Pamukkale Univ Muh Bilim Derg, 26(1), 140-149, 2020

Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi

Pamukkale University Journal of Engineering Sciences

Akış tipi çizelgeme problemlerinin hibrit ateşböceği ve parçacık sürü optimizasyonu algoritmasıyla çözümünde başlangıç popülasyonlarının etkileri

The effects of initial populations in the solution of flow shop scheduling problems by hybrid firefly and particle swarm optimization algorithms

Serkan KAYA[1], İzzettin Hakan KARAÇİZMELİ[2], İbrahim Berkan AYDİLEK[3], Mehmet Emin TENEKECİ[4], Abdülkadir GÜMÜŞÇÜ[5]*

[1,2]Endüstri Mühendisliği Bölümü, Mühendislik Fakültesi, Harran Üniversitesi, Şanlıurfa, Türkiye.
serkankaya@harran.edu.tr, hkaracizmeli@harran.edu.tr
[3,4]Bilgisayar Mühendisliği Bölümü, Mühendislik Fakültesi, Harran Üniversitesi, Şanlıurfa, Türkiye.
berkanaydilek@harran.edu.tr, etenekeci@harran.edu.tr
[5]Elektrik Elektronik Mühendisliği Bölümü, Mühendislik Fakültesi, Harran Üniversitesi, Şanlıurfa, Türkiye.
agumuscu@harran.edu.tr

Geliş Tarihi/Received: 08.02.2019, Kabul Tarihi/Accepted: 11.07.2019 doi: 10.5505/pajes.2019.94763
* Yazışılan yazar/Corresponding author
Araştırma Makalesi/Research Article

Öz

Klasik akış tipi çizelgeleme problemi, birbiri ardına sıralanmış makinelerin bulunduğu ve her iş için aynı makine sırasının takip edilmesi prensibine dayalıdır. İş ve makine sayılarının artmasıyla akış tipi çizelgeleme problemleri çok karmaşık hale dönüşmektedir. Bu karmaşık problemleri çözmek üzere birçok meta sezgisel yöntem kullanılmaktadır. Meta sezgisel yöntemlerle optimum çözüm aranırken başlangıç popülasyonlarının etkisi çok büyük önem arz etmektedir. Bu çalışmada hibrit ateşböceği parçacık sürü optimizasyonu algoritması kullanılarak literatürdeki akış tipi çizelgeleme problemlerinde, farklı başlangıç popülasyonlarının etkisinin gözlemlenmesi amaçlanmaktadır. Bu amaçla beş farklı başlangıç popülasyonu oluşturma yöntemi ele alınarak, karşılaştırma testleri yapılmıştır. Nawaz-Enscore-Ham algoritmasını içeren yöntemlerin ortalama göreli sapma değerlerinin daha iyi olduğu belirlenmiştir. Nawaz-Enscore-Ham algoritmasının farklı parçacık sayısı düzeyleri için başarısı test edilmiş ve sonuçlar sunulmuştur.

Anahtar kelimeler: Akış tipi çizelgeleme, Meta sezgisel yöntemler, Hibrit ateşböceği parçacık sürü optimizasyonu, Başlangıç popülasyonu, NEH

Abstract

The classical flow shop scheduling problem is based on the principle that the machines are sequenced sequentially and that the same machine sequence is followed for each job. The flow shop scheduling problems become very complex, with the increase in the number of jobs and machines. Many meta-heuristic methods are used to solve these complex problems. The effect of initial populations has great importance for searching optimal solutions by meta-heuristics methods. In this study, it is aimed to observe the effect of different initial populations in flow shop scheduling problems in the literature by using hybrid firefly particle swarm optimization algorithm. For this purpose, 5 different initial population generation methods were set and comparison tests were performed. The mean relative deviation values of the methods including the Nawaz-Enscore-Ham algorithm were determined to be better. The success of the Nawaz-Enscore-Ham algorithm for different particle count levels has been tested and the results are presented.

Keywords: The flow shop scheduling problem, meta-heuristic methods, Hybrid firefly particle swarm optimization, Initial population, NEH

1 Giriş

Üretim ve hizmet sektörlerinin tümünde kaynak planlamasına dayalı çizelgeleme problemi, çözülmesi gereken en önemli problemlerden birisidir. Örneğin üretim sektöründe çizelgeleme problemi, atanacak iş ve kullanılacak makine sayısına bağlı olarak oldukça karmaşık duruma gelebilmektedir. Bu problemlerde amaç, genellikle eldeki makinelerde yapılması gereken işlerin en uygun sıralama ile en uygun bitirilme zamanında işlenmesini sağlamaktır. Her bir makinede gerçekleştirilecek işlerin çalışma süresi farklı olabilir. Belirlenen sıralama ile tüm işler makinelerde işlenecektir. İdeal sıralama yapılamaması durumunda makinelerin boş beklemesi veya bazı makinelerde iş kuyruklarının oluşması olasıdır. Bu durumda toplam iş bitirme süresi de olumsuz etkilenebilir.

İş sayısı ve makine sayısı, problemin boyutunu belirleyen parametrelerdir. Problem boyutunun artması problemin çözümünü karmaşık hale getirmektedir. Bunun temel sebebi çözüm uzayının üstel bir biçimde büyümesidir. Çok sayıda iş ve makine içeren akış tipi çizelgeleme problemleri birleşik eniyileme problemi özelliğindedir ve NP-zor tipi problemler sınıfındadır [1]. Bu durumda tüm çözüm ihtimallerinin denenmesi zamansal olarak mümkün olmamaktadır. Bunun yanı sıra küçük boyutlu problemlerde optimum sonuca ulaştıran geleneksel yöntemler de problem boyutunun büyümesi sonucu istenen sonuca ulaşamamaktadır. Bu sebeple akış tipi çizelgeleme problemlerinin çözümü için birçok meta sezgisel yöntem kullanılmıştır.

Çoğu mühendislik problemlerinin çözümünde meta sezgisel yöntemlerin kullanımı, son yıllarda önemli bir oranda artmıştır. Matematiksel modelin oluşturulamadığı veya model kurmanın çok maliyetli olduğu optimizasyon problemlerine, meta sezgisel yöntemlerin kolaylıkla uygulanabilir olması, iyi bir hesaplama gücünün olması, tatmin edici sonuçlar alınabilmesi ve bir problem için geliştirilen meta sezgiselin başka bir

probleme de uygulanabilir olması gibi nedenler, problem çözümünde meta sezgisel yöntemlerin tercih edilme nedenleri olmuştur [2].

Bu çalışmada Aydilek [3] tarafından geliştirilen hibrit ateş böceği ve parçacık sürü optimizasyonu algoritması (HAPSO) ile daha iyi çözümler elde edilebilmesi için başlangıç popülasyonuna bazı sıralama kuralları adapte edilmiştir. Kullanılan HAPSO algoritması, parçacık tabanlı bir algoritmadır. Bu tip algoritmalarda, kullanılan başlangıç popülasyonu algoritmanın verimliliğini önemli ölçüde etkilemektedir. Bu nedenle doğru başlangıç popülasyonunun seçilmesi oldukça önemli olup başarı oranını etkilemektedir.

Çalışmanın ikinci bölümünde akış tipi çizelgeleme problemleri üzerine yapılmış önceki çalışmalara yer verilmiştir. Üçüncü bölümde, akış tipi çizelgeleme ve Aydilek [3] tarafından geliştirilen HAPSO algoritmasına değinilmiştir. Çalışmanın dördüncü bölümünde deneysel çalışmadan, beşinci bölümünde ise elde edilen bulgulardan bahsedilmiştir. Altıncı bölümde ise sonuçlar toparlanmış ve gelecekteki çalışmalarla ilgili önerilerde bulunulmuştur.

2 Literatür taraması

Literatürde akış tipi çizelgeleme problemlerini ele alan ilk araştırmacı 1954 yılında Johnson [4] olmuştur. Yazar, *n* sayıda işin 2 makinede çizelgelenmesi probleminin çözümü için kendi adıyla anılan algoritmayı geliştirmiştir. Daha sonraki yıllarda makine sayısının ikiden fazla olduğu akış tipi çizelgeleme problemleri üzerine çalışmalar yapılmıştır. Makine sayısının ikiden fazla olduğu akış tipi çizelgeleme problemleri NP-zor problemler kapsamında olduğundan bu tür problemlerin çözümü için çeşitli sezgisel yöntemler geliştirilmiştir. 1965 yılında Palmer [5] tarafından geliştirilen Eğim dizisi yöntemi; 1970 yılında Campbell ve diğ. [6] tarafından sunulan Campbell, Dudek Smith (CDS) Algoritması; Gupta (1971) tarafından geliştirilen sezgisel algoritma [7]; Dannenbring Yöntemi [8] (1972); Navaz ve diğ. [9] tarafından sunulan Nawaz, Enscore ve Ham (NEH) Yöntemi (1983); Hundal ve Rajgopal Yöntemi [10] (1988); Widmer ve Hertz Yöntemi [11] (1989); Ho ve Chang (HC) Yöntemi [12] (1991) bu alandaki çalışmalardan bazıları [13] olup zaman içerisindeki gelişimlerine göre verilmiştir.

Janiak [14] akış tipi çizelgeleme problemlerinin en büyük tamamlanma zamanını en küçükleme amaçlı çözümü için dal sınır algoritması, Tandon ve diğ. [15] tavlama benzetimi, Benavides ve Ritt [16] yapıcı iterasyonlu yerel arama sezgiseli, Cui ve diğ. [17] aynı problemin kullanılabilirlik kısıtlı çözümü için yerel arama ve genetik algoritmadan hibrit bir algoritma, Henneberg ve Neufeld [18] tavlama benzetimi, Pugazhenthi ve Xavior [19] ise genetik algoritma ile yarasa algoritmasını birleştirdikleri hibrit bir algoritma sunmuşlardır. Aynı problemin toplam tamamlanma zamanı amaçlı çözümü için Benavides ve Ritt [20] komşuluk yapısını esas alan yerel arama algoritması sunmuşlardır. Reeves ve Yamada [21] toplam akış süresini en küçükleme amaçlı yerel arama ve genetik algoritmayı birleştirdikleri hibrit bir algoritma, Rajendran ve Ziegler [22] toplam akış süresi ve en büyük tamamlanma zamanını en küçükleme amaçlı karınca kolonileri algoritması, Taşgetiren ve diğ. [23] ise parçacık sürü optimizasyonu algoritması sunmuşlardır.

İşler [24] en büyük tamamlanma zamanını en küçükleme amaçlı esnek akış tipi çizelgeleme problemlerinin çözümü için paralel doyumsuz algoritma, Alaykıran ve diğ. [25] karınca kolonileri algoritması, Kahraman ve diğ. [26] genetik algoritma, Kianfar ve diğ. [27] ortalama gecikme ölçütlü çözüm için hibrit genetik algoritma, Jolai ve diğ. [28] en büyük gecikme ve en büyük tamamlanma zamanını en küçükleme amaçlı çözümü için tavlama benzetimi sunmuşlardır.

Keskin [29] beklemesiz akış tipi çizelgeleme probleminin çözümü için çok amaçlı genetik algoritma sunmuştur. Problemi en büyük tamamlanma zamanı ve toplam akış zamanı olmak üzere çok amaçlı ele almıştır. Laha ve Sapkal [30] toplam akış zamanını en küçükleme amaçlı sezgisel bir algoritma, Tseng ve Lin [31] en büyük tamamlanma zamanını en küçükleme amaçlı çözüm için yerel arama ve genetik algoritmadan elde edilmiş hibrit bir algoritma, Chaundry ve Mahmood [32] genetik algoritma sunmuşlardır. Czogalla ve Fink [33] aynı problemin gecikme süresini en küçükleme amaçlı çözümü için yerel arama ve parçacık sürü optimizasyonu algoritmalarını birleştirdikleri hibrit bir algoritma sunmuşlardır.

Pan ve diğ. [34] gecikme süresi ve en büyük tamamlanma zamanını en küçükleme amaçlı diferansiyel gelişim algoritması, Moghaddam ve diğ. [35] yapay bağışıklık algoritması, Allahverdi ve Aldowaisan [36] hibrit genetik algoritma sunmuşlardır. Araujo ve Nagano [37] sıra bağımlı hazırlık zamanlı beklemesiz akış tipi çizelgeleme problemlerinin en büyük tamamlanma zamanını en küçükleme amaçlı çözümü için sezgisel bir algoritma, Kumar ve Singhal [38] aynı problemin çözümü için genetik algoritma sunmuşlardır.

3 Akış tipi çizelgeleme

Akış tipi çizelgeleme, birbirine ardışık bağlı *m* adet makinenin bulunduğu ve her bir işin aynı rotayı izleyerek makinelerin her birinde işlem gördüğü süreci ifade etmektedir. Makinede tamamlanan bir iş, sistemde sonraki bağlı diğer bir makinede işlem görür. Bu sistemlerde genellikle tek bir ürün veya birbirine benzeyen ürün grubu bulunmaktadır. İş istasyonları, ürünün üretimi için gereken işlem sıralarına göre oluşturulmuştur. Özel amaçlı makineler, üretim akışına göre sıralanmıştır ve iş akışı düzgündür [1][39].

Başka bir ifade ile akış tipi çizelgeleme, birbirinden farklı *m* adet makine ve *n* sayıda işin bulunduğu; her bir işin belirli sayıda operasyondan oluştuğu, her bir operasyonun farklı makinelerde yapıldığı ve bütün işlerin operasyonlarının aynı sırayla yapıldığı problemlere denir. Basit akış tipi iş çizelgeleme problemlerinde her iş seri makinelerde tam olarak aynı sırayla işlem görmektedir [40]. Bu çizelgeleme problemi akış tipi çizelgeleme problemlerinin en yalın hali olup permütasyon akış tipi çizelgeleme problemi olarak da adlandırılmaktadır.

Araştırmacılar, tek amaçlı akış tipi çizelgeleme problemlerinin küçük boyutlu problemlerinin çözümünde matematiksel model kurarak optimal sonuçlara ulaşmışlardır. Büyük boyutlu problemlerin çözümünde ise henüz optimal sonucu veren klasik yöntem olmadığından ve doğrusal programlama modelleri problemleri çözmekte yetersiz kaldığından dolayı araştırmacılar meta sezgisel yöntemleri tercih etmişlerdir. Kullanılan meta sezgisel yöntemler ile kısa süre içerisinde optimale yakın sonuçlar elde edildiği gösterilmiştir. Araştırmacılar, meta sezgisel yöntemleri kullanırken genellikle klasik yöntemlerden birini tercih etmişlerdir. En çok kullanılan meta sezgisel yöntemin, genetik algoritma olduğu görülmektedir [41]. Kullanılan yöntemin en iyi parametre değerlerini belirlemek için ise genelde deney tasarımı yapılmıştır. Az sayıda olsa da bazı araştırmacılar meta sezgisel yönteme yerel arama gibi çözümleri geliştirici bazı kuralları

adapte etmişlerdir. Bazı araştırmacılar ise daha etkin sonuçlara ulaşabilmek için başlangıç popülasyonlarını belirli kurallar dahilinde oluşturmuşlardır.

3.1 Ayrık hibrit ateşböceği ve parçacık sürü optimizasyonu algoritması

Ateşböceği optimizasyonu (ABO) algoritması doğadan esinlenen sürü tabanlı bir meta sezgisel algoritmadır. Ateşböceği sürüsü içindeki bir bireyin daha çekici olan bireye doğru yaklaşımının modellenmesi sonucu Yang [42] tarafından optimizasyon algoritması olarak önerilmiştir. Ateşböcekleri sürü halinde yaşarken her bir bireyin kendisine göre daha parlak olana doğru yaklaştığı varsayılmıştır. Hareketi etkileyen faktör, ışık şiddeti olduğundan bunu etkileyen içinde bulunulan ortama ait katsayı ve uzaklık değerleri ateşböceklerini yönlendirmektedir. Uzaklık arttıkça ışık şiddetinin etkisi de azalacağı için uzaktaki ateşböceği yakındakinden daha az çekici olmaktadır. Ters kare kanununa göre ışık şiddeti Eşitlik 1 ile ifade edilmektedir. Eşitlikteki; *I(r)* elde edilecek ışık şiddetini, I_s ışık kaynağının gücünü ve r değeri ise uzaklığı ifade etmektedir.

$$I(r) = \frac{I_s}{r^2} \tag{1}$$

Bununla birlikte ışığın içinde yayıldığı ortamın ışık emilimi de hesaba katılmalıdır. Işığın içinde yayıldığı ortam özelliklerine göre ışık kaynağı gücüne karşı emilim göstermektedir. Bunlara bağlı olarak uzaklık sıfır olduğunda da tanımsız durumu oluşmaması için Eşitlik 2 ile ateşböceklerinin çekicilik miktarı ifade edilmektedir. Eşitlikteki γ değeri emilim katsayısıdır.

$$B(r) = B_0 e^{-\gamma r^2} \tag{2}$$

Ateşböceklerinin Eşitlik 3'te gösterildiği gibi yeni konumları hesaplanarak, en uygun çözüm bulunmaya çalışılır. X_i ve X_j farklı iki ateşböceğini ifade etmektedir. Eşitlik 3 ile X_i nin yeni konumu bulunmaktadır. ϵ_i Rastgele değişkenler vektörü ve α ise rastgele değişken katsayı parametresidir.

$$X_i = X_i + B_0 e^{-\gamma r_{i,j}^2}\left(X_j - X_i\right) + \alpha\epsilon_i \tag{3}$$

Parçacık sürü optimizasyonu (PSO) algoritması Kennedy ve Eberhart [43] tarafından önerilmiş ve ABO algoritması gibi sürü tabanlı bir yakınsama yapmaktadır. Kuşların veya balıkların birlikte hareket etmelerini modellemektedir. Algoritmada her bir parçacık en iyi konumunu saklar ve küresel optimuma ulaşmak için her adımda konumunu günceller. Bu güncellemeler yapılırken *peniyi* ve *geniyi* değerleri dikkate alınarak işlemler gerçekleştirilir. *peniyi* değeri her bir parçacığın elde ettiği en iyi konumu ifade ederken, *geniyi* değeri ise tüm sürünün elde ettiği en başarılı konumu göstermektedir.

$$v_i^{t+1} = w.v_i^t + c_1.r_1^t.(peniyi_i^t - x_i^t) + c_2.r_2^t.(geniyi_i^t - x_i^t) \tag{4}$$

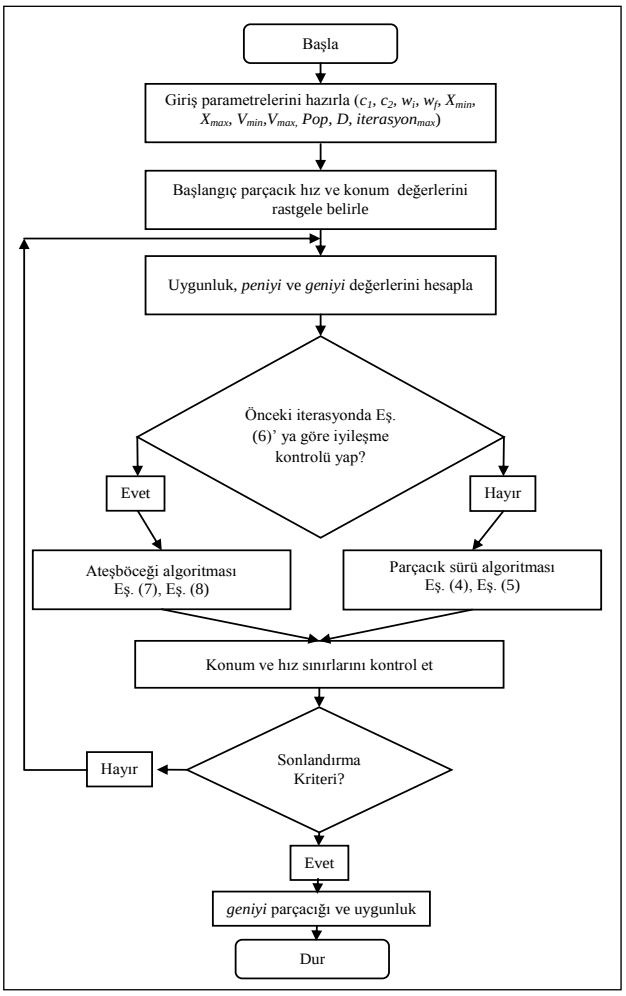
x_i değeri parçacıkların konum değerlerini ve v_i değeri ise parçacıklara ait hız değerlerini ifade etmektedir. c_1 ve c_2 ivme katsayı parametreleridir. r_1 *ve* r_2 ise [0,1] aralığındaki rastgele değerleri ifade etmektedir. x_i^{t+1} parçacığın sonraki döngü sonucundaki yeni konumunu ifade etmekte ve Eşitlik 4 ve Eşitlik 5 ile hesaplanmaktadır.

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{5}$$

PSO algoritması hız parametresi kullanımı ve en iyi değerini (*peniyi*) saklaması sebebiyle oldukça başarılı bir algoritmadır ve hızlı arama işlemi gerçekleştirmektedir. Bu özelliği ile küresel aramada etkin bir yöntem olarak literatürde yer edinmiştir. Ancak bunun yanında yerel aramalarda bazen salınımlar yaparak sonuca ulaşması gecikebilmektedir. Bundan dolayı yerel arama yeteneği küresel arama yeteneğine göre biraz daha az başarılı olabilmektedir.

ABO algoritması ise hız parametresi ve geçmiş kişisel en iyi değerleri saklayacak parametreleri barındırmamaktadır. Bu sebep ile yerel aramada daha az etki altında kalmakta ve daha uygun bir çözümü bulabilmektedir. Bu şekilde başarılı bir yerel arama, sömürü gerçekleştirmektedir. Ancak bazen PSO'ya göre daha uzun sürede küresel sonuca ulaşabilmektedir.

Aydilek [3] tarafından önerilen HAPSO algoritması, ABO algoritması ile PSO algoritmasının hibrit bir şekilde kullanılması sonucu oluşturulmuştur. Yazar tarafından her iki yöntemin güçlü yanları bir arada kullanılarak daha başarılı bir algoritma elde edildiği gösterilmiştir. HAPSO algoritması; ABO ve PSO algoritmalarının arama kabiliyetlerini birleştirerek kullanmaktadır. PSO'nun hızlı arama özelliği ile küresel çözümleri, ABO'nun başarılı sömürme özelliği ile yerel çözümleri arama başarıları birleştirilmiştir. Şekil 1'de ifade edilen HAPSO yönteminin akış şeması verilmektedir.



Şekil 1: HAPSO akış şeması.

Her bir döngüde kullanılacak olan meta sezgisel yöntem, önceki döngüdeki uygunluk fonksiyonunda gerçekleştirilen

iyileştirme ile belirlenmektedir. Uygunluk fonksiyonunda gerçekleştirilecek iyileştirme kontrolü Eşitlik 6'ya göre yapılmaktadır.

$$f(i,t) = \begin{cases} doğru, eğer\ uygunluk(parça_i^t) \leq geniyi^{t-1} \\ yanlış, eğer\ uygunluk(parça_i^t) > geniyi^{t-1} \end{cases} \quad (6)$$

Parçacıklara ait yeni pozisyon ve hız değerleri Eşitlik 7 ve Eşitlik 8'e göre hesaplanmaktadır.

$$X_i(t+1) = X_i(t) + B_0 e^{-\gamma r_{i,j}^2}(X_i(t) - geniyi^{t-1}) + \alpha\epsilon_i \quad (7)$$

$$V_i(t+1) = X_i(t+1) - X_{i_geçici} \quad (8)$$

PSO da atalet ağırlığı olan w parametresi arama ve sömürü arasında dengeyi ayarlamak için kullanılır. Bu değer her aşamada Eşitlik 9'a göre lineer azaltılarak kullanılmaktadır [44].

$$w = ((w_i - w_f)/iterasyon_{max}) \times iterasyon \quad (9)$$

Parçacık mevcut *geniyi* değerinden daha başarılı bir değere ulaşırsa yerel arama yapması için ABO algoritması ile yerel aramaya geçilmektedir. Aksi durumda PSO yöntemine uygun olarak arama yapılmaya devam edilmektedir. Belirlenen sonlandırma kriterine ulaşıldığında algoritmanın çalışması sonlandırılır ve *geniyi* ile uygunluk değeri çıktısı sonuç olarak döndürülmektedir.

HAPSO algoritmasında kullanılan karar değişkenleri sürekli değerlerden oluşmaktadır. HAPSO algoritması sürekli değerleri hesaplamak ve iyileştirmek üzerine sahip bir çalışma yapısına sahiptir. Oysa akış tipi çizelgeleme problemi doğası gereği kesikli, ayrık bir yapıdadır. Akış tipi problemlerde aranan sonuç, işlerin makinelere verildiği en uygun sıra diziliminin bulunmak istenmesidir. Bu nedenle HAPSO algoritması da en küçük pozisyon değeri kuralı (EPD) Taşgetiren ve diğ. [23], Bean [45] benimsenerek ayrık bir hale dönüştürülmüştür. Sürekli değerler küçükten büyüğe sıralanarak uygunluk fonksiyonu hesabı yapılırken sıra indeksleri kullanılmaktadır. Örneğin: (1.25, -2.15, 4.75, 3.05, -1.45) değerlerinden oluşan bir çözüm örneği küçükten büyüğe sıralanarak indeks değerleri dizisi olan (2, 5, 1, 4, 3) haline getirilmiş ve uygunluk değeri böylece hesaplanabilmiştir. Bu şekilde sürekli değerler ayrık değerlere dönüştürülmekte ve ayrık HAPSO (AHAPSO) algoritması elde edilmektedir.

3.2 Sıralama kuralları

Çizelgeleme problemlerinde sıralama yapılırken göz önünde bulundurulan basit kurallar sıralama kuralları olarak adlandırılırlar. Bu kurallara göre, işlerin makinede sıralaması yapılır. Sıralama kuralları, çizelgeleme problemlerine kolaylıkla uygulanabilmeleri ve uzun hesaplama süresi gerektirmemelerinden dolayı tercih edilirler. Küçük boyutlu problemler için tatmin edici sonuçlar vermelerine rağmen, problem boyutu büyüdükçe bu kurallar yetersiz kalabilirler. Bu kuralların avantajı olarak, anlaşılması ve uygulanmasının kolay olması söylenebilir. Dezavantajı olarak, en iyi çözümü garanti etmedikleri söylenebilir. Farklı çizelgeleme amaçları için farklı kurallar geliştirilmiştir [2]. En kısa işlem zamanı olan iş önce (EKİZ), en uzun işlem zamanı olan iş önce (EBİZ), ağırlıklandırılmış en kısa işlem zamanı önce, ilk gelen ilk çıkar, son gelen ilk çıkar, bu kurallardan sadece birkaçıdır.

4 Deneysel çalışma

Sıralama kuralları, başlangıç popülasyonunun daha verimli olması için kullanılmaktadır. Problemin çözümüne tamamı rastgele değerler içeren popülasyon ile başlamak yerine sıralama kuralları ile daha iyi bir çözümden başlanması amaçlanmaktadır. Böylece meta sezgisel algoritmanın çalışması süresince elde edilen sonuçların daha iyi olacağı düşünülmektedir. Bu çalışmada AHAPSO algoritmasının daha iyi sonuçlar verebilmesi için aşağıda verilen kurallar çerçevesinde 5 farklı başlangıç popülasyonu oluşturulmuştur. Yapılan denemelerin sonuçlarına göre bu farklı kurallar karşılaştırılmıştır. Bunlar;

1. Tüm parçacıklar rastgele (ST),
2. Başlangıç parçacık sayısının %10'u NEH [9] ve geri kalanı rastgele (NEH10),
3. 1 parçacık NEH, EKİZ ve EBİZ, geri kalan parçacıklar rastgele (SKNEH1),
4. Başlangıç parçacık sayısının %10'u NEH, EKİZ ve EBİZ, geri kalan parçacıklar rastgele (SKNEH10),
5. EKİZ ve EBİZ, geri kalan parçacıklar rastgele (SK).

Bu kurallardan rastgele kuralı algoritmanın doğal halidir. EKİZ ve EBİZ'e önceki bölümde değinilmiştir. EKİZ ve EBİZ kuralları, hem her makine için ayrı ayrı hem de bir işin tüm makinelerdeki işlem zamanları toplanarak o işe ait toplam işlem zamanlarının kullanılması yoluyla farklı parçacıklar üretecek şekilde kullanılmıştır. EKİZ ve EBİZ kuralları için kaba kod Şekil 2'de görülmektedir.

```
Başlangıç değerlerini al
liste=[];
Tekrar et i (makine_Sayısı)
    geçici_liste=sırala(veri(:,i))   //Sıradaki   makine
süresine göre sırala
    liste.ekle(geçici_liste)
    geçici_liste=ters_sırala(veri(:,i)) //Sıradaki makine
    süresine göre sırala
    liste.ekle(geçici_liste)
son
Tüm işler tüm makinelerde geçirdiği toplam süreyi
küçükten büyüğe sırala
Her bir j={j1,j2,..,jk} iş için Topla({mk1,mk2,..mk}) tüm
makinelerde geçirdiği süre c={c1,c2,..,ck}
geçici_liste = sırala(c)
liste.ekle(geçici_liste);
geçici_liste = ters_sırala(c)
liste.ekle(geçici_liste);
geçici_liste=[]
Tekrar et (Tüm işler listeye eklenene kadar)
    Tüm işler için en düşük makinede çalışma süresini
belirle   jmin_indis=veri.bul(en_düşük_işlem_suresi)
    geçici_liste.iş_ekle(jmin_indis)
    Belirlenen işin makine sürelerini Inf yap
    veri(jmin_indis,:)=Inf
son
liste.ekle(gecici_liste);
gecici_liste = ters_sırala(c)
liste.ekle(gecici_liste);
son
```

Şekil 2: EKİZ ve EBİZ kuralları kaba kodu.

NEH kuralı ise Nawaz ve diğ. [9] tarafından sunulan Nawaz-Enscore-Ham (NEH) algoritmasıdır. NEH algoritmasında en büyük toplam işlem zamanına sahip iki iş önce uygun şekilde sıralanmaktadır. Daha sonra tekrarlı şekilde bu iki işe geri

kalan işler sırasıyla olası permütasyonlara göre eklenmektedir. Bu şekilde algoritmanın sonunda minimum en büyük toplam tamamlanma zamanını (C_{max}) veren dizilim elde edilmektedir. Bu algoritmanın kaba kodu ise Şekil 3'te görülmektedir.

```
Başla
    Başlangıç parametrelerini belirle
    Her bir iş için makinelerde harcanan toplam süreyi belirle j=[j1,j2,...,jn], n iş sayısı
    sırala(j,büyükten_küçüğe)
    Eğer hesaplama_süresi([1,2])<hesaplama_süresi([2,1])
        liste = [1, 2]
    değilse
        liste = [2, 1]
    son

    Tekrar et k (j3,j4,...,jn) //Büyükten küçüğe sıralı durumda
        Tekrar et b (boy(liste)+1)
            yeni_liste = jk işi b. konuma yerleştir
            süre = hesaplama_süresi(yeni_liste)
        son
    yeni_liste = yeni_liste de min(süre)
        liste.ekle(yeni_liste)
    son
son
```

Şekil 3: NEH algoritması kaba kodu.

Denemeler, Taillard [46] tarafından akış tipi çizelgeleme problemleri için oluşturulan veri setleri üzerinde gerçekleştirilmiştir. Veri setlerinde 12 farklı problem grubu bulunmaktadır. Her problem grubunda da 10 farklı örnek yer almakta yani toplam 120 farklı problem vardır.

Beş yöntem her problem için 10 tekrar olacak şekilde Intel Core i5-4570 işlemcili 4 GB RAM özellikli bilgisayarda çalıştırılmıştır. Popülasyon büyüklüğü 120, iterasyon sayısı 10000 olarak sabit alınmış ve toplamda 6000 adet deneme yapılmıştır. Yöntemleri karşılaştırmak için ortalama göreli sapma (OGS) değeri kullanılmıştır [47]. Denklem 10'da C^{Hs} hesaplanan C_{max} değerini; C^{UB} ise ilgili Taillard probleminin bilinen en iyi çözümünü veya üst sınır değerini ifade etmektedir.

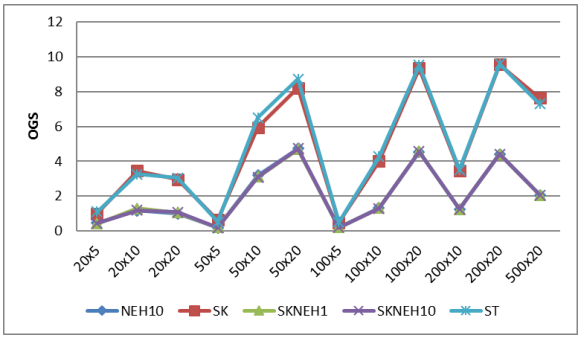
$$OGS = \frac{(C^{Hs} - C^{UB}) \times 100}{C^{UB}} \quad (10)$$

Wang ve diğ. [47] akış tipi çizelgeleme problemlerinin çözümü için parçacık sayısının %10'unu NEH algoritmasından elde edecek şekilde kurguladıkları hibrit stratejisiyle kullandıkları guguk kuşu algoritmasından başarılı sonuçlar elde etmişlerdir. Bölümün başında bahsedilen 5 farklı başlangıç popülasyonu denemesinden sonra ayrıca farklı NEH düzeylerini karşılaştırmak için de denemeler yapılmıştır. Taillard veri setlerindeki 12 farklı problem grubunun her birinden seçilen bir problem için, 120 parçacıklı popülasyonun, sırasıyla 1 parçacığı, %10'u, %25'i, %50'si ve %100'ü NEH algoritmasıyla oluşturulmuştur. Bu denemeler de 10 tekrarlı olacak şekilde gerçekleştirilmiştir. Böylece NEH algoritması düzeyi ile rastgelelik düzeyi arasında karşılaştırma yapılabilmiştir.

5 Bulgular ve tartışma

Yöntemleri birbirleriyle karşılaştırabilmek için yapılan deneylerde hesaplanan OGS değerlerinin 12 problem grubu için ortalama değerleri Tablo 1'de görülmektedir. *n*, iş sayısını; *m* ise makine sayısını ifade etmek üzere her problem grubundaki en düşük OGS değerleri koyu renkle belirtilmiştir. SK ve ST yöntemleri ile hiçbir problem grubunda en düşük OGS değeri elde edilememiştir. NEH10, SKNEH10 ve SKNEH1 yöntemleriyle sırasıyla 7, 6 ve 5 problem grubunda en düşük OGS değerlerine ulaşılmıştır. Tüm problem grupları için genel ortalamaya bakıldığında SKNEH10, NEH10 ve SKNEH1 yöntemlerinin daha düşük ortalama OGS değerlerine sahip olduğu görülmektedir.

Şekil 4'teki grafikte de görüldüğü gibi SK ve ST yöntemleri problem gruplarına göre genelde daha yüksek OGS değerleri üretirken, diğer üç yöntem görece daha düşük OGS değerleri üretmiştir. NEH10, SKNEH1 ve SKNEH10 yöntemlerinin sonuçları tüm problem gruplarında birbirlerine oldukça yakındır. Buradan hareketle hangi yöntemin istatistiksel olarak daha iyi olduğunu belirlemek için öncelikle Friedman testi yapılmıştır [47]. Tablo 2'de görüldüğü gibi SKNEH10 yöntemi en düşük ortalama sıra değerine ulaşmıştır. Ancak NEH10 ve SKNEH1 yöntemlerinin ortalama sıra değerleri de SKNEH10 yöntemine oldukça yakındır.



Şekil 4: Problem gruplarına göre OGS değerleri.

Tablo 1: Yöntemlere göre OGS değerleri.

Taillard	nxm	NEH10	SK	SKNEH1	SKNEH10	ST
001-010	20x5	0.46	0.99	0.42	0.45	1.11
011-020	20x10	1.17	3.45	1.29	1.19	3.26
021-030	20x20	0.99	2.97	1.05	1.08	3.05
031-040	50x5	0.19	0.64	0.20	0.18	0.53
041-050	50x10	3.19	5.95	3.10	3.08	6.50
051-060	50x20	4.73	8.21	4.70	4.73	8.72
061-070	100x5	0.20	0.48	0.20	0.20	0.50
071-080	100x10	1.32	4.01	1.32	1.30	4.26
081-090	100x20	4.54	9.34	4.57	4.56	9.52
091-100	200x10	1.26	3.45	1.26	1.26	3.51
101-110	200x20	4.41	9.55	4.41	4.41	9.61
111-120	500x20	2.07	7.63	2.07	2.07	7.29
Genel Ortalama		2.04	4.72	2.05	2.04	4.82

Tablo 2: Friedman testine göre yöntemler için ortalama sıralar.

Yöntem	Ort. Sıra
SKNEH10	1.88
NEH10	2.00
SKNEH1	2.13
SK	4.25
ST	4.75

Beş yöntem arasında istatistiksel olarak fark olup olmadığını belirlemek için Wilcoxon işaretli sıralar testi yapılmıştır. Tablo 3'te yapılan testin sonuçları görülmektedir.

Tablo 3: Wilcoxon işaretli sıralar testi sonuçları.

Yöntem	p_değeri
NEH10	0.941
SKNEH1	0.766
SK	0.000
ST	0.000

En düşük sıralama değerine sahip olan SKNEH10 yöntemi ile diğer yöntemler sırayla karşılaştırılmıştır. α=0.05 düzeyi için, SKNEH10 yöntemi ile NEH10 ve SKNEH1 yöntemleri arasında istatistiksel olarak anlamlı bir fark bulunamamıştır. SKNEH10 ile SK ve ST yöntemleri arasında ise istatistiksel olarak anlamlı bir fark vardır ve SKNEH10 yöntemi ile daha iyi sonuçlar elde edilmektedir. Özetle bu çalışma kapsamında araştırılan beş yöntemden SKNEH10, SKNEH1 ve NEH10 yöntemleri ile diğerlerinden daha iyi sonuçlara ulaşıldığından ve bu üç yöntem arasında da istatistiksel olarak anlamlı bir fark bulunmadığından, başlangıç popülasyonu oluşturmak için bu üç yöntemden herhangi biri tercih edilebilir denilebilir.

Geçmiş çalışmalardan dolayı NEH algoritmasının başarısı zaten bilinmektedir. Elde edilen sonuçlardan anlaşıldığı kadarıyla NEH algoritması AHAPSO algoritmasına da katkı sağlamıştır. Tablo 4'te ise NEH algoritması kullanımında farklı parçacık sayısı düzeyleri için ortalama OGS değerleri görülmektedir. *n*, iş sayısını; *m* ise makine sayısını ifade etmek üzere her problemdeki en düşük OGS değerleri koyu renkle belirtilmiştir.1, %10 ve %25 düzeylerinde sırasıyla 4, 4 ve 5 problemde en düşük OGS değerleri elde edilmiştir. %50, %75 ve %100 düzeyleri için bu durum 1, 2 ve 1 adetle sınırlı kalmıştır. Genel ortalamada ise en düşük değer %10 düzeyinde elde edilirken; 1, %25 ve %50 düzeylerinin ortalama OGS değerleri ise birbirine oldukça yakın oluşmuştur.

Sonuçları istatistiksel olarak yorumlayabilmek için Friedman testi ve Wilcoxon işaretli sıralar testi yapılmıştır. Tablo 5'te görüleceği üzere %10 düzeyi en düşük ortalama sıra değerine sahiptir. p değerlerine bakıldığında ise α=0.05 düzeyi için, %10 düzeyi ile %25 düzeyi hariç diğer düzeyler arasında istatistiksel olarak anlamlı bir fark vardır. %10 düzeyi daha iyi bir performansa sahiptir. %10 düzeyi ile %25 düzeyi istatistiksel olarak anlamlı farka sahip değildir. Yani her iki düzey de tercih edilebilir durumdadır.

Son olarak başlangıç çözümünün ve AHAPSO algoritmasının çözüme katkısını anlayabilmek için %10 NEH düzeyi için algoritmayla elde edilen başlangıç çözümü ve son çözüm değerleri karşılaştırılmıştır. Algoritmanın 12 problemde 10 tekrarlı şekilde çalıştırıldığı ve 120 parçacık içeren popülasyon kullanıldığı daha önce belirtilmişti. Bu doğrultuda Tablo 6'da başlangıç çözümü ve son çözüm sonuçları görülmektedir.

Min değerleri ilgili çözümdeki 120 parçacıktan elde edilen en küçük çözümlerin 10 tekrar için ortalamasını, *max* değerleri aynı mantıkla en büyük çözümlerin ortalamasını, *ort.* değerleri 120 parçacıktan elde edilen çözümlerin ortalamasını ve *SS* değerleri ise bunların standart sapmalarını göstermektedir. *UB* değerleri, Taillard probleminin bilinen en iyi çözümünü veya üst sınır değerini ifade etmek üzere, *OGS1* başlangıç çözümlerinin ortalaması ile *UB* arasındaki yüzdelik sapmayı, *OGS2* başlangıç çözümlerinin ortalaması ile son çözümlerin ortalaması arasındaki yüzdelik sapmayı, *OGS3* ise son çözümlerin ortalaması ile *UB* arasındaki yüzdelik sapmayı ifade etmektedir.

Başlangıç çözümü ile *UB* arasında genel ortalamada yaklaşık %19.8 fark varken son çözüm ile *UB* arasındaki fark %3.2'ye düşmektedir. Yani başlangıç çözümünden sonra AHAPSO algoritmasının katkısı yaklaşık %16.5 düzeyindedir. Özellikle *OGS1* ve *OGS2* değerlerinde problem büyüdükçe yani çizelgelenecek iş sayısı arttıkça düşüş eğilimi gözlemlenmekte yani başlangıç çözümünün etkisi artmaktadır.

Öte yandan *SS* değerlerine bakıldığında başlangıç çözümünde problem büyüdükçe *SS* değeri artarken, son çözümde görece daha yatay bir seyir izlemektedir. Yani iş sayısı arttıkça başlangıç çözümü daha başarılı sayılabilecek bir tarama yapmakta ancak bununla birlikte *SS* değerleri de artmaktadır. Amaçlandığı gibi başlangıç çözümünün ardından AHAPSO algoritması devreye girdiğinde ise iyi çözümlere doğru hızla odaklanılmaktadır.

Tablo 4: NEH düzeylerine göre OGS değerleri.

Taillard	nxm	1*	%10	%25	%50	%75	%100
010	20x5	0.45	0.11	0.00	0.20	0.65	0.32
020	20x10	1.18	1.11	1.57	1.32	1.65	1.69
030	20x20	0.81	0.82	0.98	0.91	1.03	0.76
040	50x5	0.00	0.00	0.01	0.01	0.01	0.00
050	50x10	4.04	3.68	3.95	3.80	3.83	4.06
060	50x20	4.05	4.08	4.27	4.66	4.78	5.07
070	100x5	0.16	0.14	0.14	0.20	0.30	0.34
080	100x10	0.99	0.99	0.99	0.99	0.99	1.01
090	100x20	3.15	3.09	3.09	3.04	3.02	3.08
100	200x10	0.92	0.88	0.87	0.98	0.91	0.89
110	200x20	4.66	4.63	4.59	4.66	4.62	4.61
120	500x20	1.68	1.70	1.72	1.71	1.70	1.71
Genel Ort.		1.84	1.77	1.85	1.87	1.96	1.96

*: 1 adet parçacığı ifade etmektedir.

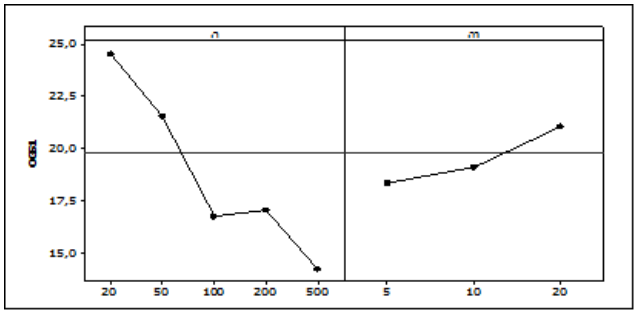
Tablo 5: NEH düzeyleri için istatistiksel sonuçlar.

Düzey	Friedman Ort. Sıra	Wilcoxon p_değeri
%10	2.46	-
%25	3.17	0.133
1	3.33	0.021
%50	3.79	0.006
%75	4.00	0.018
%100	4.25	0.033

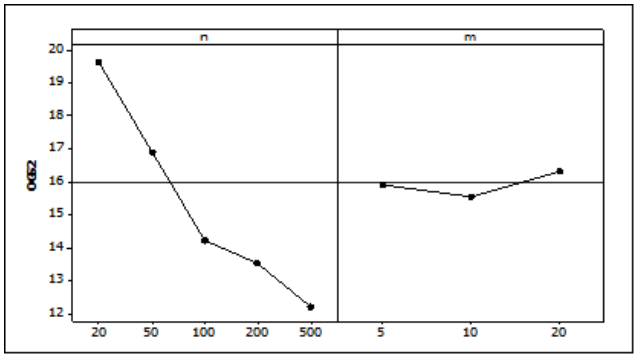
Tablo 6: Başlangıç çözümü ve son çözüm karşılaştırmaları.

	Başlangıç Çözümü				Son Çözüm							
Taillard	Min	Max	Ort.	SS	Min	Max	Ort.	SS	UB	OGS1	OGS2	OGS3
10	1.151	1.639	1.407	110	1.109	1.273	1.147	34	1.108	27.01	22.66	3.55
20	1.653	2.238	1.972	130	1.609	1.809	1.662	48	1.591	23.95	18.67	4.45
30	2.277	2.966	2.672	159	2.196	2.482	2.273	66	2.178	22.67	17.56	4.34
40	2.790	3.589	3.209	177	2.782	3.004	2.829	45	2.782	15.34	13.42	1.69
50	3.257	4.096	3.773	199	3.178	3.394	3.214	37	3.065	23.11	17.40	4.87
60	4.079	5.111	4.742	246	3.909	4.226	3.956	49	3.756	26.26	19.87	5.33
70	5.341	6.453	6.002	268	5.330	5.600	5.376	48	5.322	12.78	11.65	1.01
80	5.918	7.249	6.758	314	5.903	6.137	5.950	45	5.845	15.62	13.57	1.80
90	6.677	8.390	7.845	418	6.633	6.828	6.677	41	6.434	21.93	17.50	3.77
100	10.807	12.890	12.159	493	10.769	10.947	10.804	35	10.675	13.91	12.54	1.21
110	11.869	14.298	13.568	601	11.807	11.976	11.845	37	11.284	20.24	14.55	4.97
120	26.984	31.535	30.230	1.132	26.906	27.073	26.940	33	26.457	14.26	12.22	1.82
							Genel Ortalama			19.76	15.97	3.23

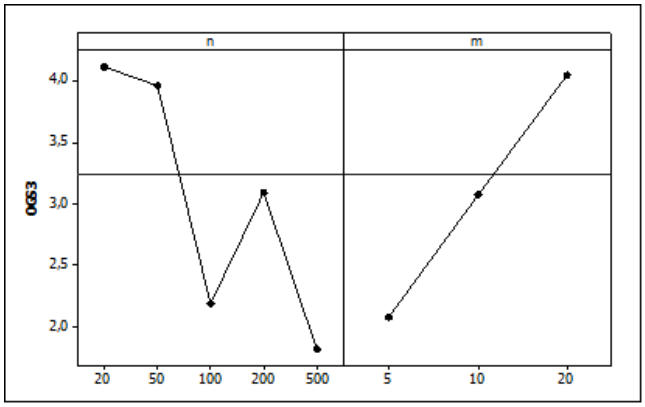
Şekil 5(a), (b) ve (c)'de iş sayısı (n) ve makine sayısı (m) faktörlerine karşılık sırasıyla *OGS1*, *OGS2* ve *OGS3*'teki değişimi gösteren ana etki grafikleri görülmektedir. Grafiklerde hemen hemen benzer bir görüntü olduğu söylenebilir. İş sayısı arttıkça *OGS* değerleri düşmektedir. İş sayısının 200 olduğu düzeyde biraz farklı sonuçlar alınmış olsa da genel eğilim düşüş yönündedir. Makine sayısının artması ise *OGS* değerlerinde artışa neden olmaktadır. Ancak bu artışın eğimi iş sayısındaki eğime göre daha düşüktür.



(a): OGS1 iş ve makine sayısı ana etki grafiği.

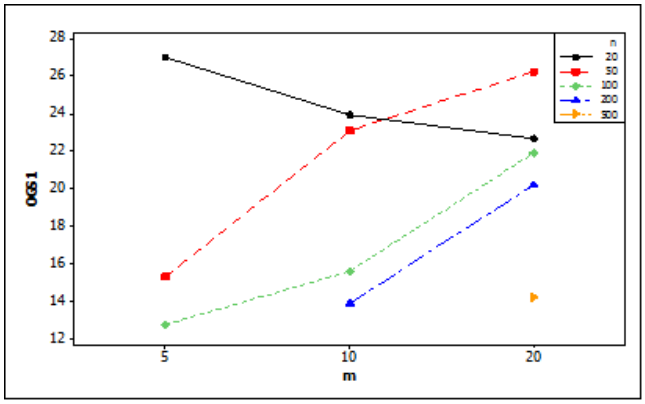


(b): OGS2 iş ve makine sayısı ana etki grafiği.

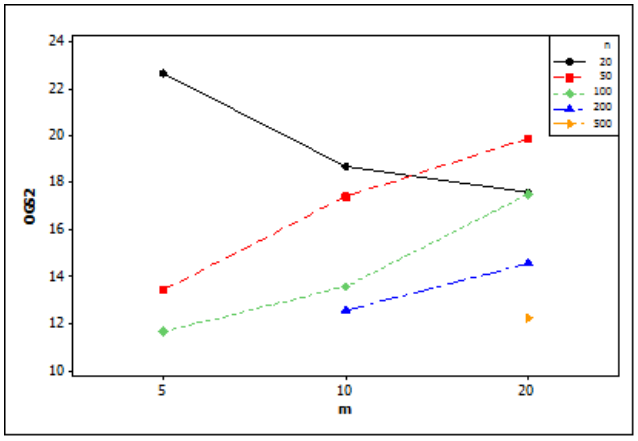


(c): OGS3 iş ve makine sayısı ana etki grafiği.

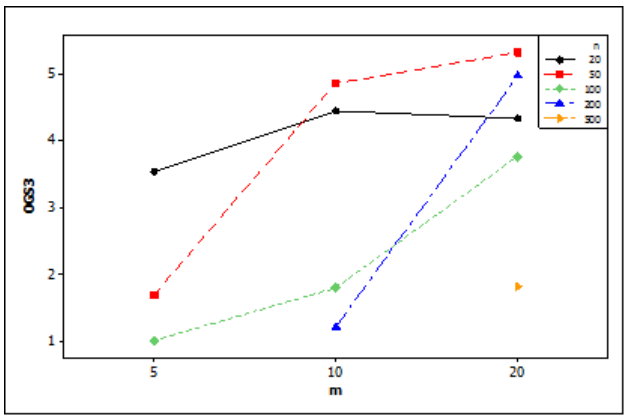
Şekil 5(d), (e) ve (f)'de ise *OGS* değerlerine karşılık iş sayısı ve makine sayısı faktörlerinin etkileşim durumları görülmektedir.



(d): OGS1 iş ve makine sayısı etkileşim grafiği.



(e): OGS2 iş ve makine sayısı etkileşim grafiği.



(f): OGS3 iş ve makine sayısı etkileşim grafiği.

Şekil 5: İş sayısı ve makine sayısı ana etki ve etkileşim grafikleri.

20 işin olduğu düzey yani görece az işin olduğu düzey daha farklı sonuçlar üretse de genel olarak iş sayısı ve makine sayısı faktörleri arasında, *OGS* değerleri açısından ciddi bir etkileşim olmadığı sonucuna varılabilir. Sonuçta iş sayısı arttıkça *OGS* değerleri düşmekte, makine sayısı arttıkça *OGS* değerleri artmaktadır. Yani iş sayısının fazla makine sayısının görece az olduğu problemlerde sapma değerleri daha düşük değerler almış. Bu tür problemlerde başlangıç çözümü daha da etkili olmuştur denilebilir.

6 Sonuçlar ve öneriler

Bu çalışmada meta sezgisel yöntemlerle akış tipi çizelgeleme problemlerinin çözümünde başlangıç popülasyonu belirleme yöntemlerinin etkisini gözlemlemek amaçlanmıştır. Bu doğrultuda HAPSO algoritmasının akış tipi çizelgelemeye uyumlu hali olan AHAPSO algoritmasının başlangıç popülasyonlarını oluştururken ST, NEH10, SKNEH1, SKNEH10 ve SK olmak üzere beş farklı başlangıç popülasyonu belirleme yöntemi uygulanmış ve karşılaştırılmıştır. Akış tipi çizelgeleme problemi olarak Taillard'ın [46] akış tipi çizelgeleme problemi veri setlerinde yer alan 12 problem grubundaki toplam 120 problem kullanılmıştır.

Elde edilen sonuçlar ele alındığında rastgele başlangıç popülasyonu oluşturma (ST) durumunda en yüksek OGS değeri elde edilmiştir ki bu AHAPSO algoritmasının doğal halidir. Bu da başlangıç popülasyonlarının rastgele oluşturulmasının optimum çözüme ulaşmada kısıtlı kaldığını göstermektedir. Bunun yanında rastgele seçilen başlangıç popülasyonlarına EKİZ ve EBİZ durumları (SK) eklendiğinde OGS değerinde az da olsa iyileşme sağlandığı görülmüştür.

NEH10, SKNEH1, SKNEH10 yöntemleriyle başlangıç popülasyonlarının oluşturulduğunda ise en başarılı OGS değerleri elde edilmiştir. NEH algoritması ile başlangıç popülasyonları belirlemenin akış tipi çizelgeleme probleminin AHAPSO algoritması ile çözümüne önemli katkılar sağladığı gözlemlenmiştir. Ayrıca başlangıç popülasyonu oluştururken NEH algoritmasına sıralama kurallarının adapte edilmesi her ne kadar istatistiksel olarak katkı sağlamış gibi görünmese de sayısal olarak az da olsa daha küçük sonuçlar üretilmiştir. Bu durumda diğer meta sezgisel yöntemleri başlangıç popülasyonları ile geliştirirken NEH algoritması kullanılabileceği gibi sıralama kurallarını da göz ardı etmemek gerekmektedir.

Öte yandan NEH algoritmasının farklı kullanım düzeylerinin çözüme etkisini test edebilmek için 12 problem grubundan seçilen birer problem üzerinde 120 parçacıklı popülasyonun, sırasıyla 1 parçacığı, %10'u, %25'i, %50'si ve %100'ü NEH algoritmasıyla oluşturulmuştur. Buradan elde edilen sonuç %10 ve %25 NEH düzeylerinin diğer düzeylerden daha iyi sonuçlar ürettiği ancak istatistiksel olarak birbirlerinden ayrışmadığıdır. Bununla beraber ortalama sıra ve genel ortalama OGS değerleri de göz önüne alındığında %10 NEH düzeyinin gayet olumlu sonuçlar üreten ve tercih edilebilecek bir düzey olduğu söylenebilir.

Başlangıç çözümü ile son çözüm değerlerine bakılarak başlangıç çözümünün ve AHAPSO algoritmasının performansına bakıldığında özellikle iş sayısının arttığı durumlarda başlangıç çözümünün daha da başarılı olduğu *OGS* değerlerinden anlaşılmaktadır. Öte yandan AHAPSO algoritmasının da iyi bir başlangıç çözümünden sonra *UB*'den sapma anlamında başarılı değerler üretebildiği gözlemlenmiştir.

Bundan sonraki çalışmalarda, popülasyondaki parçacık sayısı ve iterasyon sayısı parametreleri için deney tasarımı yapılabilir, adaptif modeller üzerine çalışmalar yapılabilir. Akış tipi çizelgeleme problemlerinin çözümünde kullanılacak diğer meta sezgisel yöntemlerin başlangıç popülasyonlarının çözüm kalitesine etkisi araştırılabilir. Ayrıca diğer üretim tiplerindeki çizelgeleme problemlerinin HAPSO veya AHAPSO algoritmalarıyla çözümleri üzerine çalışılabilir.

7 Teşekkür

Bu çalışma TÜBİTAK tarafından 118E355 numaralı, "Akış Tipi Çizelgeleme Probleminin Yeni Kaotik Metasezgisel Optimizasyon Algoritmaları ile Çözülmesi" başlıklı proje ile desteklenmiştir. Katkılarından dolayı teşekkür ederiz.

8 Kaynaklar

[1] Yağmahan B, Yenisey MM. "Akış tipi çizelgeleme problemi için KKE parametre eniyileme". *İTÜ Dergisi*, 5(2), 133-141, 2006.

[2] Kaya S, Fığlalı N. "Çok amaçlı esnek atölye tipi çizelgeleme problemlerinin çözümünde meta sezgisel yöntemlerin kullanımı". *Harran Üniversitesi Mühendislik Dergisi*, 3(3), 222-233, 2018.

[3] Aydilek İB. "A hybrid firefly and particle swarm optimization algorithm for computationally expensive numerical problems". *Applied Soft Computing*, 66, 232-249, 2018.

[4] Johnson SM. "Optimal two and three stage production schedules with setup time ıncluded". *Naval Research Logistics Quarterly*, 1(1), 61-68, 1954.
[5] Palmer D. "Sequencing jobs through a multi-stage process in the minimum total time-a quick method of obtaining a near optimum". *Operational Research Quarterly*, 16(1), 101-107, 1965.
[6] Campbell HG, Dudek RA, Smıth BL. "A heuristic algorithm for the n job, m machine sequencing problem". *Management Science*, 16(10), 630-637, 1970.
[7] Gupta JND. "A Functional heuristic algorithm for flow-shop scheduling problem". *Operations Research*, 22, 39-47,1971.
[8] Dannenbring DG. "An evaluation of flow-shop sequencing heuristic". *Management Science*, 23(11), 1174-1182, 1977.
[9] Nawaz M, Enscore EJ, Ham I. "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem". *Omega, the International Journal Of Management Science*, 11(1), 91-95, 1983.
[10] Hundal TS, Rajgopal J. "An extension of palmer's heuristic for the flow shop scheduling problem". *International Journal of Production Research*, 26, 1119-1124, 1988.
[11] Widmer M, Hertz A. "A new heuristic method for the flowshop sequencing problem". *European Journal of Operational Research*, 41, 186-193, 1989.
[12] Ho JC, Chang Y. "A new heuristic for the n-Job, m-Machine flow-shop problem". *European Journal of Operational Research*, 52(2), 194-202, 1991.
[13] Engin O, Fığlalı A. "Akış tipi çizelgeleme problemlerinin genetik algoritma yardımı ile çözümünde uygun çaprazlama operatörünün belirlenmesi". *Doğuş Üniversitesi Dergisi*, 6, 27-35, 2002.
[14] Janiak A. "General Flow-shop scheduling with resource constraints". *International Journal of Production Research*, 26(6), 1089-1093, 1988.
[15] Tandon M, Cummings PT, LeVan MD. "Flowshop sequencing with non-permutation schedules". *Computing Chem Engineering,* 15(8), 601-607, 1991.
[16] Benavides AJ, Ritt M. "Two simple and effective heuristics for minimizing the makespan in non-permutation flow shops". *Computer and Operation Research*, 66, 160-169, 2016.
[17] Cui WW, Lu Z, Zhou B, Li C, Han X. "A hybrid genetic algorithm for non-permutation flow shop scheduling problems with unavailability constraints". *International Journal Computing Integrated Manufacturing*, 29(9),1-18, 2016.
[18] Henneberg M, Neufeld JS. "A constructive algorithm and a simulated annealing approach for solving flowshop problems with missing operations". *International Journal Production Research*, 54(12), 3534-50, 2016.
[19] Pugazhenthi R, Xavior MA. "Computation of Makespan Using Genetic Algorithm in a Flowshop". *American-Eurasian Journal of Scientific Research*, 9, 105-113, 2014.
[20] Benavides AJ, Ritt M. "Iterated local search heuristics for minimizing total completion time in permutation and non-permutation flow shops". *TwentyFifth International Conference on Automated Planning and Scheduling ICAPS*, Jerusalem, Israel, 7-11 June 2015.
[21] Reeves CR, Yamada T. "Genetic algorithm path relinking and the flow shop sequencing problem". *Evolutionary Computation*, 6, 45-60, 1998.
[22] Rajendran C, Ziegler H. "Ant-colony algorithms for permutation flow shop scheduling to minimize makespan total flowtime of jobs". *European Journal of Operational Research*, 155, 426-438, 2004.
[23] Taşgetiren F, Liang Y C, Sevkli M, Gencyilmaz G. "A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flow shop sequencing problem". *European Journal of Operational Research*, 177, 1930-1947, 2007.
[24] İşler M. Bulanık Esnek Akış Tipi Çizelgeleme Problemlerinin Paralel Doyumsuz Algoritma Ile Çözümü: Bir Hazır Giyim işletmesine Uygulanması. Yüksek Lisans Tezi, Selçuk Üniversitesi, Konya, Türkiye, 2009.
[25] Alaykıran K, Engin O, Döyen A. "Using Ant Colony Optimization to Solve Hybrid Flow Shop Scheduling Problems". *International Journal Advanced Manufacturing Technology*, 35, 541-550, 2007.
[26] Kahraman C, Engin O, Kaya İ, Yılmaz MK. "An application of effective genetic algorithms for solving hybrid flow shop scheduling problems". *International Journal of Computational Intelligence Systems*, 1(2), 134-147, 2008.
[27] Kianfar K, Ghomi SMTF, Jadid AO. "Study of stochastic sequence-dependent flexible flow shop via developing a dispatching rule and a hybrid GA". *Engineering Applications of Artificial Intelligence*, 25, 494-506, 2012.
[28] Jolai F, Asefi H, Rabiee M, Ramezani P. "Bi-objective simulated annealing approaches for no-wait two-stage flexible flow shop scheduling problem". *Scientia Iranica*, 20, 861-872, 2013.
[29] Keskin K. Beklemesiz Akış Tipi Çizelgeleme Problemlerinin Çok Amaçlı Genetik Algoritma ile Çözümü, Yüksek Lisans Tezi, Selçuk Üniversitesi, Konya, Türkiye, 2010.
[30] Laha D, Sapkal SU. "An improved heuristic to minimize total flow time for scheduling in the m-machine no-wait flow shop". *Computers & Industrial Engineering*, 67, 36-43, 2014.
[31] Tseng L, Lin Y. "A hybrid genetic algorithm for no-wait flow shop scheduling problem". *International Journal of Production Economics*, 128, 144-152, 2010.
[32] Chaudhry IA, Mahmood S. "No-wait Flow shop Scheduling Using Genetic Algorithm". No-wait Flowshop Scheduling Using Genetic Algorithm, 3, 4-6, 2012.
[33] Czogalla J, Fink A. "Design and analysis of evolutionary algorithms for the no-wait flow shop scheduling problem". *Metaheuristics in the Service Industry, Lecture Notes in Economics and Mathematical Systems*, 624, 99-126, 2009.
[34] Pan QK, Wang L, Qian B. "A novel differential evolution algorithm for bi criteria no wait flow shop scheduling problems". *Computers & Industrial Engineering*, 36, 2498-2511, 2009.
[35] Moghaddam RT, Vahed ARR, Mirzaei AH. "Solving a multi-objective no-wait flow shop scheduling problem with an immune algorithm". *International Journal Advanced Manufacturing Technology,* 36, 969-981, 2008.
[36] Allahverdi A, Aldowaisan T. "No-wait flowshops with bicriteria of makespan and maximum lateness". *European Journal of Operational Research*, 152, 132-147, 2004.
[37] Araújoa DC, Nagano MS. "A new effective heuristic method for the no-wait flowshop with sequence-dependent setup times problem". *International Journal of Industrial Engineering Computations*, 2, 155-166, 2011.

[38] Kumar G, Singhal S. "Genetic Algorithm Optimization of Flowshop Scheduling Problem with Sequence Dependent Setup Time and Lot Splitting". *International Journal of Engineering, Business and Enterprise Applications (IJEBEA)*, 4(1), 62-71, 2013.

[39] İşler MC, Çelik V, Toklu B. "İki makine akış tipi öğrenme etkili çizelgelemede ortak teslim tarihinden mutlak sapmaların en küçüklenmesi". *Journal Faculty Engineering Arch Gazi University*, 24(2), 351-357, 2009.

[40] Akçay E. Akış Tipi İş Çizelgeleme Problemlerinin Yapay Bağışıklık Sistemi İle Çok Amaçlı Optimizasyonuna Yönelik Bir Model Önerisi. Doktora Tezi, Kocaeli Üniversitesi, Kocaeli, Türkiye. 2009.

[41] Rossit DA, Tohméb F, Frutos M. "The Non-Permutation Flow-Shop scheduling problem: A literature review". *Omega*, 77, 143-153, 2018.

[42] Yang XS. *Firefly algorithms for multimodal optimization*. Editors: Watanabe O, Zeugmann T. Stochastic Algorithms: Foundations and Applications, 169-178. Berlin, GERMANY, Springer, 2009.

[43] Kennedy J, Eberhart R. "Particle Swarm Optimization". *Proceedings of IEEE International Conference on Neural Networks*, 4, 1942-1948, 1995.

[44] Xin J, Chen G, Hai Y. "A Particle Swarm Optimizer with Multistage Linearly-Decreasing Inertia Weight". *International Joint Conference on Computational Sciences and Optimization (CSO-2009*), Sanya, Hainan, China. 2 4-26 April 2009.

[45] Bean JC. "Genetic algorithm and random keys for sequencing and optimization". *ORSA journal on computing*, 6(2), 154-160, 1994.

[46] Taillard E. "Benchmarks For Basic Scheduling Problems". http://mistic.heig-ve diğ.ch/taillard/problemes.dir/ordonnancement.dir/ordonnancement.html (04.02.2019).

[47] Wang H, Wang W, Sun H, Cui Z, Rahnamayan S, Zeng S. "A new cuckoo search algorithm with hybrid strategies for flow shop scheduling problems". *Soft Computing*, 21, 4297-4307, 2017.