelik endüstrisinde bir makineden çıkan erimiş çelik soğumadan ve formunu yitirmeden bir

dizi işlemden geçmesi gerekmektedir. Ürünün makineler arasında beklemesi yapısında bozulmaya sebebiyet verebileceği için bu tür problemler beklemesiz kısıtlı ele alınmalıdır. Günümüzde üretim sürecinde kaynak kullanımının dengelenmesi, akış süresini en aza indirmesi sayesinde üretim ortamında önemli bir yaklaşımdır (Keskin ve Engin, 2021). BATÇ problemlerinde makespan performansı yaygın olarak kullanılan bir kriterdir. Burada makespan kriteri kaynak kullanımı ile ilgilidir (Keskin ve Engin, 2021).

Bu çalışmada BATÇ problemlerinde maksimum tamamlanma zamanını minimize etmek için hibrit bir gri kurt optimizasyon algoritması geliştirilmiştir. Gri Kurt Optimizasyon algoritmasının çözüme yakınsama performansını arttırmak için yerel arama algoritması ile hibrit hale getirilmiştir. Ayrıca çeşitli algoritmalar ile üretilen başlangıç değerlerinin rastgele üretilen başlangıç popülasyonuna eklenmesi ile çözüme daha hızlı bir yakınsama sağlanmıştır. Önerilen algoritma 192 benchmark problem seti üzerinde test edilmiştir. Algoritmanın test sonuçları daha önceden aynı veri setini kullanan Engin ve Günaydın (2011), tarafından sunulan Uyarlanabilir Öğrenme Yaklaşımı (UÖY), Aldowaisan ve Allahverdi (2003), tarafından sunulan Genetik Sezgisel 2 (Gen2) ve Engin ve Güçlü (2018), tarafından sunulan Hibrit Karınca Koloni Optimizasyonu (HKKO) algoritmaları ile karşılaştırılmıştır.

Bu tez çalışmasında; Bölüm 2'de BATÇ problemleri ve GKO ile ilgili literatürde yapılmış çalışmalar özetlenmiştir. Bölüm 3'de BATÇ, doğrusal programlama yöntemi, önerilen yöntemin teorik çerçevesi, ALA, Gen2, HKKO algoritmalarının teorik çerçevesi, önerilen yöntemlerin test sürecinde kullanılan veri seti, önerilen algoritmalarda kullanılan parametreler alt başlıkları ile materyal ve yöntemler verilmiştir. Bölüm 4'de tez kapsamında önerilen HGKO algoritmasının deneysel çalışma sonuçları ve yorumlar belirtilmiştir. Son olarak beşinci bölümde, sonuç ve öneriler verilmiştir.

2. ÖNCEKİ ÇALIŞMALAR

Çizelgeleme problemleri literatürde geniş çapta tartışılmaktadır. Çalışanların iş sırasının hazırlanması, trenler ve otobüsler için sefer saatlerinin belirlenmesi, eğitim kurumlarındaki ders programlarının hazırlanmasında çizelgeleme yöntemleri kullanılmaktadır. Bu tür çizelgeleme problemleri karmaşık yapıda olabilir. Bundan dolayı matematiksel formüller ile çözümün sağlanması zordur. Bir problemlerin çözümünde kullanılan en etkili algoritmanın, çalışma süresi, verilen veriye polinom cinsinden bağlı ise bu tür algoritmalara Polinomal (P) zamanda çalışan algoritma denilmektedir. Örnek olarak verilen bir sayının asal sayı olup olmadığı tespit etmek için geliştirilen bir algoritma P kategorisinde bir algoritma olacaktır. P tipi problemlere çözüm üretebilecek uygun bir algoritma, belirli bir süre içerisinde problemi çözebilmektedir. Eğer bir problem polinomal değilse, Nondeterministik Polinomal (NP) olarak adlandırılır ve bu tür problemleri sınırlı bir sürede çözmek çok zordur. "NP tipi problemlere polinomsal zamanda bir çözüm bulunamamakta, ancak üstel zamanda etkin olarak çözülebilmektedir" (Özsağlam, 2009).

Beklemesiz akış tipi çizelgeleme endüstride özellikle kimya, plastik sanayi, petrol, ilaç sanayisi gibi alanlarda kullanımında makinelerde işlenecek işlerin diğer makineye geçişinde herhangi bir bekleme olmaması gerekir. Aksi takdirde ara ürünlerin bozulması veya maliyetlerin artırmasına sebep olabilir. Bu tür problemler literatürde Beklemesiz akış tipi çizelgeme olarak adlandırılmaktadır.

Sivrikaya ve ark. (1998), iki makineli permütasyon akış tipi probleminde ağırlıklı maksimum tamamlanma süresini ve ağırlıklı ortalama akış süresini minimize etmek için Nagar ve ark. (1995) önerdiği dal-sınır yaklaşımını iyileştirmiş ve iş sayısının 18 ve daha küçük olan problemler için çözmüştür. Önerilen dal-sınır algoritmasının optimum değerlerinin Ignall and Schrege (1965)'e daha başarılı olduğunu ortaya koymuşlardır.

Burke ve Newall (1999)'a göre yüksek boyutlu çizelgeleme problemlerinin çözümünün sağlanması çok zordur. Yazarlara göre optimizasyon yöntemlerde önemli

çözüm yöntemlerinden genetik algoritma ve diğer ileri arama yöntemlerinin kullanılması sonucunda da optimum çözümün elde edilmesi zordur. Yazarlar çalışma kapsamında yüksek boyutlu problemlerinin çözümünü sağlayabilmek için genetik algoritma yöntemini kullanmışlardır. Genetik algoritmanın problemin çözümünü sağlayabilmesi için yüksek boyutlu problemleri küçük parçalara ayrıştırarak çözebilen bir yöntem geliştirmişlerdir. Önerilen yöntemin deneysel çalışmalar sonucunda çözüm süresini azaltmış ve diğer çalışmalara göre daha başarılı olduğu göstermişlerdir.

Taşgetiren ve ark. (2011), permütasyon akış tipi çizelgeleme problemlerinde toplam akış süresinde optimum çözüme ulaşmak için kesikli arı koloni algoritmasını geliştirmişlerdir. Minimum toplam akış süresine karşılık gelen sıralamayı bulmak için arı koloni algoritması ve açgözlü algoritmayı birlikte kullanmışlardır. Çalışmanın performansı Taillard'ın 90 karşılaştırma problemine uygulanmış ve test edilmiştir. Geliştirilen algoritma, 90 problemin 44 tanesinde daha iyi sonuç üretmiştir.

Aydilek ve Allahverdi (2012), BATÇ problemleri için birden fazla sezgisel algoritma önermişlerdir. Problemi maksimum tamamlanma zamanı ve ortalama tamamlanma zamanı amaç fonksiyonu ile ele almışlardır. Benzetimli tavlama ve HA ismini verdikleri yeni bir sezgisel algoritmanın hibrit bir şekilde kullanılmasıyla 5 farklı sezgisel algoritma önermişlerdir.

Liu ve ark. (2013), BATÇ problemleri için çeşitli sezgisel algoritmalar önermişlerdir. Problemin maksimum gecikme amaç fonksiyonu ile çözümü için 6 yeni sezgisel algoritma önermişlerdir. Hesaplama zamanını azaltmak için hızlandırma teknikleri uygulanmış ve Gelişmiş NEH algoritmasının problemin çözümünde etkili olduğunu ortaya koymuşlardır.

Qin ve Zhang (2016), Beklemesiz Esnek Atölye Tipi Çizelgeleme problemlerinin iki amaçlı çözümü için Elit Parçacık Sürü Optimizasyon algoritması geliştirmişlerdir. Problemi maksimum tamamlanma zamanı ve maksimum gecikme zamanı amaç fonksiyonu ile ele almışlardır. Algoritmanın başlangıç popülasyonu oluşturma aşamasında 5 sezgisel yöntem ve rastgele üretilmiş değerler kullanılmıştır. PSO'da yerel optimuma takılmaları önlemek için çift ekleme karışıklığı algoritması ve

her parçacığın kişisel en iyi değerini (pbest) güncellemek için çaprazlama mekanizmasını eklenmiştir.

Allahverdi ve ark. (2018), BATÇ problemlerini, toplam gecikme ve tamamlanma zamanı olarak iki performans ölçütü ile ele almışlardır. Benzetimli tavlama ve ekleme algoritmalarının birleşiminden oluşan AA algoritmasını geliştirmişlerdir. Benzetimli tavlama kısmı toplam gecikmeyi azaltmak için yeni çözümleri keşfederken ekleme algoritması bulunan çözümleri problem kısıtları sağlayacak şekilde iyileştirmektedir. Önerilen AA algoritmasını literatürde iyi bilinen algoritmalarla aynı problem seti üzerinde test edip karşılaştırmışlar. Deneysel sonuçlar AA algoritmasının karşılaştırılan diğer algoritmalara göre daha başarılı olduğunu göstermektedir.

Taşgetiren ve ark. (2019), BATÇ problemlerini toplam enerji tüketimi ve toplam akış zamanı amaç fonksiyonu ile ele almışlardır. Problem matematiksel olarak çözümü için karışık tam sayılı programlama (Mixed Integer Linear Programming, MILP) kodlanmıştır. Problemin Np-Zor olmasından dolayı sezgisel yöntem olarak Ayrık Yapay Arı Koloni algoritması ve Genetik Algoritma geliştirilmiştir.

Keskin ve Engin (2021), önerdikleri çalışmada Hazırlık Zamanlı Beklemesiz Akış Tipi Çizelgeleme problemleri için hibrit bir algoritma önermişlerdir. Problemi maksimum tamamlanma zamanı (Cmax) ve toplam akış zamanı (Tmax) amaç fonksiyonları için çözmeyi amaçlamışlardır. Problemi iki amaçlı çözümü için Hibrit Genetik Yerel ve Global Arama Algoritması (HGLSA) geliştirmişlerdir. Geliştirilen hibrit genetik algoritmaya, eklemeli arama ve kendini tamir etme algoritmaları adapte edilmiştir. Yaptıkları deneysel çalışmalar sonucunda HGLSA algoritmasının hazırlık zamanlı BATÇP çözümünde etkili ve verimli olduğunu ortaya koymuşlardır.

Samarghandi ve ark. (2014), yaptıkları çalışmada BATÇ problemlerini sıra bağımlı ve hazırlık zamanlı olarak ele almışlardır. Problem Makespan kriteri ile alınmıştır. Problemin karmaşıklığı ile baş edebilmek için Matris kodlama ismi verilen bir şifreleme yöntemi ile iş sıralamaları kodlamıştır. Geliştirilen Parçacık Sürü Optimizasyonu algoritması arama uzayında şifrelenen daha iyi keşfetmek için iş

sıralamaları üzerinde uygulanmıştır. Algoritmanın başarısını test etmek için literatürde bulunan çok sayıda benchmark problemine uygulanmıştır. Önerilen algoritmanın test edilen diğer algoritmalara göre daha başarılı olduğunu belirlemişlerdir.

Samarghandi (2015), BATÇ problemlerini teslim tarihi kısıtı bulunan maksimum tamamlanma zamanı amaç fonksiyonu ile ele almışladır. Problemin çözümü için Parçacık Sürü Algoritması (PSO) geliştirilmiştir. Ayrıca başlangıç popülasyonunun üretilmesi için çeşitli dağıtım kullarının etkisi incelenmiştir. Parametrelerin farklı değerlerinin algoritmanın performansı üzerindeki etkisini belirlemek için Taguchi tabanlı bir deneysel ortam tasarlanmıştır.

Kaya ve ark. (2020), yaptıkları çalışmada Aydilek (2018) tarafından geliştirilen Hibrit ateş böceği ve parçacık sürü optimizasyonu algoritması (HAPSO) başlangıç çözümlerinin Akış tipi çizelgeleme problemin çözümündeki etkisini incelemek için 5 farklı yöntem kullanmışlar. Hangi başlangıç çözümünün daha iyi çözümler ürettiği farklı problem setleri üzerinde test edilmiştir. HAPSO algoritması için başlangıç çözümleri Neh algoritması ile sıralandığında daha başarılı sonuçlar üretildiğini belirtmişlerdir.

Karakoyun (2021), yaptığı çalışmada Kurbağa Sıçrama (SFLA) ve Gri Kurt Optimizasyonu (GKO) algoritmalarını hibrit bir şekilde kullanarak çok amaçlı optimizasyon problemlerine uygulamıştır. Önermiş olduğu algoritmayı çok amaçlı optimizasyon algoritmalarının performansıyla karşılaştırmıştır. Algoritmanın local en iyi değere takılmasını önlemek için Levy Uçuşu yönetmini kullanmıştır. Ayrıca GKO algoritmasındaki Alfa, Beta, Delta kurtlarının çözüm değerlerinin ağırlıklarını dinamik olarak seçerek çözümlerin iyileşmesini sağlamıştır.

Yang ve Liu (2018), Bulanık bloklamalı akış tipi çizelgeleme problemleri için çok amaçlı hibrit bir Gri Kurt Optimizasyon algoritması geliştirmişlerdir. Hibrit çok amaçlı gri kurt optimizasyonunda, belirli bir kalitede ilk çözümler üretmek için yeni bir sezgisel işlem tasarlanmıştır ve algoritmanın sömürü yeteneğini arttırmak için yerel bir arama stratejisi entegre edilmiştir. Hibrit çok amaçlı gri kurt optimizasyonunun performansı, gemi yapımında panel blok montajının üretim örnekleri ile test edilmiştir.

Jieran ve ark. (2020), yaptıkları çalışmada Esnek Atölye tipi çizelgeleme problemleri için hibrit bir Gri Kurt Optimizasyon algoritması önermişlerdir. Problemi minimum tamamlanma zamanı kısıtı ile ele almış ve algoritmanın keşif yeteneğini arttırmak için algoritmaya yerel arama algoritması entegre etmişlerdir.

Makhadmeh ve ark. (2021), akıllı evlerde enerji çizelgeleme problemleri için min-çatışma yerel arama algoritması Minimum çakışma algoritması ve GKO algoritmalarını hibrit bir şekilde kullanmışlardır. Çizelgeleme probleminde GKO algoritmasının yakınsama hızını arttırmak için Minimum çakışma algoritması kullanılmıştır. Çalışmada evde kullanılan enerji tüketimi minimize edilmesi ile elektrik faturalarında önemli bir tasarruf sağladığı gösterilmiştir. Geliştirilen algoritmayı aynı alanda yapılmış daha önceki çalışmalar ile kıyasladıklarında kayda değer şekilde daha başarılı olduğunu söylemişlerdir.

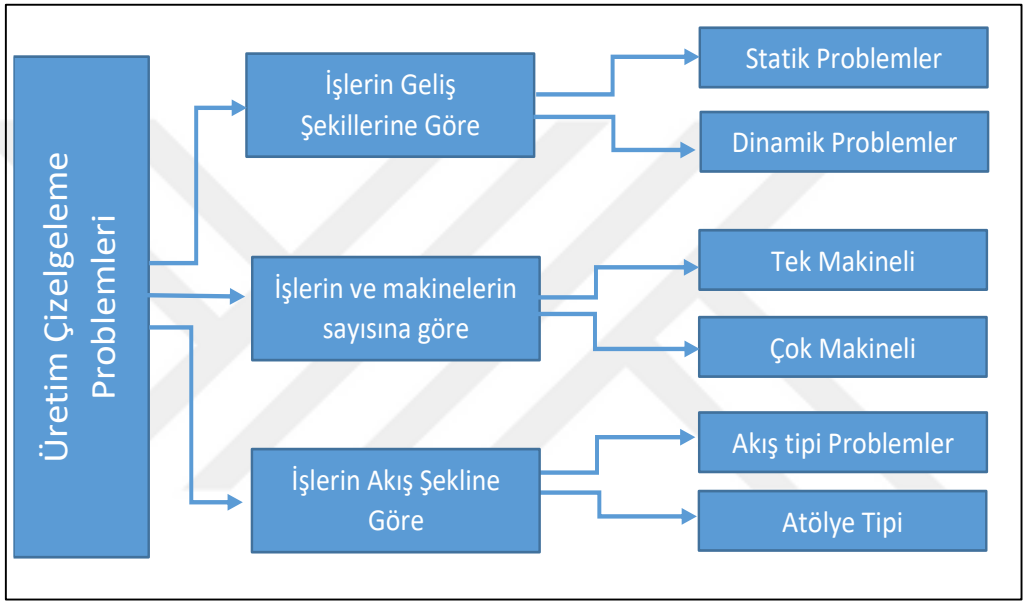
Jiang ve Zhang (2018), atölye tipi çizelgeleme ve esnek atölye tipi çizelgeleme problemlerinin maksimum tamamlanma zamanını minimize etmek için ayrık bir GKO algoritması geliştirmişlerdir. Geliştirilen algoritmada arama operatörü, algoritmanın çalışmasını doğrudan ayrık bir alanda sürdürmek için çaprazlama işlemine dayalı olarak tasarlanmıştır. Daha sonra popülasyon çeşitliliğini sağlamak ve erken yakınsamayı önlemek için uyarlanabilir bir mutasyon yöntemi tanıtılmıştır. Ek olarak algoritmanın keşif yeteneğini arttırmak için değişken bir komşuluk arama yöntemi kullanılmıştır. Deneysel sonuçlar önerilen algoritmanın bu çalışmada ele alınan çizelgeleme problemlerinde daha iyi performans gösterdiğini ortaya koymuştur.

Peng ve Zhou (2019), otomotiv endüstrisinde kamyon çizelgeleme problemleri için iki amaçlı hibrit bir GKO algoritması önermişlerdir. Algoritmanın yakınsama ve keşif kabiliyetini geliştirmek için Gaus mutasyonu ve uyarlanmış yerel arama algoritmaları eklenmiştir. Deneysel sonuçlara göre önerilen algoritmanın iki amaçlı kamyon çizelgeleme problemleri için etkili ve verimli olduğunu ortaya koymuşlardır.

Zhao ve ark. (2019), çalışmalarında BATÇ problemleri için hibrit bir parçacık sürü algoritması önermişlerdir. Algoritmanın başlangıç popülasyonun oluşturulmasında K en yakın komşu ve NEH algoritmalarını kullanmışlardır.

3. MATERYAL ve YÖNTEM

Çizelgeleme problemleri, işlerin geliş şekillerine göre, işlerin ve makinelerin sayılarına göre ve işlerin akış şekillerine göre çeşitli alt problemlere ayrılabilir. Çizelgeleme problemleri Şekil 3.1.'deki gibi en çok kullanılan problem tiplerine göre alt problemlere ayrılabilir (Tekbaş, 2011; İkucukkoc, 2022).



Şekil 3.1 Üretim çizelgeleme problemlerinin genel olarak sınıflandırılması (İkucukkoc, 2022)

3.1 Beklemesiz Akış Tipi Çizelgeleme

BATÇ'de n adet iş ve bu işlerin atanacağı ve sırasıyla işleneceği m adet makine bulunur. Makineler ardışık olarak yerleştirilmiştir ve işler sırasıyla aynı makineler üzerinde aynı aşamalardan geçer (Zhao ve ark., 2020). Her iş öncelikle birinci makinede işlenmeli daha sonra bir sonraki makinede işlenerek en sondaki makineye kadar ardışık olarak devam etmelidir (Lin ve ark., 2018). İşler bir makineden diğerine geçerken herhangi bir bekleme ve gecikme olmamalıdır (Keskin ve Engin, 2021). Her işin farklı makinelerdeki işlem süresi farklılık gösterebilmektedir (Lin ve ark., 2018). Bir iş bir makineye geldiğinde o esnada makinenin meşgul olmaması ve işin

beklememesi için işlem süreleri önceden hesaplanmalı ve o işin ilk makinedeki başlangıç zamanı ertelenmelidir. BATÇ problemlerinde ihtiyaca göre kullanılabilecek birçok amaç fonksiyonu bulunmaktır. Bunlardan literatürde en çok kullanılanlar aşağıda verilmiştir.

Cmax, amaç fonksiyonunda makinelerden son çıkan işin bitiş zamanını ifade etmektedir. Maksimum tamamlanma zamanı olarak da ifade edilmektedir.

Tmax, amaç fonksiyonu ise işlerin makinelerde işlenirken toplam harcanan zamanı ifade etmektedir.

Teslim tarihi, kısıtı ise müşterilerin talepleri doğrultusunda işlerin teslim edilmesi gereken zamandan minimum sapmayı ifade eden bir amaç fonksiyonudur.

Enerji verimliliği, son yıllarda üzerinde çalışılan diğer bir amaç fonksiyonudur. Burada makinelerde işlenecek işler için minimum düzeyde kaynak ayrılması makinelerin olabildiğince verimli çalışarak fazla enerji tüketiminin önüne geçmektir.

Bunun gibi problemin tipine göre birçok farklı amaç fonksiyonu bulunmaktadır. Ayrıca birçok farklı amacın aynı anda sağlanması gereken çok amaçlı kullanımlar da mevcuttur. Bu çalışmada ise BATÇ problemleri için en yaygın olarak kullanılan amaç fonksiyonu Maksimum Tamamlanma Zamanı (Makespan, Cmax) kullanılmıştır. Cmax matematiksel olarak aşağıdaki şekilde tanımlanabilir;

j işinin k makinesi üzerindeki işlem süresi $p_{(j,k)}$ olarak ifade edilmektedir. n adet işin (j=1,2, … , n) her biri m adet makine (k=1,2, … , m) üzerinde sırasıyla işlenecektir. Her bir j işi her bir makinede işlenecek m adet (o_{j1}, o_{j2}, … , o_{jm}) operasyondan oluşmaktadır.

Beklemesiz kısıtını sağlamak için o_{jk} operasyonun tamamlanma süresi o_{jk+1} operasyonunun en erken başlama zamanına eşit olması gerekir (k= 1, 2, … , m-1). Yani her bir n iş için ardışık makinelerdeki operasyonları arasında herhangi bir bekleme olmaması gerekiyor. Amacımız bu şartları sağlayan ve maksimum

tamamlanma zamanını minimize edecek bir çizelge bulmaktır. Herhangi bir aşamada bir iş sadece bir makine üzerinde işlenebilmektedir ve bir makine her seferinde yalnızca bir iş üzerinde çalışabilmektedir.

$\Pi = \{\pi_1, \pi_2, \ldots, \pi_n\}$ makinelerde işlenecek işlerin bir sıralamasını ifade etmektedir. $L(\pi_{j-1}, \pi_j)$ beklemesiz kısıtı altında π_j, π_{j-1} ardışık iki işin son makinedeki tamamlanma süreleri farkını belirtmektedir. Ardışık iki işi arasındaki gecikme Denklem 3.1 ve Denklem 3.2 ye göre hesaplanabilir (Pan ve ark, 2008; Qian ve ark. 2009).

$$L(\pi_{j-1}, \pi_j) = p(\pi_{j-1}, 1) + \max\left[0, \max_{2 \le k \le m}\left\{\sum_{h=2}^{k} p(\pi_{j-1}, h) - \sum_{h=1}^{k-1} p(\pi_j, h)\right\}\right] \quad (3.1)$$

Cmax maksimum tamamlanma süresinin hesaplanması;

$$C_{max}(\pi) = \sum_{j=2}^{n} L(\pi_{j-1}, \pi_j) + P_{sum}(\pi_n) \quad (3.2)$$

$$P_{sum}(\pi_n) = \sum_{k=1}^{m} p(\pi_n, k) \quad (3.3)$$

Makespan kriterine sahip BATÇ, tüm permütasyonlar (Π) kümesinde bir permütasyon (Π*) bulunmaktır, Π öyle ki;

$$\pi^* = \arg\{C_{max}(\pi)\} \rightarrow min \quad \forall \pi \in \Pi \quad (3.4)$$

Çizelge 3.1.'de verilen 2 makine ve 3 işin olduğu örnek bir BATÇ problemi için işerlerin makinelerdeki hazırlık süreleri ve işlem süreleri verilmiştir. Verilen örnek problem için maksimum tamamlanma zamanının hesaplanması Şekil 3.1.'de gösterilmiştir (Shyu ve ark., 2004).

Çizelge 3.1 3 iş 2 makine problemi hazırlık ve işlem süreleri

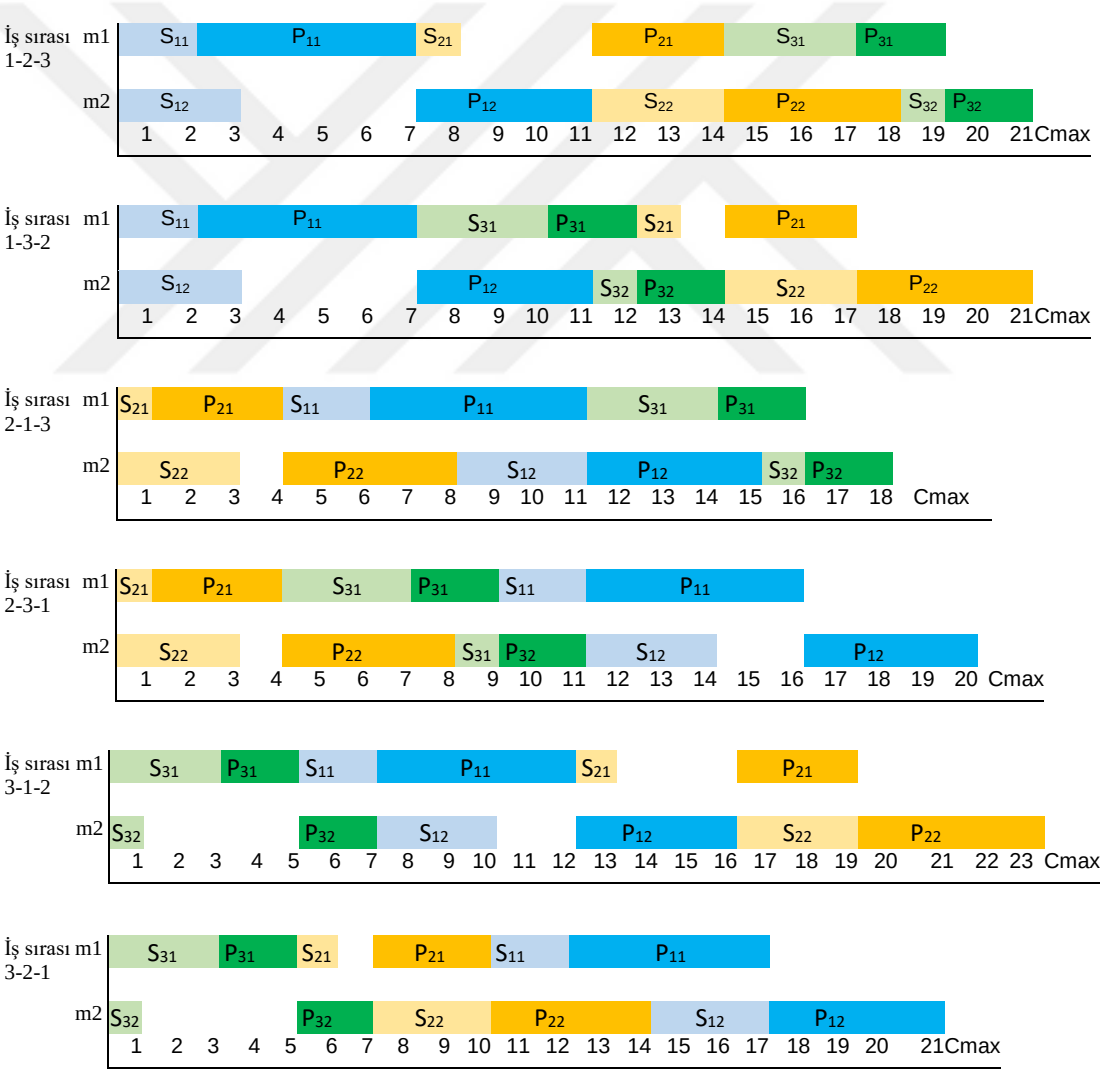
İşler	1	2	3
P_{i1}	5	3	2
P_{i2}	4	4	2
S_{i1}	2	1	3
S_{i2}	3	3	1

P_{i1} : i işinin 1. makinedeki işlem süresi

P_{i2} : i işinin 2. makinedeki işlem süresi

S_{i1} : i işinin 1. makinedeki hazırlık süresi

S_{i2} : i işinin 2. makinedeki hazırlık süresi



Şekil 3.2 3 iş 2 makine problemi Cmax değerleri

BATÇ problemlerinde işlerin farklı permütasyonları için maksimum tamamlanma zamanları farklık gösterir. Çizelge 3.2.'de verilen örnek problem için elde edilebilecek 6 farklı sıralama için Cmax değerleri hesaplanmıştır. En küçük Cmax değeri işlerin 2-1-3 şeklinde sıralanması ile elde edilmiş ve Cmax değeri 18 olarak bulunmuştur (Başar, 2021).

Çizelge 3.2 Örnek BATÇP Cmax değerleri 2 iş, 3 makine

İş sıralaması	Cmax değeri
1-2-3	21
1-3-2	21
2-1-3	18
2-3-1	20
3-1-2	23
3-2-1	21

3.2 Doğrusal Programlama Yöntemi

Önerilen yöntemin başarısı kıyaslamak amacıyla problem Cplex ortamında Karışık Tamsayılı Lineer Programlama (Mixed Integer Linear Programming, MILP) yöntemi ile çözülmüştür. Fakat problem boyutu büyüdüğü zaman matematiksel model kısıtlı bir süre içerisinde optimum sonuç bulamamaktadır. Dolayısıyla iş sayısının 8,10,12 olduğu 72 problem için optimum değerler hesaplanmıştır. Daha büyük problemler için programın maksimum çalışma süresi 3600 saniye ayarlanarak sonuçlar alınmıştır. MILP için problem formülasyonu aşağıda verilmiştir.

n : Çizelgelenecek iş sayısı(j = 1,2, …, n)
m : Makine sayısı(k = 1,2, …, m)
$P_{j,k}$: j işinin k makinesindeki işlem süresi
P[j,k] : j pozisyonunda olan işin k makinesinde işlem süresi
$S_{j,k}$: j işinin k makinesindeki hazırlık süresi
S[j,k]: j pozisyonunda olan işin k makinesindeki hazırlık süresi
C_{max} : Maksimum tamamlanma zamanı(cmax)
$C_{j,k}$: j işinin k makinesindeki tamamlanma süresi
C[j,k]: j pozisyonunda olan işin k makinesindeki tamamlanma süresi
$X_{[j,k]}$: Binary bir değişken, j işi k. sırada yapılıyorsa 1, diğer durumda 0

Problemde hazırlık süreleri hem işlere hem de makinelere bağımlıdır. Eğer k. makinedeki hazırlık süresi tamamlandıysa ve (j-1) pozisyonunda bulunan işin (k-1) makinesinde operasyonu tamamlandıysa, o zaman j. pozisyonda bulunan iş k. makinede hemen başlamalıdır. j pozisyondaki işin hazırlık zamanının başlaması için (j-1) pozisyonundaki iş bitmesine gerek yoktur. C_j tamamlanma zamanı işlerin her iki operasyonun bittiği zamanı ifade eder.

Minimize C_{max}

$$C_{[j,k]} = C_{[j-1,k]} + \sum_{j=1}^{n} x_{[j,k]} \left[\max\{S_{[j,k]}, P_{[j,k-1]} - P_{[j-1,k-1]} - P_{[j-1,k]} + \max\left(S_{[j,k-2]} + P_{[j,k-2]}, P_{[j-1,k-1]} + S_{[j,k-1]}\right)\} + P_{[j,k]}\right], \forall j, k > 1 \quad (3.5)$$

$$C_{[j,k]} \geq \sum_{j=1}^{n} x_{[j,k]} \left[S_{[j,k]} + P_{[j,k]}\right], \quad \forall j, k = 1 \quad (3.6)$$

$$C_{[j,k]} \geq 0, \quad \forall j, k \quad (3.7)$$

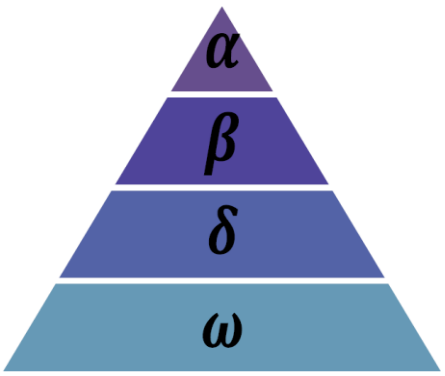
$$X_{[j,k]} \in \{0, 1\}, \quad \forall j, k \quad (3.8)$$

Modelde amaç fonksiyonumuz maksimum tamamlanma süresini minimize etmektir. Denklem (3.5) iki ardışık makindeki işlerin tamamlana süreleri arasındaki ilişkiyi ifade etmektedir. Denklem (3.6) ilk makinedeki tamamlanma süresinin en az hazırlık süresi ve işlem süresinin toplamı kadar olmasını sağlıyor. Denklem (3.7) her işin tamamlanma süresinin negatif değer almasını önlemektedir. Denklem (3.8) ikili değişkenleri tanımlamaktadır.

3.3 Gri Kurt Optimizasyon Algoritması

GKO algoritması Mirjalili tarafından gri kurtların doğada avlanma, liderlik ve hiyerarşi davranışlarından esinlenilerek oluşturulmuş bir metasezgisel algoritmadır (Mirjalili, 2014). Gri kurtlar besin zincirinin tepesinde bulunan uç yırtıcılardır. Grup içerisinde çok sıkı sosyal hiyerarşi bulunur ve her bir kurt sürüsü ve ortalama 5-12

kurttan oluşur. Gri kurtlar Alfa (α), Beta (β), Delta (δ), Omega (ω) olmak üzere dört gruba ayrılmıştır. Gri kurt sürüsünde lider kurt Alfa olarak isimlendirilmektedir. Alfa kurt ava öncülük etme, dinlenme ve uyanma vakitlerinin belirlenmesi gibi birçok konuda karar verici konumdadır. Sürüye liderlik eder ve verdiği komutlar diğer kurtlar tarafından uygulanır. Beta kurtlar ise hiyerarşide Alfa kurttan sonra gelmektedir. Beta kurtlar, Alfa kurdun komutlarına uyar ve grubun diğer bireylerine iletir ayrıca Alfa kurdun karar vermesinde yardımcı olurlar. Hiyerarşinin üçüncü adımında ise Delta kurtlar bulunur. Bunlar da Alfa ve Beta kurtların komutlarına uyar ve Omega kurtlara komut verirler. Omega kurtların ise hiyerarşide bir yetkileri yoktur. Diğer kurtların verdiği komutlara uymak zorundadırlar.



Şekil 3.3 GKO algoritmasında hiyerarşi (Mirjalili, 2014)

Ayrıca gri kurtların avlanması üç adımda ele alınmıştır. Bunlar; av arama, avı kuşatma, ve ava saldırma adımlarından oluşmaktadır.

3.3.1 Sosyal hiyerarşi

GKO algoritmasında en iyi çözüm alfa, ikinci en iyi çözüm beta, üçüncü en iyi çözüm ise delta olarak adlandırılmaktadır. Bunların dışında kalan çözümler ise omega olarak kabul edilmektedir. GKO algoritmasında avlanma α, β ve δ kurtlar tarafından yönlendirilmektedir. ω kurtlar ise bu 3 kurdu takip eder (Mirjalili, 2014).

3.3.2 Avı kuşatma

Gri kurtlar avlanma esnasında avlarını kuşatır ve yavaş yavaş avlarına yaklaşırlar. GKO algoritmasında avı kuşatmak Denklem (3.9) ve Denklem (3.10)'a göre yapılmaktadır.

$$D = |C.X_p(t) - X(t)| \quad (3.9)$$

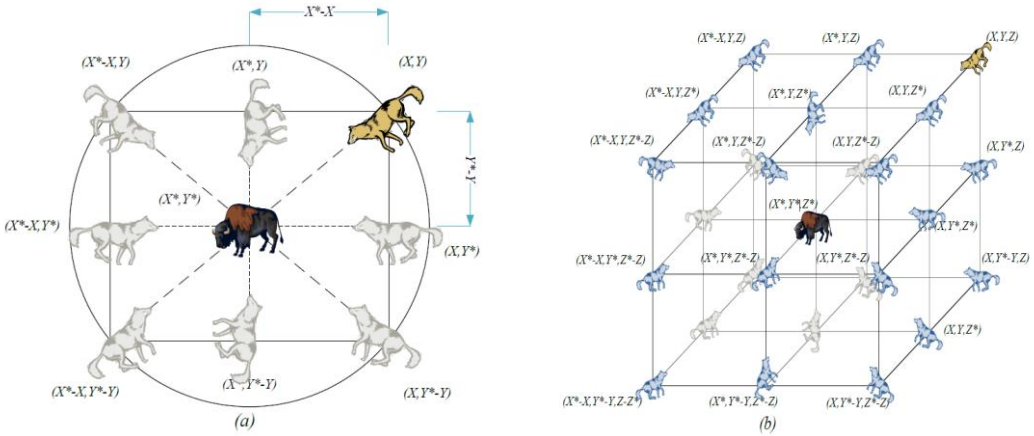
$$X(t+1) = X_p(t) - A.D \quad (3.10)$$

Denklem (3.9) ve (3.10) da bulunan t algoritmadaki iterasyon sayısını belirtmektedir. A ve C katsayı vektörlerini, $X_p$avın konumunu gösteren vektör, X ise bir gri kurdun konumu gösteren bir vektördür. A ve C vektörleri aşağıdaki formüllere göre hesaplanmaktadır.

$$A = 2\,\alpha.r_1 - \alpha \quad (3.11)$$

$$C = 2.r_2 \quad (3.12)$$

Denklem (3.11)'de bulunan α katsayısı algoritmanın başlangıcında 2 olarak belirlenmiştir. Daha sonrasına iterasyonlar boyunca lineer olarak azalır ve 0'a ulaşır. r_1 ve r_2 değerleri ise [0,1] aralığında rastgele değerler ile oluşturulur ve gri kutların avın etrafında rastgele arama yapmasını sağlar. Denklem (3.12)'de gri kurtlar avın konumuna göre arama uzayında konumlarını güncellemektedirler. Şekil 3.4.'te gri kurtların 2 ve 3 boyutlu uzayda arama yaparken gidebilecekleri bir sonraki konumlar gösterilmektedir.



Şekil 3.4 Gri kurtların 2 ve 3 boyutlu uzayda muhtemel sonraki konumları (Mirjalili, 2014)

3.3.3 Avlanma

Gri kurtlar avlarının konumunu tahmin edebilir ve avı kuşatabilirler. Avlanma genellikle alfa kurt tarafından yönetilir. Fakat bazen beta ve delta kurtlar da avın yönetilmesine katılabilir. Başlangıçta avın arama uzayında hangi konumda olduğu hakkında bilgi bulunmamaktadır. Gri kurtların avlanmasını matematiksel olarak ifade etmek için alfa, beta ve delta kurtların avın konumunu daha iyi bildiği kabul edilir. Bu nedenle şimdiye kadar elde edilen ilk üç arama ajanının (gri kurt) çözümleri kaydedilip diğer tüm arama ajanlarının konumları bu en iyi çözümlere göre güncellenmektedir. Diğer arama ajanların konum güncellemesi aşağıdaki formüllere göre yapılmaktadır.

$$D_\alpha = |C_1 . X_\alpha - X| \tag{3.13}$$

$$D_\beta = |C_2 . X_\beta - X| \tag{3.14}$$

$$D_\delta = |C_3 . X_\delta - X| \tag{3.15}$$

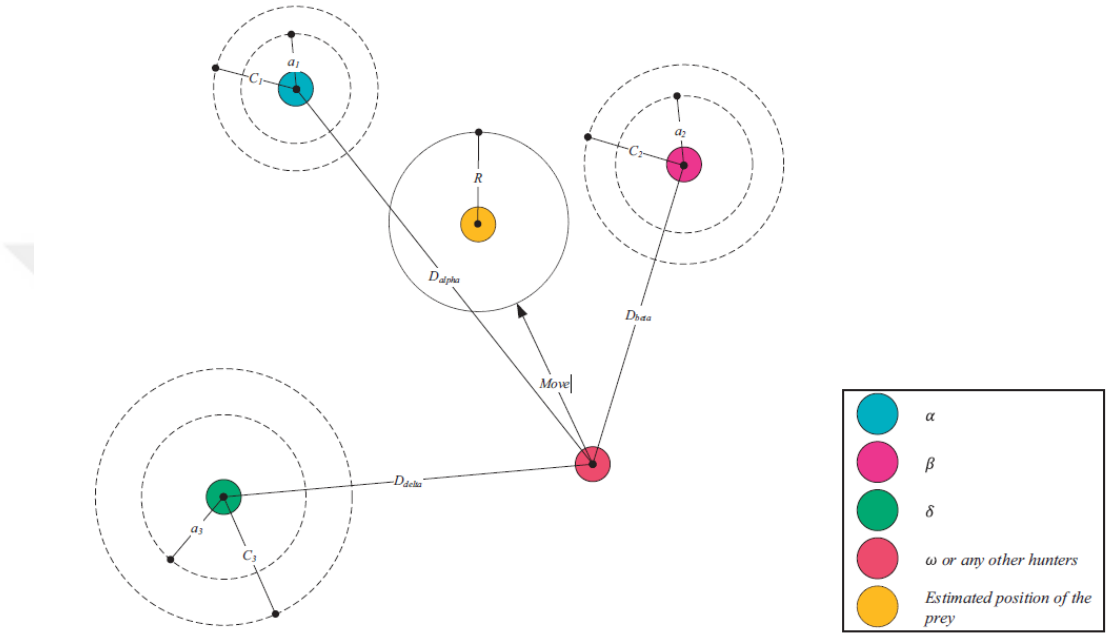
$$X_1 = |X_\alpha - A_1 . D_\alpha| \tag{3.16}$$

$$X_2 = |X_\beta - A_2 . D_\beta| \tag{3.17}$$

$$X_3 = |X_\delta - A_3 . D_\delta| \tag{3.18}$$

$$X_{(t+1)} = \frac{(X_1 + X_2 + X_3)}{3} \tag{3.19}$$

Şekil 3.5.'de bir arama ajanının alfa, beta ve delta kurtlara göre konumunu nasıl güncellediği gösterilmiştir. Arama ajanının son konumu alfa, beta, delta tarafından belirlenen çemberin içerisindeki rastgele bir pozisyon olacaktır. Başka bir deyişle alfa, beta, delta kurtları avın konumunu tahmin eder diğer arama ajanları da avın etrafında rastgele olarak konumlarını güncellerler.



Şekil 3.5 GKO algoritmasında pozisyon güncellemesi (Mirjalili, 2014)

3.3.4 Ava saldırma

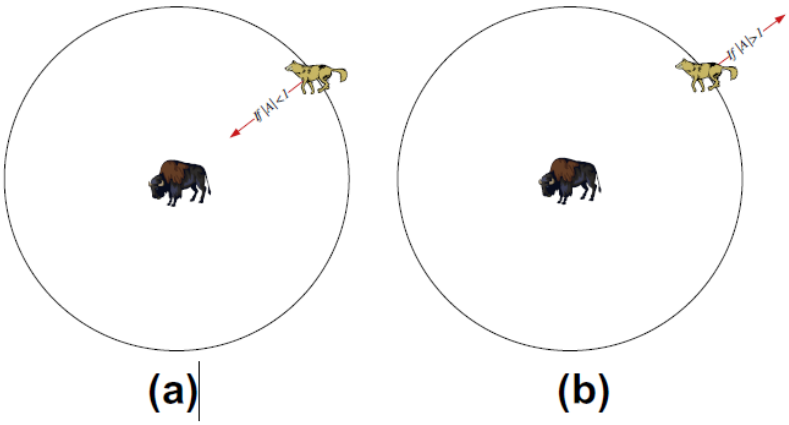
Av hareket etmeyi bıraktığı zaman gri kurtlar ava saldırarak avlanmayı sonlandırırlar. Ava yaklaşmayı matematiksel olarak ifade edebilmek için her iterasyonda a katsayısı azaltılır. A'nın dalgalanma aralığı α katsayısı ile azalır. Dolayısıyla A, $[2\alpha, -2\alpha]$ aralığında rastgele bir değer alır ve iterasyonlar boyunca 2'den 0'a düşürülür. A'nın değeri [-1, 1] aralığında olduğu durumda arama ajanının bir sonraki konumu arama ajanın şu anki konumu ile av arasında herhangi bir nokta olabilir. $|A| < 1$ olduğu durumda gri kurtlar ava saldırmaya zorlanmış olur.

Gri kurt optimizasyon algoritmasında her bir gri kurt, yukarıda bahsedilen mekanizmalar yardımı ile Alfa, Beta, Delta kurtların konumlarını baz alarak kendi

konumlarını günceller ve ava saldırma aşamasına gelir. Fakat bahsedilen bu adımlar ile GKO algoritması yerel çözümlere takılması sebebiyle zayıf kalmaktadır. Önerilen kuşatma mekanizması ile algoritma bir noktaya kadar keşif yapabilir fakat GKO'nun çözüm uzayını keşfetmesi için daha farklı mekanizmalara ihtiyaç vardır.

3.3.5 Av arama (keşif)

Gri kurtlar çoğunlukla alfa, beta, delta kurtların konumlarını dikkate alarak avlarını ararlar. Av arama aşamasında birbirlerinden ayrılırlar ve ava saldırmak için tekrar bir araya gelirler. Birbirlerinden uzaklaşma mekanizmasını modellemek için A katsayısı 1'den büyük veya -1'den küçük bir değer seçilerek kullanılır. Bu arama ajanın global olarak keşif yapmasına olanak tanır. Şekil 3.6.'da $|A| > 1$ olduğu durumda arama ajanın daha iyi bir av bulmak amacıyla avdan uzaklaştığı görülmektedir.



Şekil 3.6 Ava saldırma veya av arama (Mirjalili, 2014)

C, GKO'daki keşif aşamasını etkileyen başka bir bileşen faktörüdür. C katsayısı [0, 2] aralığında rastgele değerler içerir. Denklem (3.12)'de mesafenin hesaplanmasında avın etkisini arttırmak için (C>1) veya etkisini azaltmak için (C<1) durumunda av için rastgele ağırlıklar oluşturur. Böylece GKO'nun optimizasyon boyunca daha fazla rastgele davranışlar göstermesine yardımcı olur ve yerel optimumdan kaçınmayı sağlar. Burada C, A gibi her iterasyonda lineer olarak azalmaz, keşif yeteneğini sürekli olarak vurgulamak için C her zaman rastgele değerler alır. Bu

bileşen özellikle son iterasyonlarda yerel optimuma takılmayı engellemek açısından önemlidir.

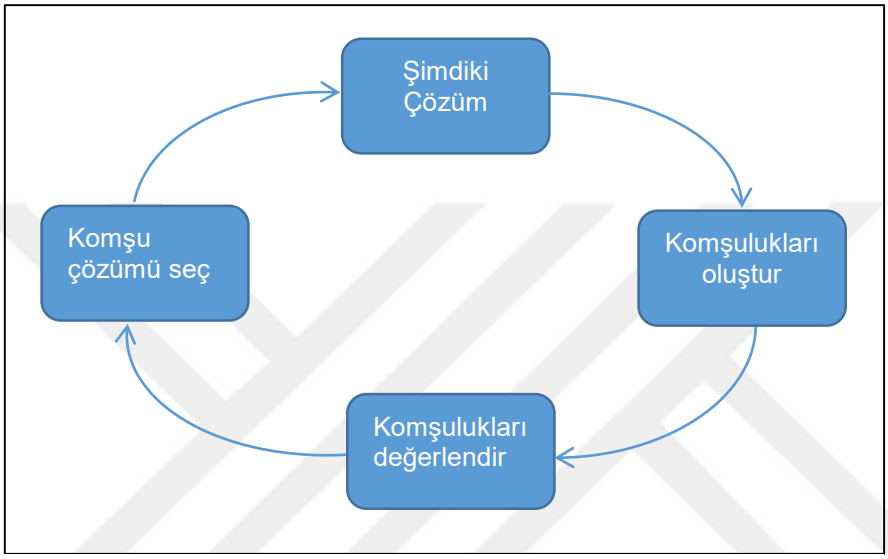
```
Gri kurtların başlangıç pozisyonlarını başlat
Her gri kurt için maliyet değerlerini hesapla
En iyi çözümü alfa kurt olarak kaydet
İkinci en iyi çözümü beta kurt olarak kaydet
Üçüncü en iyi çözümü delta kurt olarak kaydet
while(k < maksimum iterasyon & durdurma kriteri)
    α yı azalt
    for (her gri kurt için)
        Alfa, beta, delta katsayı vektörlerini oluştur
        Mesafe vektörlerini hesapla
        Deneme vektörlerini hesapla
        Gri kurdun konumunu güncelle
     end for
     Güncellenen gri kurtların maliyet değerlerini hesapla
    for (her gri kurt için)
        if (gri kurt < alfa kurt)
            alfa kurdu güncelle
        else if (gri kurt < beta kurt)
            beta kurdu güncelle
        else if (gri kurt < delta kurt)
delta kurdu güncelle
        end if
    end for
end while
```

Şekil 3.7 GKO algoritması sözde kodu

3.4 Yerel Arama Algoritması

Açgözlü algoritmalarda, çözüm adım adım yapılır. Algoritmanın ara adımlarında tam çözümler bulunmaz ve algoritma erken sonlandırılamaz. Algoritma iterasyonların ortasında durdurulduğu zaman elimizde eksiksiz bir çözüm olmaz. Sezgisel bir yöntem olan yerel arama algoritması başlangıçta eksiksiz bir çözüm ile başladığı için

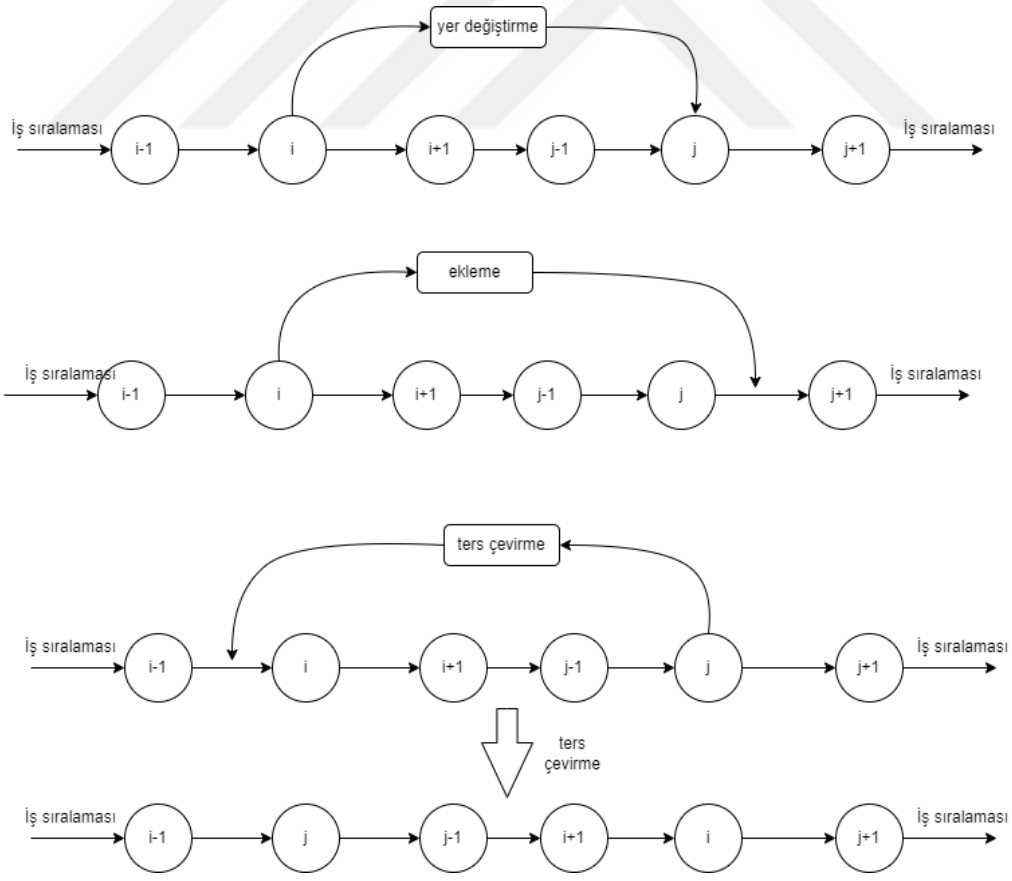
algoritmanın herhangi bir zaman aralığında sonlanması çözümü etkilemeyecektir. Her iterasyonda çözümü iyileştirmek için ufak değişiklikler yapılır. Algoritmanın herhangi bir adımında erken sonlanması durumunda bir çözüm elde edilecektir. Ancak elde edilen çözüm optimal çözüm olmayabilir. Yerel arama algoritmasında iterasyon sayısı hesaplama gücüne göre ayarlanabilir. Daha fazla iterasyonda optimal değere ulaşma ihtimali artacaktır.



Şekil 3.8 Yerel arama algoritması çalışma adımları

Literatürde birçok hibrit metasezgisel algoritma için yerel arama algoritmalarının birlikte kullanıldığı görülmektedir. Zhao ve ark. (2019), BATÇ problemler makespan amaç fonksiyonlu çözümü için faktöriyel tabanlı Parçacık Sürü Optimizasyon algoritması geliştirmişlerdir. Mevcut en iyi çözüm etrafında yerel arama yapmak için ekleme ve yer değiştirme komşulukları temelli Değişken Komşuluk Arama algoritması kullanmışlardır. Pan ve ark. (2008), BATÇ problemlerinin iki amaçlı çözümü için Ayrık Parçacık Sürü Algoritması önermişlerdir. Önerilen algoritmanın performansını iyileştirmek için Değişken komşuluk iniş algoritması kullanılmıştır. Qian ve ark. (2009), BATÇ problemlerinin makespan amaçlı çözümü için Hibrit Diferansiyel Evrim Algoritması (HDE) önermişlerdir. Algoritmanın yerel aramadaki sömürü yeteneğini arttırmak için Değişken Komşuluk Arama (DKA) algoritması uygulanmıştır.

Bu çalışmada GKO algoritmasının sömürü yeteneğini arttırmak amacıyla Hansen ve Mladenovic (2001), tarafından geliştirilen DKA algoritması kullanılmıştır. DKA bir çözüm üzerinde komşuları değiştirerek bir çözümden yeni bir çözüm üreten bir yerel arama algoritmasıdır. DKA algoritmasında ekleme, yer değiştirme, takas, ters alma, 2-opt gibi birden fazla komşuluk çeşidi önerilmiştir. Bunlar içerisinde en çok kullanılan ekleme ve yer değiştirme mekanizmalarının diğerlerinden daha etkili olduğu bilinmektedir (Zhao ve ark. 2017). DKA algoritmasında arama uzayında çözümler aranırken algoritmanın yerel optimum değerlere takılmasını engellemek için komşuluk değişimi kullanılır. İş sıralamasından rastgele r1 ve r2 şeklinde iki konum seçilir. Eğer r1 < r2 ise r1 işi r2'nin arkasına eklenir. Diğer durumda ise r2 işi r1 işinin önüne eklenir. Bu şekilde elde edilen sıralamalardan optimum çözüm değerini veren sıralama yeni çözüm olarak kabul edilir. Yerel arama algoritmasına ait yer değiştirme, ekleme, ters çevirme mekanizmaları Şekil 3.9'da gösterilmiştir. DKA algoritmasının sözde kod Şekil 3.10.'da gösterilmiştir.



Şekil 3.9 DKA algoritmasında ekleme ve yer değiştirme adımları(Li ve Yin, 2013)

Yer değiştirme: Bir çözümde bulunan iş çizelgesi üzerinde iki rastgele konum seç. Bu konumlarda bulunan işlerin yerlerini değiştir.

Ekleme: İş çizelgesi üzerinde iki rastgele konum seç ve geride olan işi önde olan işin önüne ekle.

Ters çevirme : İki rastgele konum arasında bulunan işlerin sırasını ters çevir. (Li ve Yin, 2013)

Başlangıç

Aramada kullanılacak N_k, Komşuluk yapısını seç (k=1, … ,kmax)

Bir başlangıç çözümü üret π

Bir durma kriteri belirle

Durma kriteri sağlanıncaya kadar aşağıdaki adımları tekrarla

(1) k = 1 olarak ata
(2) Whlile (k <= kmax) aşağıdaki adımları devam ettir
   a) Sallama: k. komşuluğundan, rastegele bir π' çözümü üret (π' € $N_k(\pi)$)
   b) Yerel arama: π' çözümüne yerel arama uygula ve elde edilen en iyi çözümü π'' olarak seç
   c) Değiştirme: Eğer üretilen π'' yerel optimum değeri daha iyi ise $\pi = \pi''$ yap ve N_1 (k=1) ile yeni komşulukları aramaya devam et. Diğer durumda, k = k+1 yap

Şekil 3.10 DKA algoritması sözde kodu

3.5 Hibrit Gri Kurt Optimizasyon Algoritması

Gri kurt optimizasyon algoritması literatürde çizelgeleme problemlerinde sıklıkla kullanılmaktadır. Enerji üretim tahminleri(Makhadmeh ve ark. 2021), servis çizelgeleme (Peng ve Zhou, 2019) vb. birçok alanda kullanıldığı görülmektedir. Son yıllarda optimizasyon problemlerinde tercih edilen GKO diğer metasezgisel algoritmalara göre rekabetçi bir performans gösterdiği görülmektedir (Li ve Feng, 2021). GKO algoritmasının birçok karmaşık problemleri çözmedeki performansı,

birçok alan kolayca uygulanabilir olması, arama uzayını tararken keşif ve sömürü yeteneğinin gelişmiş olması bu çalışmada tercih edilme nedeni olmuştur.

GKO algoritmasının performansını iyileştirmek için birçok algoritma ile hibrit şekilde kullanımları bulunmaktadır. Önerilen yöntemde, GKO algoritması ve DKA algoritmalarının hibrit şekilde kullanılmasından oluşmaktadır. Ayrıca algoritmanın çeşitli yönlerden performansını iyileştirmek için başlangıç popülasyonunu oluşturma aşamasında NEH ve Sıralı İşlem Süresi (SİS) yöntemleri kullanılmıştır. HGKO algoritması akış diyagramı Şekil 3.11'de gösterilmiştir.

3.5.1 Başlangıç popülasyonu

Metasezgisel algoritmalarda başlangıç popülasyonu genellikle arama uzayında rastgele değerler ile oluşturulur. Fakat başlangıç popülasyonun rastgele oluşturulması optimum çözüme ulaşmayı yavaşlatabilir. Literatürde yapılan çalışmalarda, başlangıç değerlerini rastgele belirlemek yerine bilinen iyi değerler ile başlatılması optimum çözüme yaklaşmayı hızlandırdığı görülmektedir. Kaya ve ark. (2020), yaptıkları çalışmada akış tipi çizelgeleme problemlerinin Hibrit Ateş Böceği ve Parçacık Sürü Algoritması ile çözümünde başlangıç popülasyonu seçiminin etkisini göstermişlerdir. Başlangıç popülasyonuna NEH ve SİS kurallarına göre üretilen çözümlerin eklenmesinin algoritmanın performansını arttırdığını ortaya koymuşlardır.

Çizelge 3.3 HGKO başlangıç popülasyonu optimizasyonu için kullanılan parametreler

Algoritma	Yüzdelik kullanımları					
Rastgele	100	98	90	80	70	60
NEH	0	1	5	10	20	30
SİS	0	1	5	10	20	30

Bu çalışmada başlangıç popülasyonun oluşturulması aşamasında parçacıkların arama uzayında rastgele oluşturulması yanında NEH ve SİS yöntemleri kullanılmıştır. Başlangıç popülasyonunda parçacıkların yüzde kaçının bu yöntemler ile üretilmesi gerektiğini tespit etmek amacıyla Çizelge 3.3'de görüldüğü üzere bir deneysel çalışma yapılmıştır. Yapılan deneysel çalışmalar sonucunda başlangıç popülasyonunun %10'u

NEH algoritması, %10'u SİS yöntemi ve %80'i ise arama uzayı aralığında rastgele değerler ile üretilmesine karar verilmiştir.

3.5.2 NEH algoritması

NEH algoritması Nawaz, Enscore, Ham (1983) tarafından permütasyon akış tipi çizelgeleme problemlerinin çözümü için geliştirilen sezgisel bir algoritmadır. NEH algoritmasında temel düşünce bir işin tüm makinelerdeki işlem sürelerinin toplamı en büyük olan iş daha önceliklidir. NEH algoritmasının adımları aşağıdaki gibidir;

Adım1: İşlerin makinelerdeki toplam işlem sürelerine bakılarak azalan bir sıralama elde edilir. Elde edilen işlerin sıralaması $\pi(j) = [\pi(1), \pi(2), \pi(3), \dots, \pi(n)]$ şeklinde olacaktır.

Adım2: $\pi(1)$ ve $\pi(2)$ sıralamadaki ilk iki iş alınır ve işlerin yer değiştirmesi ile elde edilecek sıralamalar oluşturulur. Daha sonra hangi sıralamanın tamamlanma süresi daha düşük ise o sıralama seçilir. Örneğin $\pi(1)$ - $\pi(2)$ sıralamasının tamamlanma süresi 28 saniye ve $\pi(2)$ - $\pi(1)$ sırlamasının tamamlanma süresi 25 saniye ise $\pi(2)$ - $\pi(1)$ sıralaması seçilerek kısmi bir çizelge elde edilir.

Adım3: $\pi(3)$ için daha önceki adımda elde edilen kısmi sıralamadaki mümkün olan tüm pozisyonlara yerleştirilir ve tamamlanma süresi en düşük olan çizelge alınır. Daha sonra sırasıyla $\pi(j)$ $j = 4, 5$ ... alınarak tüm işler sıralanıncaya kadar devam ettirilir.

Çizelge 3.3.'te GKO algoritmasının başlangıç popülasyonu ve yerel arama algoritmalarının eklenmesiyle OGS değerlerinde iyileşme gözlemlenmiştir. Birinci aşamada, GKO algoritmasında başlangıç popülasyonu rastgele oluşturulmuş ve problemler için ortalama sapma değeri hesaplanmıştır. İkinci aşamada, sırasıyla başlangıç popülasyonu NEH ve SİS algoritmaları ile iyileştirilmiş ve yerel arama eklendiğinde algoritmanın performansı ölçülmüştür. Üçüncü aşamada, hem başlangıç popülasyonu iyileştirilmiş hem de yerel arama algoritması eklenerek algoritmanın performansı test edilmiştir. Algoritmada başlangıç popülasyonunun iyileştirilmesiyle

ortalama sapma değeri 2.3837'den, 2.2067 ye düştüğü gözlemlenmiştir. Algoritmaya sadece DKA yerel arama algoritması eklendiğinde OGS değeri 0.4529'a düşmüştür. GKO algoritmasında başlangıç popülasyonunun iyileştirilmesi ve yerel arama algoritmasının uygulanması sonucunda OGS değeri 0.3632'a düştüğü gözlemlenmiştir.

Çizelge 3.4 Başlangıç popülasyonu ve yerel aramanın etkisi

Algoritma	OGS_{ort}
GKO	2.3837
GKO + NEH + SİS	2.2067
GKO + DKA	0.4529
GKO + NEH + SİS + DKA	0.3632

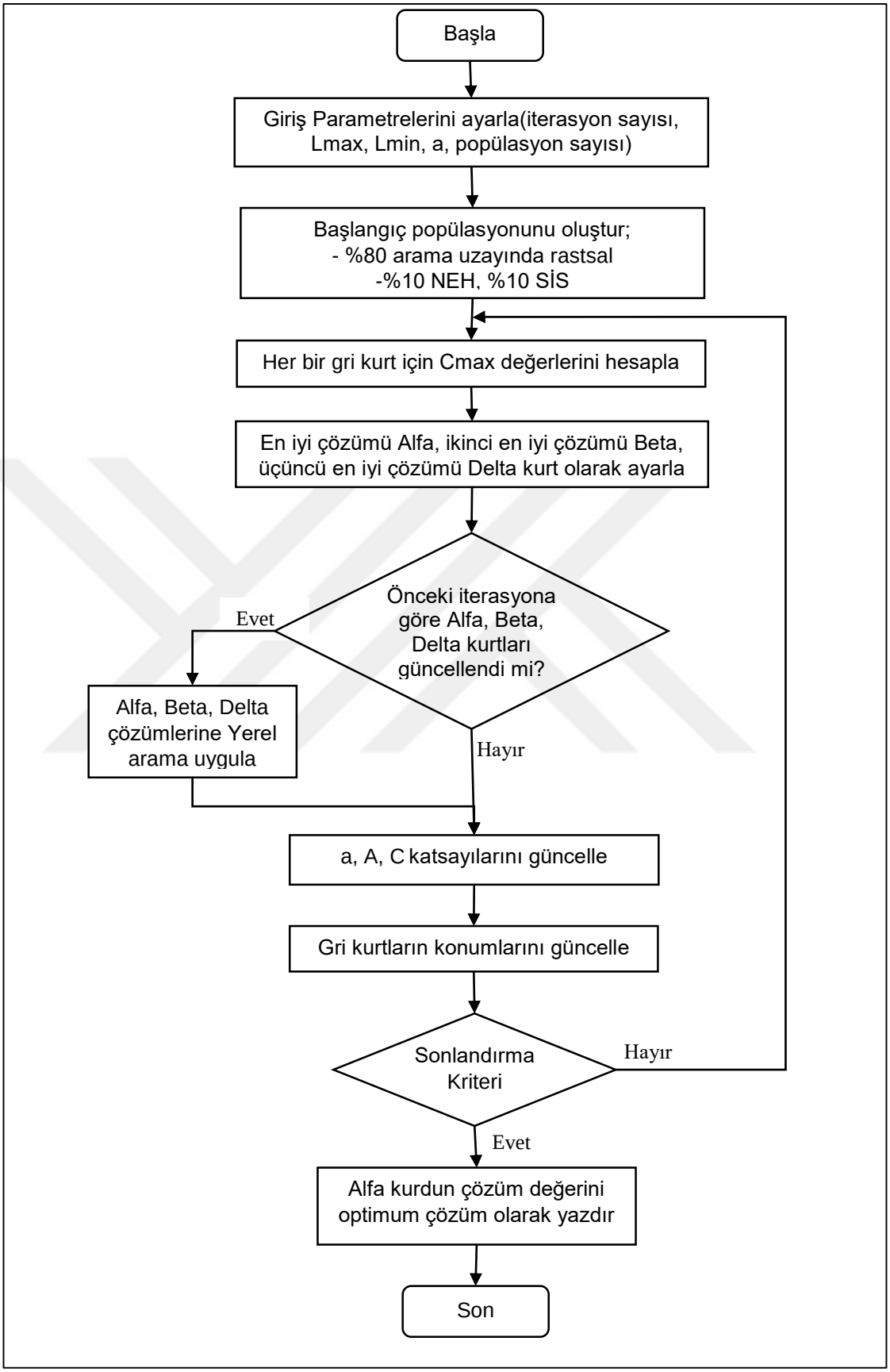
HGKO algoritmasında kullanılan parametre değerleri, yapılan deneysel çalışmalar sonucunda Çizelge 3.4.'daki gibi belirlenmiştir.

Çizelge 3.5 HGKO için kullanılanı parametre ayarları

Parametre	Katsayı
Popülasyon Sayısı	100
Maksimum iterasyon sayısı	1000
a	2
r1, r2	[0,1] aralığında rastgele
Arama uzayı aralığı Lmin, Lmax	-100, 100

3.5.3 Sıralı İşlem Süresi (SİS)

SİS işlerin tüm makinelerdeki işlem süreleri toplamının sıralanmış hali ile elde edilen bir çözümdür. SİS, En büyük işlem zamanı (EBİZ) ve En küçük işlem zamanı (EKİZ) sıralama kurallarından oluşmaktadır. EBİZ işlerin tüm makinelerdeki toplam işlem sürelerine göre azalan şekilde sıralanması ile elde edilen bir çözümdür. EKİZ ise işlerin tüm makinelerdeki toplam işlem sürelerine göre artan şekilde sıralanması ile elde edilen bir çözümdür. HGKO'nun başlangıç popülasyonunda %10 SİS yöntemi ile elde edilen çözümler eklenerek başlangıç popülasyonu çeşitlendirilmiş ve algoritmanın optimum çözüme ulaşması hızlandırılmıştır.



Şekil 3.11 HGKO algoritması akış diyagramı

3.6 Karşılaştırılan Algoritmalar

Geliştirilen algoritmanın performansı literatürde BATÇ problemleri için daha önceden yapılmış çalışmalar ile kıyaslanmıştır. Ayrıca problem seti için optimum çözümleri hesaplamak amacıyla Cplex programlama ortamında MILP ile algoritmanın kodları yazılmış ve test edilmiştir. Cplex ile veri setindeki küçük boyutlu 72 problem için optimum değerler hesaplanmıştır Problem setinde çok büyük boyutlu problemler için ise kısıtlı bir süre içerisinde çözüm bulunamamıştır. Büyük boyutlu 120 problem için programın çalışma süresi 3600'sn olarak sınırlandırılmış ve bu süre içerisinde elde edilen en iyi çözümler alınmıştır.

3.6.1 Uyarlanabilir öğrenme yaklaşımı

Engin ve Günaydın yaptıkları bu çalışmada BATÇ problemleri için UÖY algoritmasını kullanmışlardır. Hazırlık zamanlı BATÇ problemleri için 192 problemden oluşan bir veri seti hazırlamışlardır ve UÖY algoritmasını bu problem seti üzerinde test etmişlerdir. Önermiş oldukları algoritmayı 192 problem üzerinde test edip algoritmanın performansını Gen2 algoritması ile karşılaştırmışlardır (Engin ve Günaydın, 2011).

3.6.2 Genetik sezgisel algoritma

BATÇ problemlerini makespan kısıtı ile çözmek için benzetimli tavlama ve Genetik algoritma temelli 6 algoritma önermişlerdir. Benzetimli tavlama algoritma temelli SA, SA-1, SA-2 algoritmalarını geliştirmişlerdir. Genetik algoritmayı temel alan GA, GA-1, GA-2 algoritmalarını geliştirmişlerdir. Deneysel çalışmalar sonucunda SA-2 ve GA-2 algoritmalarının daha başarılı sonuçlar ürettiği gözlemlenmiştir (Aldowaisan ve Allahverdi, 2003).

3.6.3 Karınca koloni optimizasyon algoritması

Karınca koloni optimizasyon algoritmasında, genetik algoritmada bulunan çaprazlama ve mutasyon mekanizmalarını dahil ederek hibrit bir algoritma

geliştirmişlerdir. Geliştirdikleri hibrit karınca koloni algoritmasını 192 adet benchmark problem setinin makespan kriterli çözümünü sağlamak için kullanmışlardır. Karınca koloni algoritmasında karınca sayısı kadar feromon miktarı değişmektedir. Elde edilen sonuçlara göre önerdikleri algoritmanın karşılaştırılan algoritmalara göre daha başarılı sonuçlar ortaya koymuşlardır (Engin ve Güçlü, 2018).

3.7 Kullanılan Problem Seti

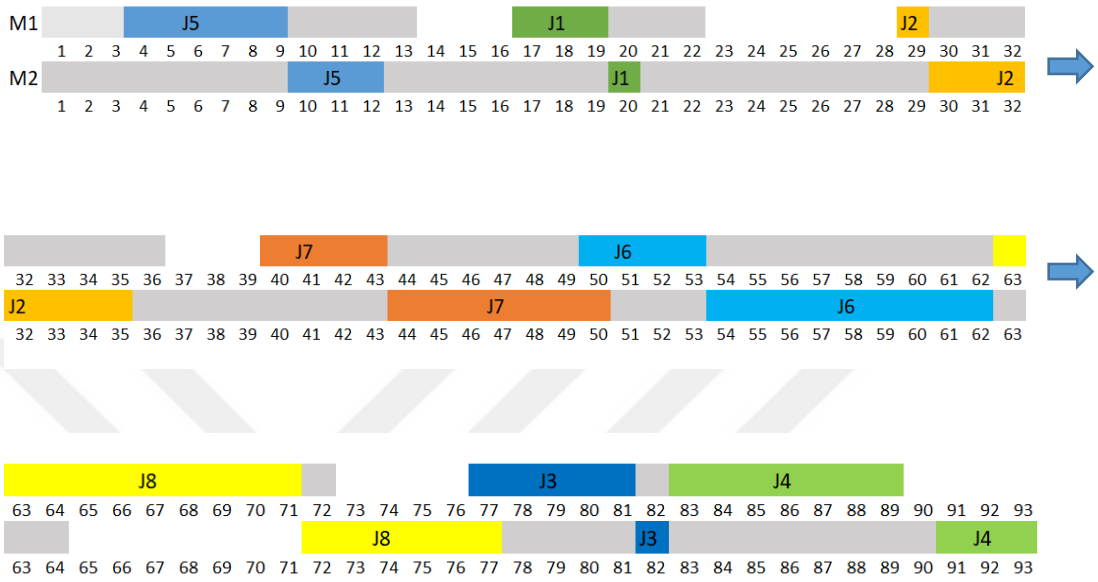
HGKO algoritması, literatürde kullanılan 192 BATÇ benchmark problemi üzerinde test edilmiştir. Problem setinde her makinedeki işlem süreleri, farklı aralıklardan, ayrık düzgün dağılımlı, rastgele değerlerden oluşturulmuştur. Problem setinde a tipi problemler [1, 10], b tipi problemler [1, 50] ve c tipi problemler [1, 100] aralığında rastgele değerler ile oluşturulmuştur. Ayrıca işlerin her makinedeki hazırlık süreleri [1, 10] aralığında düzgün dağılımlı rastgele değerlerden oluşturulmuştur. İş sayılarına göre 8, 10, 12, 50, 100, 150, 200, 250 olmak üzere 8 tip problem bulunmaktadır. Makine sayılarına göre 2, 3, 5, 8, 10, 15, 20, 25 olmak üzere 8 problem tipi vardır. Örnek olarak 2 makine 8 iş için işlem süreleri Çizelge 3.5.'de verilmiştir.

Allahverdi ve ark. (1999), Aldowaisan (2001) BATÇ problemleri için hazırlık zamanın öneminden bahsetmişlerdir. Literatürde yapılan birçok çalışmada işlerin makinelerdeki hazırlık süreleri dikkate alınmamış veya işlem sürelerine eklenmiştir. Bu çalışmada ise işlerin hazırlık süreleri işlem sürelerinden ayrı olarak ele alınmış ve bu şekilde optimizasyonu sağlanmıştır.

Çizelge 3.6 HGKO için kullanılanı parametre ayarları

	J1	J2	J3	J4	J5	J6	J7	J8
Makine 1 işlem süresi	3	1	5	7	6	4	4	9
Makine 2 işlem süresi	1	6	1	3	3	9	7	6
Makine 1 hazırlık süresi	4	3	1	1	3	6	6	9
Makine 2 hazırlık süresi	7	9	4	8	9	3	8	2

Çizelge 3.6'da verilen 8 iş 2 makine problemi için optimum çizelge J5 , J1, J2, J7, J6, J8, J3, J4 şeklinde olacaktır. Örnek problem için Cmax değerinin hesaplanması Şekil 3.6.'da gösterilmiştir.



Şekil 3.6 8 iş 2 makine problemi Cmax hesaplanması

4. ARAŞTIRMA BULGULARI ve TARTIŞMA

Bu bölümde, önerilen HGKO algoritması literatürde kullanılan 192 kıyaslama problemi ile test edilmiştir. Kullanılan problem seti Engin ve Günaydın(2011), tarafından geliştirilmiştir. Ayrıca yazarlar UÖY algoritmasını geliştirerek problem seti üzerinde test etmek için kullanmışlardır.

Bu çalışma kapsamında geliştirilen algoritma literatürde problem seti üzerinde test edilen algoritmalar ile kıyaslanmıştır. Problem boyutu küçük ve matematiksel modelle çözülebilecek 72 problem Cplex ortamında yazılmış MILP yöntemi ile çözülmüştür. Önerilen uygulamada MILP yöntemi ile optimal değerler elde edilmiştir. Büyük boyutlu problemlerde matematiksel yöntemler yetersiz kaldığı için sezgisel algoritmalar ile ulaşılan Cmax değerleri ile kıyaslanmıştır. Algoritmaların ulaştıkları Cmax değerleri optimum değerlere göre Ortalama Göreli Sapma (OGS) fonksiyonu ile karşılaştırılmıştır. OGS değerlerinin hesaplanması aşağıda Denklem (4.1)'de verilmiştir.

$$OGS = \frac{Sezgisel(Cmax) - Optimum(Cmax)}{Optimum(Cmax)} x100 \quad (4.1)$$

Optimum değerlerin tam olarak belli olmadığı büyük boyutlu problemler için OGS değerleri algoritmalar içerisinde elde edilen en iyi değerlere göre kıyaslanmıştır.

$$OGS = \frac{Sezgisel(Cmax) - En\ iyi(Cmax)}{En\ iyi(Cmax)} x100 \quad (4.2)$$

HGKO algoritması ile elde edilen sonuçların tutarlı olduğunu test etmek için algoritma problem seti üzerinde bağımsız olarak 10 tekrar ile çalıştırılmıştır. 10 tekrar ile elde edilen Cmax değerlerinin ortalamaları ve OGS değerleri diğer algoritmalarla kıyaslanarak çizelgelerde verilmiştir. Ayrıca önerilen algoritmanın tüm problem tipleri

için tutarlı sonuçlar ortaya koyup koymadığını görebilmek için OGS değerlerine ait standart sapma değerleri hesaplanmış ve çizelgelerde belirtilmiştir.

Çizelge 4.1 İş sayısı 8 olan problemler için OGS değerleri

Problem	Cmax					OGS			
	CPLEX	Gen2	UÖY	HKKO	**HGKO**	Gen-2	UÖY	HKKO	**HGKO**
8x2-a	93	93	93	93	93	**0.0000**	**0.0000**	**0.0000**	**0.0000**
8x2-b	255	255	255	255	255	**0.0000**	**0.0000**	**0.0000**	**0.0000**
8x2-c	465	465	468	468	465	0.0000	0.6452	0.6452	**0.0000**
8x3-a	92	92	92	92	92	**0.0000**	**0.0000**	**0.0000**	**0.0000**
8x3-b	290	290	304	302	290	0.0000	4.8276	4.1379	**0.0000**
8x3-c	558	565	562	560	565	1.2545	0.7168	0.3584	1.2545
8x5-a	126	127	126	126	126	0.7937	**0.0000**	**0.0000**	**0.0000**
8x5-b	396	396	398	398	396	**0.0000**	0.5051	0.5051	**0.0000**
8x5-c	744	752	744	744	744	1.0753	**0.0000**	**0.0000**	**0.0000**
8x8-a	132	132	132	132	132	**0.0000**	**0.0000**	**0.0000**	**0.0000**
8x8-b	518	518	518	518	518	**0.0000**	**0.0000**	**0.0000**	**0.0000**
8x8-c	990	990	990	990	992	**0.0000**	**0.0000**	**0.0000**	0.2020
8x10-a	151	153	152	152	151	1.3245	0.6623	0.6623	**0.0000**
8x10-b	602	602	602	602	602	**0.0000**	**0.0000**	**0.0000**	**0.0000**
8x10-c	1162	1162	1169	1162	1162	**0.0000**	0.6024	**0.0000**	**0.0000**
8x15-a	193	193	193	193	193	**0.0000**	**0.0000**	**0.0000**	**0.0000**
8x15-b	773	777	777	775	773	0.5175	0.5175	0.2587	**0.0000**
8x15-c	1487	1487	1487	1487	1487	**0.0000**	**0.0000**	**0.0000**	**0.0000**
8x20-a	216	217	216	216	216	0.4630	**0.0000**	**0.0000**	**0.0000**
8x20-b	887	892	892	892	887	0.5637	0.5637	0.5637	**0.0000**
8x20-c	1742	1742	1742	1742	1742	**0.0000**	**0.0000**	**0.0000**	**0.0000**
8x25-a	249	249	249	249	249	**0.0000**	**0.0000**	**0.0000**	**0.0000**
8x25-b	1050	1078	1069	1062	1050	2.6667	1.8095	1.1429	**0.0000**
8x25-c	2051	2102	2051	2051	2051	2.4866	**0.0000**	**0.0000**	**0.0000**
					Ortalama	0.4644	0.4521	0.3448	**0.0607**
					S. Sapma	0.7811	1.0268	0.8647	**0.2576**

Çizelge 4.1.'de İş sayısının 8 olduğu problemler için HGKO algoritmasını ile elde edilen Cmax değerleri verilmiş ve Gen2, UÖY, HKKO algoritmaları ile elde edilen Cmax değerleri ile karşılaştırılmıştır. Algoritmaların ulaştığı sapma değerleri Cplex algoritması ile elde edilen optimum Cmax değerlerine göre hesaplanmış ve

Çizelge 4.1.'de verilmiştir. Çizelge 4.1.'e göre HGKO algoritması 24 problemden 22 tanesinde optimum sonuca ulaşmıştır. 8 iş problemleri için HGKO algoritması 0.0607 OGS değerine ulaşılırken Gen2 0.4644, UÖY algoritmaları ile sırasıyla 0.4521, HKKO 0.3448 OGS değerleri elde edilmiştir.

Çizelge 4.2 İş sayısı 10 olan problemler için OGS değerleri

Problem	Cmax					OGS			
	CPLEX	Gen2	UÖY	HKKO	**HGKO**	Gen-2	UÖY	HKKO	**HGKO**
10x2-a	110	110	110	110	110	**0.0000**	**0.0000**	**0.0000**	**0.0000**
10x2-b	337	338	344	342	337	0.2967	2.0772	1.4837	**0.0000**
10x2-c	581	593	584	583	581	2.0654	0.5164	0.3442	**0.0000**
10x3-a	117	119	120	118	117	1.7094	2.5641	0.8547	**0.0000**
10x3-b	362	379	367	362	362	4.6961	1.3812	**0.0000**	**0.0000**
10x3-c	667	693	688	672	667	3.8981	3.1484	0.7496	**0.0000**
10x5-a	138	142	140	138	138	2.8986	1.4493	**0.0000**	**0.0000**
10x5-b	462	494	513	501	462	6.9264	11.0390	8.4416	**0.0000**
10x5-c	873	912	904	894	873	4.4674	3.5510	2.4055	**0.0000**
10x8-a	169	171	171	170	170	1.1834	1.1834	**0.5917**	**0.5917**
10x8-b	571	618	623	614	571	8.2312	9.1068	7.5306	**0.0000**
10x8-c	1076	1180	1180	1166	1076	9.6654	9.6654	8.3643	**0.0000**
10x10-a	176	182	184	178	176	3.4091	4.5455	1.1364	**0.0000**
10x10-b	612	654	679	656	612	6.8627	10.9477	7.1895	**0.0000**
10x10-c	1159	1290	1228	1192	1159	11.302	5.9534	2.8473	**0.0000**
10x15-a	215	223	221	215	216	3.7209	2.7907	**0.0000**	0.4651
10x15-b	849	887	879	856	850	4.4759	3.5336	0.8245	**0.1178**
10x15-c	1632	1683	1707	1658	1632	3.1250	4.5956	1.5931	**0.0000**
10x20-a	257	264	262	260	257	2.7237	1.9455	1.1673	**0.0000**
10x20-b	1043	1090	1081	1062	1043	4.5062	3.6433	1.8217	**0.0000**
10x20-c	2025	2115	2097	2062	2035	4.4444	3.5556	1.8272	**0.4938**
10x25-a	275	285	284	278	275	3.6364	3.2727	1.0909	**0.0000**
10x25-b	1195	1250	1248	1215	1195	4.6025	4.4351	1.6736	**0.0000**
10x25-c	2342	2428	2389	2342	2346	3.6721	2.0068	**0.0000**	0.1708
					Ortalama	4.2458	4.0420	2.1681	**0.0766**
					S. Sapma	2.7270	3.1443	2.7310	**0.1760**

Çizelge 4.2'de iş sayısı 10 olan problemler için algoritmaların elde ettiği Cmax değerleri verilmiş olup Cplex algoritması ile elde edilen optimum değerler ile karşılaştırılmıştır. HGKO algoritması 24 problemden 19'unda optimum çözüme ulaşmıştır. Algoritmaların ulaştığı Cmax değerleri Cplex ile elde edilen optimum

değerlere göre ortalama yüzde sapma değerleri hesaplanmıştır. HGKO algoritması 0.0766 OGS değerine ulaşırken Gen2, UÖY, HKKO algoritmaları ile sırasıyla 4.2458, 4.0420, 2.1681 OGS değerleri elde edilmiştir.

Çizelge 4.3 İş sayısı 12 olan problemler için OGS değerleri

Problem	Cmax					OGS			
	CPLEX	Gen2	UÖY	HKKO	**HGKO**	Gen-2	UÖY	HKKO	**HGKO**
12x2-a	140	140	140	140	141	**0.0000**	**0.0000**	**0.0000**	0.7143
12x2-b	396	396	396	396	398	**0.0000**	**0.0000**	**0.0000**	0.5051
12x2-c	712	731	712	712	713	2.6685	**0.0000**	**0.0000**	0.1404
12x3-a	136	136	136	136	137	**0.0000**	**0.0000**	**0.0000**	0.7353
12x3-b	391	398	391	391	457	1.7903	**0.0000**	**0.0000**	16.879
12x3-c	845	863	888	863	845	2.1302	5.0888	2.1302	**0.0000**
12x5-a	165	172	176	172	165	4.2424	6.6667	4.2424	**0.0000**
12x5-b	544	586	568	568	544	7.7206	4.4118	4.4118	**0.0000**
12x5-c	1029	1086	1102	1088	1029	5.5394	7.0943	5.7337	**0.0000**
12x8-a	192	203	196	196	192	5.7292	2.0833	2.0833	**0.0000**
12x8-b	679	730	702	702	679	7.5110	3.3873	3.3873	**0.0000**
12x8-c	1286	1417	1292	1292	1286	10.186	0.4666	0.4666	**0.0000**
12x10-a	195	214	202	202	202	9.7436	**3.5897**	**3.5897**	**3.5897**
12x10-b	746	782	813	782	749	4.8257	8.9812	4.8257	**0.4021**
12x10-c	1427	1563	1474	1474	1467	9.5305	3.2936	3.2936	**2.8031**
12x15-a	242	265	252	252	242	9.5041	4.1322	4.1322	**0.0000**
12x15-b	927	1037	983	983	927	11.866	6.0410	6.0410	**0.0000**
12x15-c	1807	2024	1896	1896	1807	12.008	4.9253	4.9253	**0.0000**
12x20-a	278	282	296	282	278	1.4388	6.4748	1.4388	**0.0000**
12x20-b	1155	1218	1218	1218	1155	5.4545	5.4545	5.4545	**0.0000**
12x20-c	2280	2462	2314	2314	2280	7.9825	1.4912	1.4912	**0.0000**
12x25-a	314	337	328	328	314	7.3248	4.4586	4.4586	**0.0000**
12x25-b	1310	1437	1368	1368	1310	9.6947	4.4275	4.4275	**0.0000**
12x25-c	2567	2632	2824	2632	2567	2.5321	10.0117	2.5321	**0.0000**
					Ortalama	5.8094	3.8533	2.8777	**1.0737**
					S. Sapma	3.8459	2.9166	**2.0530**	3.4859

Çizelge 4.3.'te iş sayısı 12 olan problemler için algoritmaların elde ettiği Cmax değerleri verilmiş olup Cplex algoritması ile elde edilen optimum değerler ile karşılaştırılmıştır. HGKO algoritması 24 problemden 16'sında optimum çözümü

bulmuştur. Algoritmaların ulaştığı Cmax değerleri Cplex ile elde edilen optimum değerlere göre ortalama yüzde sapma değerleri hesaplanmıştır. HGKO algoritması 1.0737 OGS değerine ulaşırken Gen2, UÖY, HKKO algoritmaları ile sırasıyla 5.8094, 3.8533, 2.8777 OGS değerleri elde edilmiştir.

Çizelge 4.4 İş sayısı 50 olan problemler için OGS değerleri

Problem	OGS				
	CPLEX	Gen2	UÖY	HKKO	**HGKO**
50x2-a	**0.0000**	6.2847	5.7301	5.7301	**0.0000**
50x2-b	**0.0000**	11.0622	13.0735	11.0622	**0.0000**
50x2-c	0.0339	12.6230	11.9444	11.9444	**0.0000**
50x3-a	0.9042	8.8608	13.9241	8.8608	**0.0000**
50x3-b	0.4905	23.9730	16.6156	16.6156	**0.0000**
50x3-c	1.4532	27.5769	20.3785	20.3785	**0.0000**
50x5-a	0.6504	16.9106	14.7967	14.7967	**0.0000**
50x5-b	2.6231	29.7109	28.0514	28.0514	**0.0000**
50x5-c	2.1795	25.2760	30.9369	25.2760	**0.0000**
50x8-a	3.2353	19.7059	12.0588	12.0588	**0.0000**
50x8-b	5.3352	32.1933	23.3014	23.3014	**0.0000**
50x8-c	6.3727	36.1037	29.5857	29.5857	**0.0000**
50x10-a	3.4341	19.2308	11.6758	11.6758	**0.0000**
50x10-b	4.4916	31.2372	21.7640	21.7640	**0.0000**
50x10-c	7.4002	34.1146	26.4757	26.4757	**0.0000**
50x15-a	8.1437	11.6168	19.2814	11.6168	**0.0000**
50x15-b	7.7653	28.5521	20.7867	20.7867	**0.0000**
50x15-c	10.6716	20.5520	29.2018	20.5520	**0.0000**
50x20-a	10.0793	20.2718	12.3443	12.3443	**0.0000**
50x20-b	8.5241	28.0723	19.6386	19.6386	**0.0000**
50x20-c	11.2674	22.2847	30.5829	22.2847	**0.0000**
50x25-a	7.2403	22.3505	16.8940	16.8940	**0.0000**
50x25-b	8.9814	28.9430	18.4556	18.4556	**0.0000**
50x25-c	12.1208	28.0618	14.7331	14.7331	**0.0000**
Ortalama	5.1416	22.7320	19.2596	17.7035	**0.0000**
S.Sapma	4.0041	8.2649	7.0399	6.3669	**0.0000**

Çizelge 4.4.'te iş sayısının 50 olduğu problemler için algoritmaların elde ettiği Cmax değerleri verilmiştir. 50 iş ve daha büyük problemlerde Cplex algoritması ile

elde edilen değerler optimum olmadığından tüm algoritmalar ile elde edilen en küçük Cmax değeri optimum olarak kabul edilmiştir. Algoritmaların sapma değerleri tüm algoritmalar içerisindeki en küçük Cmax değerine göre hesaplanmıştır. HGKO algoritması 24 problemin tamamında minimum Cmax değerine ulaşmıştır. HGKO 0.0000 OGS değeri elde ederken, Cplex, Gen2, UÖY, HKKO algoritmaları ile sırasıyla 5.1416, 22.7320, 19.2596, 17.7035 OGS elde edilmiştir.

Çizelge 4.5 İş sayısı 100 olan problemler için OGS değerleri

Problem	OGS				
	CPLEX	Gen2	UÖY	HKKO	**HGKO**
100x2-a	**0.0000**	9.5797	8.5044	8.5044	**0.0000**
100x2-b	0.6995	18.8541	19.2871	18.8541	**0.0000**
100x2-c	0.7646	21.3545	22.3557	21.8278	**0.0000**
100x3-a	1.6529	11.6621	15.3352	11.6621	**0.0000**
100x3-b	1.6640	29.3760	23.6480	23.6480	**0.0000**
100x3-c	2.1903	32.1360	29.9807	29.9807	**0.0000**
100x5-a	7.0175	22.4728	17.9616	17.9616	**0.0000**
100x5-b	5.3485	36.2237	29.1734	29.1734	**0.0000**
100x5-c	6.9971	38.5860	29.1837	29.1837	**0.0000**
100x8-a	9.7744	25.1128	16.3910	16.3910	**0.0000**
100x8-b	12.5114	31.3351	38.1698	31.3351	**0.0000**
100x8-c	15.6767	41.5064	30.2878	30.2878	**0.0000**
100x10-a	12.8096	24.2746	14.5789	14.5789	**0.0000**
100x10-b	14.7538	37.1169	37.5819	25.9776	**0.0000**
100x10-c	12.2044	30.5391	40.4348	30.5391	**0.0000**
100x15-a	10.4127	25.6508	17.2698	17.2698	**0.0000**
100x15-b	29.6831	30.1706	41.4401	30.1706	**0.0000**
100x15-c	31.3647	42.0495	30.1408	30.1408	**0.0000**
100x20-a	25.4610	26.7103	16.0619	16.0619	**0.0000**
100x20-b	30.0831	40.5980	25.0498	25.0498	**0.0000**
100x20-c	35.2478	44.0360	26.2175	26.2175	**0.0000**
100x25-a	31.0325	21.0999	31.3692	21.0999	**0.0000**
100x25-b	45.1443	43.3253	28.7132	28.7132	**0.0000**
100x25-c	39.0568	45.7015	28.1058	28.1058	**0.0000**
Ortalama	15.8980	30.3946	25.7184	23.4473	**0.0000**
S. Sapma	13.8254	10.1123	8.8129	6.8094	**0.0000**

Çizelge 4.5.'te iş sayısının 100 olduğu problemler için algoritmaların elde ettiği Cmax değerleri verilmiştir. Tüm algoritmalar ile elde edilen en küçük Cmax değeri optimum olarak kabul edilmiş ve algoritmaların sapma değerleri en küçük Cmax değerine göre hesaplanmıştır. Çizelge 4.5.'de HGKO algoritması 24 problemin tamamında minimum Cmax değerine ulaşmıştır. HGKO 0.0000 OGS değeri elde ederken, Cplex, Gen2, UÖY ve HKKO algoritmaları ile sırasıyla 15.8980, 30.3946, 25.7184, 23.4473 OGS değerleri elde etmiştir.

Çizelge 4.6 İş sayısı 150 olan problemler için OGS değerleri

Problem	OGS				
	CPLEX	Gen2	UÖY	HKKO	**HGKO**
150x2-a	**0.0000**	9.3730	9.0498	9.0498	0.3232
150x2-b	**0.0000**	19.8994	20.3111	19.9680	1.3724
150x2-c	**0.0000**	21.2159	21.0422	21.0422	1.0794
150x3-a	1.1594	12.9855	11.6522	11.6522	**0.0000**
150x3-b	**0.0000**	24.0287	23.2915	23.2915	0.2391
150x3-c	**0.0000**	27.1596	25.6088	25.6088	1.3869
150x5-a	4.6599	13.6583	16.9791	13.6583	**0.0000**
150x5-b	0.3678	26.9141	22.4005	22.4005	**0.0000**
150x5-c	5.7429	27.4489	22.6617	22.6617	**0.0000**
150x8-a	6.1028	17.5877	9.7549	9.7549	**0.0000**
150x8-b	11.7990	25.8611	16.7948	16.7948	**0.0000**
150x8-c	5.8730	18.2675	27.8121	18.2675	**0.0000**
150x10-a	9.6875	17.5893	9.6429	9.6429	**0.0000**
150x10-b	15.0603	28.0790	16.4258	16.4258	**0.0000**
150x10-c	11.9992	30.0710	19.9638	19.9638	**0.0000**
150x15-a	17.3859	22.4481	9.3776	9.3776	**0.0000**
150x15-b	23.4479	32.8829	14.5836	14.5836	**0.0000**
150x15-c	31.9055	35.2676	14.2413	14.2413	**0.0000**
150x20-a	27.9801	22.7290	9.8888	9.8888	**0.0000**
150x20-b	24.8719	20.9288	33.2392	20.9288	**0.0000**
150x20-c	24.8112	34.9650	21.3017	21.3017	**0.0000**
150x25-a	29.1248	23.9240	10.0430	10.0430	**0.0000**
150x25-b	32.8452	35.0418	19.4561	19.4561	**0.0000**
150x25-c	44.1762	15.3747	37.0985	15.3747	**0.0000**
Ortalama	13.7084	23.4875	18.4425	16.4741	**0.1834**
S. Sapma	13.1669	7.2194	7.6314	5.2208	**0.4337**

Çizelge 4.6.'te iş sayısının 150 olduğu problemler için algoritmaların elde ettiği Cmax değerleri verilmiştir. Tüm algoritmalar ile elde edilen en küçük Cmax değeri optimum olarak kabul edilmiş ve algoritmaların sapma değerleri en küçük Cmax değerine göre hesaplanmıştır. Çizelge 4.6.'da görüleceği üzere HGKO 0.1834 OGS değeri elde ederken, Cplex, Gen2, UÖY ve HKKO algoritmaları ile sırasıyla 13.7084, 23.4875, 18.4425, 16.4741 OGS değerleri elde edilmiştir.

Çizelge 4.7 İş sayısı 200 olan problemler için OGS değerleri

Problem	OGS				
	CPLEX	Gen2	UÖY	HKKO	**HGKO**
200x2-a	**0.0000**	10.2502	9.7209	9.7209	1.1550
200x2-b	**0.0000**	18.4410	19.0178	18.6058	3.1477
200x2-c	**0.0000**	20.1587	19.3921	19.3921	2.8231
200x3-a	**0.0000**	12.5883	12.6767	12.6767	0.7509
200x3-b	**0.0000**	21.7800	23.0430	21.7800	3.1723
200x3-c	**0.0000**	28.0496	27.0277	27.8243	6.3566
200x5-a	1.6949	12.5578	7.1649	7.1649	**0.0000**
200x5-b	1.3876	19.5209	8.7287	8.7287	**0.0000**
200x5-c	**0.0000**	12.5466	22.0307	12.5466	0.1705
200x8-a	7.5689	14.7890	9.0338	9.0338	**0.0000**
200x8-b	7.2104	20.4719	13.9225	13.9225	**0.0000**
200x8-c	5.6392	24.6319	18.5799	18.5799	**0.0000**
200x10-a	9.4048	4.1763	14.4689	4.1763	**0.0000**
200x10-b	8.7645	21.0536	9.4676	9.4676	**0.0000**
200x10-c	8.7440	10.5903	22.9678	10.5903	**0.0000**
200x15-a	11.8152	15.7536	3.3166	3.3166	**0.0000**
200x15-b	17.2875	23.6613	12.7150	12.7150	**0.0000**
200x15-c	20.2956	8.7922	24.8480	8.7922	**0.0000**
200x20-a	21.9762	5.8954	17.3263	5.8954	**0.0000**
200x20-b	30.3126	25.1468	7.6607	7.6607	**0.0000**
200x20-c	30.6091	8.6753	25.6027	8.6753	**0.0000**
200x25-a	19.6738	17.0234	5.1478	5.1478	**0.0000**
200x25-b	28.1849	22.7607	3.2769	3.2769	**0.0000**
200x25-c	32.1538	26.3205	6.0642	6.0642	**0.0000**
Ortalama	10.9468	16.9015	14.3000	11.0731	**0.7323**
S. Sapma	11.2598	6.7797	7.4750	6.2581	**1.5798**

Çizelge 4.7.'de iş sayısının 200 olduğu problemler için algoritmaların elde ettiği Cmax değerleri verilmiştir. Tüm algoritmalar ile elde edilen en küçük Cmax değeri optimum olarak kabul edilmiş ve algoritmaların sapma değerleri en küçük Cmax değerine göre hesaplanmıştır. Çizelge 4.7'ye göre HGKO algoritması 0.7323 OGS değeri elde ederken, Cplex, Gen2, UÖY ve HKKO algoritmaları ile sırasıyla 10.9468, 16.9015, 14.3000, 11.0731 OGS değerleri elde edilmiştir.

Çizelge 4.8 İş sayısı 250 olan problemler için OGS değerleri

Problem	OGS				
	CPLEX	Gen2	UÖY	HKKO	**HGKO**
250x2-a	**0.0000**	7.6555	8.2444	8.2444	0.6625
250x2-b	**0.0000**	17.9104	17.4043	17.4043	3.6210
250x2-c	**0.0000**	20.7989	19.4579	19.4579	5.3852
250x3-a	**0.0000**	8.9857	9.9387	8.9857	0.1361
250x3-b	**0.0000**	22.3358	21.4745	21.4745	6.0060
250x3-c	**0.0000**	25.8898	23.6129	23.6129	7.5186
250x5-a	4.2704	12.8111	6.5438	6.5438	**0.0000**
250x5-b	**0.0000**	17.9592	8.7608	8.7608	0.1769
250x5-c	**0.0000**	22.7741	13.9567	13.9567	4.1906
250x8-a	5.0177	12.1625	7.8811	7.8811	**0.0000**
250x8-b	3.4016	17.9700	12.3553	12.3553	**0.0000**
250x8-c	2.0396	18.2342	10.1001	10.1001	**0.0000**
250x10-a	9.3047	13.4202	1.8916	1.8916	**0.0000**
250x10-b	7.4906	18.8058	6.5543	6.5543	**0.0000**
250x10-c	4.3345	18.4712	6.3248	6.3248	**0.0000**
250x15-a	17.4219	16.1775	**0.0000**	**0.0000**	2.2775
250x15-b	22.1826	19.1974	**0.0000**	**0.0000**	1.1510
250x15-c	21.9569	3.9347	18.6224	3.9347	**0.0000**
250x20-a	14.2857	14.7769	3.8223	3.8223	**0.0000**
250x20-b	23.4221	19.4208	3.5813	3.5813	**0.0000**
250x20-c	21.4486	18.4932	3.1081	3.1081	**0.0000**
250x25-a	19.8916	15.9173	**0.0000**	**0.0000**	0.5620
250x25-b	24.5611	5.0058	19.1427	5.0058	**0.0000**
250x25-c	21.2977	20.1024	0.3794	0.3794	**0.0000**
Ortalama	9.2636	16.2171	9.2982	8.0575	**1.3203**
S. Sapma	9.5644	5.5144	7.4021	6.8723	**2.2627**

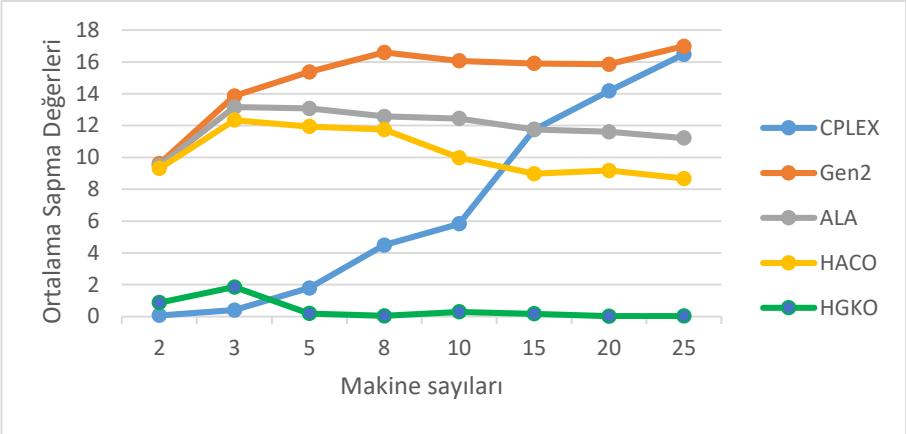
Çizelge 4.8.'de iş sayısının 250 olduğu problemler için algoritmaların elde ettiği Cmax değerleri verilmiştir. Tüm algoritmalar ile elde edilen en küçük Cmax değeri optimum olarak kabul edilmiş ve algoritmaların sapma değerleri en küçük Cmax değerine göre hesaplanmıştır. Çizelge 4.8.'e göre HGKO algoritması 1.3203 OGS değerine ulaşırken, Cplex, Gen2, UÖY ve HKKO algoritmaları ile 9.2636, 16.2171, 9.2982, 8.0575 OGS değerlerine ulaşabilmiştir.

Çizelge 4.9.'da problem setinde sırasıyla makine sayısının 2, 3, 5, 8, 10, 15, 20, 25 olduğu problemler için algoritmaların bulduğu en iyi çözüm sayıları verilmiştir. Burada Cplex algoritması iş sayısının 8, 10 ve 12 olan problemler için problem boyutu büyük olmadığından optimum çözümü bulabilmiştir. Diğer büyük boyutlu problemler için Cplex algoritması 3600 saniye çalıştırılarak ve toplamda 95 problemde en iyi çözümü elde edebilmiştir. Gen2 algoritması problem setindeki 17 problem için, UÖY 23 problem için, HKKO 28 problem için en iyi değerleri elde etmiştir. Önerilen HGKO algoritması problem setindeki toplam 154 problem için en iyi Cmax değeri elde etmiştir.

Çizelge 4.9 Önerilen HGKO performansının CPLEX, Gen2, UÖY ve HKKO ile karşılaştırılması

Problem	En iyi Cmax değerleri				
Makine Sayısı	CPLEX	Gen2	ALA	HKKO	**HGKO**
2	**21**	6	6	6	12
3	**17**	3	3	4	13
5	12	0	2	3	**21**
8	9	3	3	3	**22**
10	9	2	1	2	**21**
15	9	2	4	5	**20**
20	9	0	2	2	**23**
25	9	1	2	3	**22**
Toplam	95	17	23	28	**154**

Şekil 4.1'de algoritmaların makine sayılarına göre ortalama sapma değerleri verilmiştir.

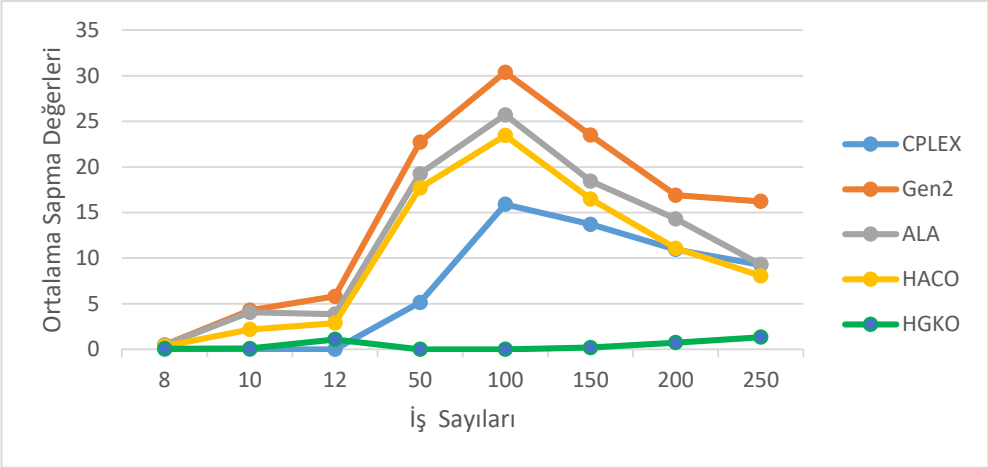


Şekil 4.1 Makine sayılarına göre algoritmaların OGS değerlerinin karşılaştırılması

Çizelge 4.10'da problem setinde sırasıyla makine sayısının 2, 3, 5, 8, 10, 15, 20, 25 olduğu problemler için algoritmaların bulduğu Cmax değerlerinin ortalama sapma miktarları hesaplanmıştır. Tüm problemler için ortalamada Cplex 6.8697, Gen2 15.0352, UÖY 11.9207, HKKO 10.2682 OGS değerine ulaşmıştır. Önerilen HGKO algoritması ise ortalamada 0.4308 OGS değerine ulaşmıştır. Şekil 4.2.'de karşılaştırılan algoritmaların iş sayılarına göre elde ettiği OGS değerleri grafiksel olarak verilmiştir. HGKO algoritmasının özellikle büyük boyutlu problemlerde diğer yöntemlere göre daha avantajlı olduğu gözlemlenmiştir.

Çizelge 4.10 Önerilen HGKO algoritmasının OGS değerlerinin Cplex, Gen2, UÖY, HKKO ile karşılaştırılması

Problem	OGS				
Makine Sayısı	Cplex	Gen2	UÖY	HKKO	**HGKO**
2	**0.0624**	9.6038	9.4906	9.3034	0.8720
3	**0.3964**	13.8694	13.1640	12.3451	1.8515
5	1.7891	15.3764	13.0853	11.9419	**0.1891**
8	4.4816	16.6017	12.5801	11.7531	**0.0331**
10	5.8285	16.0737	12.4498	9.9830	**0.2831**
15	11.7392	15.9136	11.7650	8.9697	**0.1671**
20	14.1825	15.8617	11.6106	9.1801	**0.0206**
25	16.4786	16.9820	11.2209	8.6698	**0.0305**
Ortalama	6.8697	15.0352	11.9207	10.2682	**0.4308**



Şekil 4.2 İş sayılarına göre algoritmaların OGS değerlerinin karşılaştırılması

Önerilen HGKO algoritmasının kaç problemde en iyi cmax değerine ulaştığını bulmak için Mean ranking score(ortalama sıralama puanları) değerleri hesaplanmıştır. Ortalama sıralama puanı bir algoritmanın bulduğu her bir çözümün karşılaştırılan diğer algoritmalara göre kaçıncı sırada olduğudur. Karşılaştırılan yöntem sayısı 5 olduğundan en iyi performans gösteren algoritmanın sıralama puanı 5 olarak belirlenir, daha sonra ikinci en iyi 4 bu şekilde azalarak en düşük performans gösteren algoritma 1 sıralama puanı alır. Çizelge 4.11'de karşılaştırılan algoritmaların sıralama puanlarının ortalaması gösterilmiştir. HGKO algoritması tüm problemler için ortalama 4.3854 sıralama puanı elde ederken Cplex 3.6761, HKKO 2.8542, UÖY 2.3542 ve Gen2 1.7292 sıralama puanı elde etmiştir.

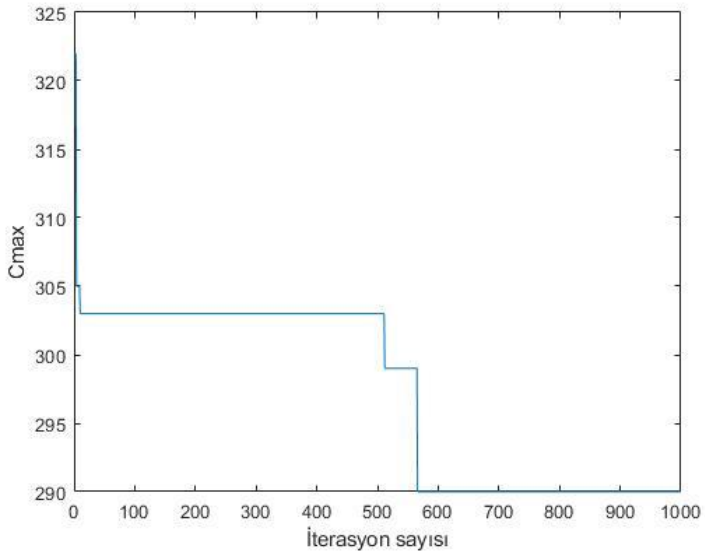
Çizelge 4.11 Önerilen HGKO algoritmasının ranking değerlerinin karşılaştırılması

Problem	Ortalama ranking				
Makine Sayısı	Cplex	Gen2	UÖY	HKKO	**HGKO**
2	**4.3125**	2.1250	2.2292	2.5625	3.7708
3	**4.3750**	1.8125	2.1458	2.7083	3.9583
5	4.1875	1.5417	2.1250	2.5833	**4.5625**
8	4.0417	1.4792	2.3125	2.6667	**4.5000**
10	3.8125	1.6458	2.1458	2.7917	**4.6042**
15	3.1250	1.7083	2.5417	3.1875	**4.4375**
20	2.8542	1.7708	2.6042	3.0833	**4.6875**
25	2.7083	1.7500	2.7292	3.2500	**4.5625**
Ortalama	**3.6771**	**1.7292**	**2.3542**	**2.8542**	**4.3854**

Çizelge 4.12'de çalışmada kullanılan veri setinde küçük boyutlu problemlerden 8 iş 3 makine b tipi problemin 10 defa çalıştırılmasına ait Cmax değerleri, amaç fonksiyonu çağırma sayısı ve algoritmanın işlem süreleri verilmiştir. HGKO algoritması probleme ortalama 4 saniyede çözüm üretmiştir. Ayrıca Cmax değerlerinin iterasyonlar boyunca değişimi Şekil 4.3'de verilmiştir.

Çizelge 4.12 8 iş 3 makine b problemi 10 defa çalıştırma sonuçları

Problem çalıştırma sayısı	Cmax	Amaç fonksiyonu çağırma sayısı	İşlem süresi (sn)
1	290	133512	5.66
2	290	132421	4.05
3	290	135027	4.52
4	290	134267	4.97
5	290	134314	3.7
6	298	132818	3.83
7	290	132941	3.8
8	290	133558	4
9	290	133950	4.19
10	290	136005	4.34



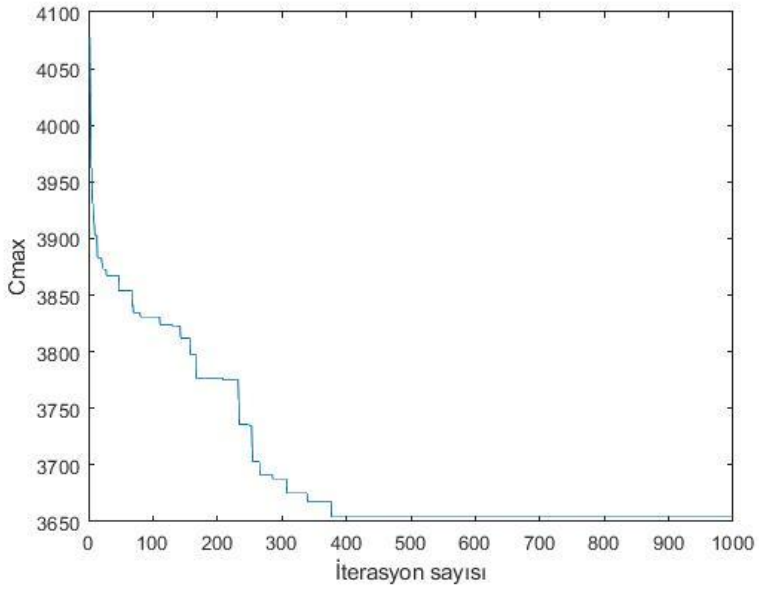
Şekil 4.3 8 iş 3 makine b problemi iterasyon sayısına göre cmax değerleri

Çizelge 4.13'de çalışmada kullanılan veri setinde orta boyutlu problemlerden 50 iş 25 makine b tipi problemin 10 defa çalıştırılmasına ait Cmax değerleri, amaç fonksiyonu çağırma sayısı ve algoritmanın işlem süreleri verilmiştir. HGKO

algoritması probleme ortalama 40 saniyede çözüm üretmiştir. Ayrıca Cmax değerlerinin iterasyonlar boyunca değişimi Şekil 4.4'de verilmiştir.

Çizelge 4.13 50 iş 25 makine b problemi 10 kez çalıştırılma sonuçları

Problem çalıştırma sayısı	Cmax	Amaç fonksiyonu çağırma sayısı	İşlem süresi (sn)
1	3737	390075	39.67
2	3696	429394	42.14
3	3674	392721	38.8
4	3721	406709	38.25
5	3716	405505	41.75
6	3701	402160	39.48
7	3706	401294	39.02
8	3738	401404	41.52
9	3682	409122	42.36
10	3654	403008	40.14

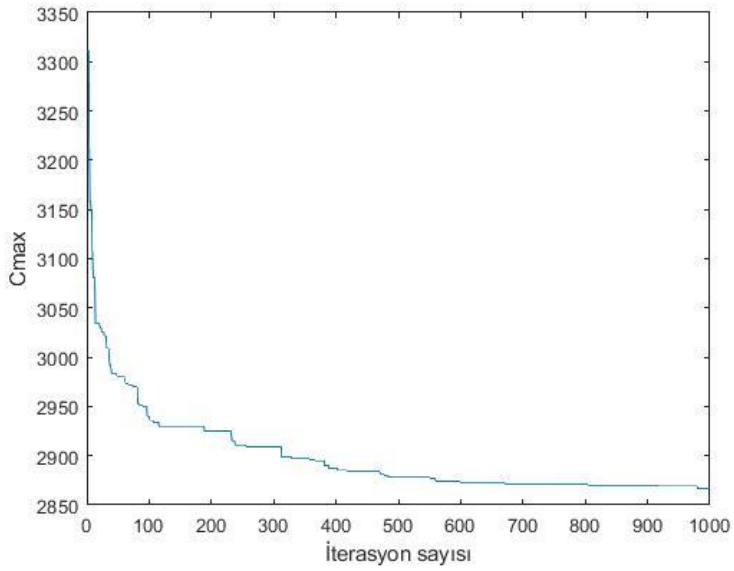


Şekil 4.4 50 iş 25 makine b problemi iterasyon sayısına göre cmax değerleri

Çizelge 4.14'de çalışmada kullanılan veri setinde büyük boyutlu problemlerden 200 iş 10 makine a tipi problemin 10 defa çalıştırılmasına ait Cmax değerleri, amaç fonksiyonu çağırma sayısı ve algoritmanın işlem süreleri verilmiştir. HGKO algoritması probleme ortalama 118 saniyede çözüm üretmiştir. Ayrıca Cmax değerlerinin iterasyonlar boyunca değişimi Şekil 4.4'de verilmiştir.

Çizelge 4.14 200 iş 10 makine a problemi 10 kez çalıştırılma sonuçları

Problem çalıştırma sayısı	Cmax	Amaç fonksiyonu çağırma sayısı	İşlem süresi (sn)
1	2868	391840	116.7
2	2885	402125	123.06
3	2854	397242	116.89
4	2850	413470	123.25
5	2851	420786	125.39
6	2843	403576	119.98
7	2856	394919	115.89
8	2854	394882	113.78
9	2839	391416	112.08
10	2832	413036	117.69



Şekil 4.5 200 iş 10 makine a problemi iterasyon sayısına göre cmax değerleri

HGKO algoritması ile elde edilen sonuçlar diğer algoritmalar ve Cplex ortamında matematiksel ile elde edilen sonuçlarla karşılaştırılmıştır. Matematiksel modelde küçük boyutlu problemler için optimum değerler elde edilmiş ancak iş sayısının 50 ve daha fazla olduğu problem setlerinde belirli bir süre içerisinde optimum değerlere ulaşılamamıştır. İş sayısının 50 ve üzere olduğu durumlarda program 3600 saniye süre kısıtlaması ile çalıştırılmış ve ulaşılan en iyi değerler tabloya yazılmıştır. Matematiksel model için deneysel çalışmalar Intel i5 6400 2.70 Ghz işlemci ve 8 GB RAM'e sahip Windows 10 işletim sistemli bir bilgisayarda yapılmıştır.

5. SONUÇLAR ve ÖNERİLER

Bu çalışmada n adet iş, m adet makineden oluşan Beklemesiz Akış Tipi Çizelgeleme problemleri ele alınmıştır. Literatürde çizelgeleme ve birçok optimizasyon probleminde kullanılan Gri Kurt Optimizasyon algoritması BATÇ problemlerine uygulanmıştır. Algoritmanın performansını iyileştirmek için başlangıç popülasyonunda NEH ve SİS yöntemlerinden yararlanılmıştır. Ayrıca GKO algoritmasın çözüm uzayında arama yaparken yerel minimumlara takılmasını önlemek ve algoritmanın sömürü yeteneğini arttırmak için DKA algoritması eklenmiştir. GKO'nun ürettiği en iyi üç çözüm olan Alfa, Beta, Delta çözümlerine DKA yerel arama algoritması uygulanmıştır. Yapılan deneysel çalışmalar ile önerilen algoritmanın performansı literatürdeki diğer algoritmalar ile kıyaslanmıştır. HGKO algoritması, problem setindeki 192 problemden 154'ünde en iyi çözümü bulmuştur. Karşılaştırılan diğer algoritmalar ise Gen2 17, UÖY 23, HKKO 28 en iyi değere ulaşmıştır. Ayrıca çizelgelere bakıldığında HGKO algoritmasının küçük boyutlu problemlerde ortalama bir performans göstermesine karşılık makine sayısının arttığı daha karmaşık problemlerde diğer algoritmalara göre daha iyi performans gösterdiği görülebilmektedir. Algoritmanın optimum değerlere göre ortalama sapma miktarı hesaplandığında HGKO 0.4308 OGS değerine ulaşırken Gen2 15.0352, UÖY 11.9207, HKKO ise 10.2682 OGS değerine ulaşmıştır. Ayrıca yukarıda Deneysel çalışmalar sonucunda önerilen algoritmanın BATÇ problemleri için etkili ve verimli bir yöntem olduğu ortaya çıkmıştır.

İlerleyen çalışmalarda GKO algoritması için başlangıç popülasyonu için daha iyi algoritmalar ile üretilen çözümler eklenebilir. Diğer problem setlerinde etkinliği bilenen diğer algoritmalar ile hibrit bir şekilde kullanılması algoritmanın performansı arttırılabilir. GKO algoritmanın performansı gerçek yaşam problemlerinde test edilerek algoritmanın performansı test edilip geliştirilebilir.

KAYNAKLAR

ABDEL-BASSET, M., MOHAMED, R., ABOUHAWWASH, M., CHAKRABORTTY, R. K., and RYAN, M. J., 2021. A simple and effective approach for tackling the permutation flow shop scheduling problem. Mathematics, 9(3), 270.

ALDOWAİSAN, T., 2001. A new heuristic and dominance relations for no-wait flowshops with setups. Computers & Operations Research, 28(6), 563-584.

ALDOWAİSAN, T., and ALLAHVERDİ, A., 2003. New heuristics for no-wait flowshops to minimize makespan. Computers & Operations Research, 30(8), 1219-1231.

ALAWAD, N. A., and ABED-ALGUNİ, B. H., 2022. Discrete Jaya with refraction learning and three mutation methods for the permutation flow shop scheduling problem. The Journal of Supercomputing, 78(3), 3517-3538.

ALLAHVERDİ, A., AYDİLEK, H., and AYDİLEK, A., 2018. No-wait flowshop scheduling problem with two criteria; total tardiness and makespan. European Journal of Operational Research, 269(2), 590-601.

ALLAHVERDİ, A., GUPTA, J. N., and ALDOWAİSAN, T., 1999. A review of scheduling research involving setup considerations. Omega, 27(2), 219-239.

AYDİLEK, H., and ALLAHVERDİ, A., 2012. Heuristics for no-wait flowshops with makespan subject to mean completion time. Applied Mathematics and Computation, 219(1), 351-359.

BABOR, M., SENGE, J., ROSELL, C. M., RODRİGO, D., and HİTZMANN, B., 2021. Optimization of No-Wait Flowshop Scheduling Problem in Bakery Production with Modified PSO, NEH and SA. Processes, 9(11), 2044.

BELABİD, J., AQİL, S., and ALLALİ, K., 2020. Solving permutation flow shop scheduling problem with sequence-independent setup time. Journal of Applied Mathematics, 2020.

BRANDA, A., CASTELLANO, D., GUİZZİ, G., and POPOLO, V., 2021. Metaheuristics for the flow shop scheduling problem with maintenance activities integrated. Computers & Industrial Engineering, 151, 106989.

BAŞAR, R., 2021. Beklemesiz akış tipi çizelgeleme problemlerinin bulanık ortamda dağınık arama yöntemi ile çözümü. Konya Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, Konya, 74s.

BURKE, E. K., and NEWALL, J. P., 1999. A multistage evolutionary algorithm for the timetable problem. IEEE transactions on evolutionary computation, 3(1), 63-74.

ĐUMİĆ, M., and JAKOBOVİĆ, D., 2021. Ensembles of priority rules for resource constrained project scheduling problem. Applied Soft Computing, 110, 107606.

ENGİN, O., and GÜNAYDİN, C., 2011. An adaptive learning approach for no-wait flowshop scheduling problems to minimize makespan. International Journal of Computational Intelligence Systems, 4(4), 521-529.

ENGİN, O., and GÜÇLÜ, A., 2018. A new hybrid ant colony optimization algorithm for solving the no-wait flow shop scheduling problems. Applied Soft Computing, 72, 166-176.

GAO, Y., WANG, Z., GAO, L., and Lİ, X., 2022. A Matheuristic Approach for the No-Wait Flowshop Scheduling Problem with Makespan Criterion. Symmetry, 14(5), 913.

GÜMÜŞÇÜ, A., KAYA, S., TENEKECİ, M. E., KARAÇİZMELİ, İ. H., and AYDİLEK, İ. B., 2021. The impact of local search strategies on chaotic hybrid firefly particle swarm optimization algorithm in flow-shop scheduling. Journal of King Saud University-Computer and Information Sciences.

HANSEN, P., and MLADENOVİĆ, N., 2001. Variable neighborhood search: Principles and applications. European journal of operational research, 130(3), 449-467.

IGNALL, E., and SCHRAGE, L., 1965. Application of the branch and bound technique to some flow-shop scheduling problems. Operations research, 13(3), 400-412.

İKUCUKKOC, 2022. EEM4129 Çizelgeleme ders notları, erişim: 10 Haziran 2022, http://ikucukkoc.baun.edu.tr/lectures/EMM4129/EMM4129-Cizelgeleme-BirlesikDersNotlari.pdf

JIANG, T., and ZHANG, C., 2018. Application of grey wolf optimization for solving combinatorial problems: job shop and flexible job shop scheduling cases. Ieee Access, 6, 26231-26240.

JIERAN, Y., AİMİN, W., YAN, G., and XINYI, S., 2020. An Improved Grey Wolf Optimizer for Flexible Job-shop Scheduling Problem. In 2020 IEEE 11th International Conference on Mechanical and Intelligent Manufacturing Technologies (ICMIMT) (pp. 213-217). IEEE.

KARAKOYUN, M., 2021. Çok amaçlı optimizasyon problemlerinin çözümü için Kurbağa Sıçrama ve Gri Kurt Optimizasyonu algoritmaları tabanlı hibrit bir yöntemin geliştirilmesi. Konya Teknik Üniversitesi, Lisansüstü Eğitim Enstitüsü, Doktora Tezi, Konya, 126s.

KAYA, S., KARAÇİZMELİ, İ. H., AYDİLEK, İ. B., TENEKECİ, M. E., ve GÜMÜŞÇÜ, A., 2020. Akış tipi çizelgeme problemlerinin hibrit ateşböceği ve parçacık sürü optimizasyonu algoritmasıyla çözümünde başlangıç popülasyonlarının etkileri. Pamukkale University Journal of Engineering Sciences vol. 26, no. 1, pp. 140-149.

KESKİN, K., and ENGİN, O., 2021. A hybrid genetic local and global search algorithm for solving no-wait flow shop problem with bi criteria. SN Applied Sciences, 3(6), 1-15.

KÜÇÜKKOÇ, İ., 2018. A nondominated sorting ant colony optimization algorithm for complex assembly line balancing problem incorporating incompatible task sets. Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi, 24(1), 141-152.

Lİ, N., and FENG, C., 2021. Research on machining workshop batch scheduling incorporating the completion time and non-processing energy consumption considering product structure. Energies, 14(19), 6079.

Lİ, X., and YİN, M., 2013. An opposition-based differential evolution algorithm for permutation flow shop scheduling based on diversity measure. Advances in Engineering Software, 55, 10-31.

LİN, S. W., LU, C. C., and YİNG, K. C., 2018. Minimizing the sum of makespan and total weighted tardiness in a no-wait flowshop. IEEE Access, 6, 78666-78677.

LİU, G., SONG, S., and WU, C., 2013. Some heuristics for no-wait flowshops with total tardiness criterion. Computers & operations research, 40(2), 521-525.

MAKHADMEH, S. N., KHADER, A. T., AL-BETAR, M. A., NAİM, S., ABASİ, A. K., and ALYASSERİ, Z. A. A., 2021. A novel hybrid grey wolf optimizer with min-conflict algorithm for power scheduling problem in a smart home. Swarm and Evolutionary Computation, 60, 100793.

MİRJALİLİ, S., MİRJALİLİ, S. M., and LEWİS, A., 2014. Grey wolf optimizer. Advances in engineering software, 69, 46-61.

MİYATA, H. H., NAGANO, M. S., and GUPTA, J. N., 2019. Integrating preventive maintenance activities to the no-wait flow shop scheduling problem with dependent-sequence setup times and makespan minimization. Computers & Industrial Engineering, 135, 79-104.

NAGAR, A., HERAGU, S. S., and HADDOCK, J., 1995. A branch-and-bound approach for a two-machine flowshop scheduling problem. Journal of the Operational Research Society, 46(6), 721-734.

NAWAZ, M., ENSCORE JR, E. E., and HAM, I., 1983. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. Omega, 11(1), 91-95.

ÖZSAĞLAM, M. Y., 2009. Parçacık sürü optimizasyonu algoritmasının gezgin satıcı problemine uygulanması ve performansının incelenmesi. Selçuk Üniversitesi, Fen Bilimleri Enstitüsü, Doktora Tezi, Konya,131s.

PAN, Q. K., TASGETİREN, M. F., and LİANG, Y. C., 2008. A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem. Computers & Operations Research, 35(9), 2807-2839.

PENG, T., and ZHOU, B., 2019. Hybrid bi-objective gray wolf optimization algorithm for a truck scheduling problem in the automotive industry. Applied Soft Computing, 81, 105513.

QİN, Y., and ZHANG, H., 2016. Elite Particle Swarm Optimization Algorithm for Solving the Bi-Criteria No-wait Flexible Flow Shop Problem. International Journal of Grid and Distributed Computing, 9(12), 267-232.

QIAN, B., WANG, L., HU, R., HUANG, D. X., and WANG, X., 2009. A DE-based approach to no-wait flow-shop scheduling. Computers & Industrial Engineering, 57(3), 787-805.

REN, J., YE, C., and YANG, F., 2021. Solving flow-shop scheduling problem with a reinforcement learning algorithm that generalizes the value function with neural network. Alexandria Engineering Journal, 60(3), 2787-2800.

SAMARGHANDİ, H., 2015., A particle swarm optimisation for the no-wait flow shop problem with due date constraints. International Journal of Production Research, 53(9), 2853-2870.

SAMARGHANDİ, H., and ELMEKKAWY, T. Y., 2014. Solving the no-wait flow-shop problem with sequence-dependent set-up times. International Journal of Computer Integrated Manufacturing, 27(3), 213-228.

SİVRİKAYA, F., ŞERİFOĞLU, F., and ULUSOY, G., 1998. A bicriteria two-machine permutation flowshop problem. European journal of operational research, 107(2): 414-430.

SHYU, S. J., LIN, B. M., and YIN, P. Y., 2004. Application of ant colony optimization for no-wait flowshop scheduling problem to minimize the total completion time. Computers & industrial engineering, 47(2-3), 181-193.

TASGETİREN, M. F., PAN, Q. K., SUGANTHAN, P. N., and CHEN, A. H., 2011. A discrete artificial bee colony algorithm for the total flowtime minimization in permutation flow shops. Information sciences, 181(16): 3459-3475.

TASGETİREN, M. F., YÜKSEL, D., GAO, L., PAN, Q. K., and Lİ, P., 2019. A discrete artificial bee colony algorithm for the energy-efficient no-wait flowshop scheduling problem. Procedia Manufacturing, 39, 1223-1231.

TEKBAŞ, H., 2011. Esnek sipariş tipi üretim sistemlerinde müşteri siparişlerinin çizelgelenmesi. İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Doktora Tezi, İstanbul, 211s.

UMAM, M. S., MUSTAFİD, M., and SURYONO, S., 2021. A hybrid genetic algorithm and tabu search for minimizing makespan in flow shop scheduling problem. Journal of King Saud University-Computer and Information Sciences.

YANG, Z., and LİU, C., 2018. A hybrid multi-objective gray wolf optimization algorithm for a fuzzy blocking flow shop scheduling problem. Advances in Mechanical Engineering, 10(3), 1687814018765535.

YAĞMAHAN, B., ve YENİSEY, M. M., 2011. Akış tipi çizelgeleme problemi için KKE parametre eniyileme. İTÜDERGİSİ/d, 5(2).

ZHAO, F., QİN, S., YANG, G., MA, W., ZHANG, C., and SONG, H., 2019. A factorial based particle swarm optimization with a population adaptation mechanism for the no-wait flow shop scheduling problem with the makespan objective. Expert Systems with Applications, 126, 41-53.

ZHAO, F., LİU, Y., ZHANG, Y., MA, W., and ZHANG, C., 2017. A hybrid harmony search algorithm with efficient job sequence scheme and variable neighborhood search for the permutation flow shop scheduling problems. Engineering Applications of Artificial Intelligence, 65, 178-199.

ZHAO, F., ZHAO, L., WANG, L., and SONG, H., 2020. An ensemble discrete differential evolution for the distributed blocking flowshop scheduling with minimizing makespan criterion. Expert Systems with Applications, 160, 113678.