Gazi Mühendislik Bilimleri Dergisi
2018, 4(2): 115-123
Araştırma Makalesi/Research Article
gmbd.gazipublishing.com

Karınca Kolonisi ve Yapay Arı Kolonisi Algoritması Kullanarak Yazılım Proje Takvimi Uygulamasının Gerçekleştirilmesi

Nurhan GÜL [*,a], Nursal ARICI[b]

[a,*] *Gazi Üniversitesi Teknoloji Fakültesi Bilgisayar Mühendisliği Bölümü, ANKARA 06500, TÜRKİYE*
[b] *Gazi Üniversitesi Teknoloji Fakültesi Bilgisayar Mühendisliği Bölümü, ANKARA 06500, TÜRKİYE*

MAKALE BİLGİSİ

Alınma: 08.06.2018
Kabul: 24.07.2018

Anahtar Kelimeler:
Yazılım proje takvimi, Karınca kolonisi algoritması, Yapay arı kolonisi algoritması, Optimizasyon algoritması.

[*]*Sorumlu Yazar:*
e-posta: nurhangul13@gmail.com

ÖZET

Yazılım projelerinin başarıya ulaşma oranı teknolojik gelişmelere rağmen hala istenen düzeyde değildir. Yazılım projelerinin büyük çoğunluğu ya istenen özelliklerde teslim edilememekte ya da planlanan bütçeyi ve zamanı aşarak teslim edilebilmektedir. Yazılım proje takvimi bu başarıya etki eden önemli faktörlerden biridir. İnsan kaynağı, zaman, maliyet ve aktivitelerin işlem sırası gibi parametreler içerdiğinden dolayı, yazılım proje takvimi oluşturmada durum uzayı çok büyüktür. Bu nedenle de proje yöneticisinin insan çabasıyla başarılı bir proje takvimi oluşturması oldukça zordur. Bu çalışmada insan kaynağı ve insan kaynağının aktiviteleri gerçekleştirme süreleri ele alınarak yazılım proje takvimi, minimum tamamlanma süresini sağlayacak şekilde oluşturulmaya çalışılmıştır. Yöntem olarak yapay zeka optimizasyon algoritmalarından Karınca Kolonisi Algoritması ve Yapay Arı Kolonisi algoritması kullanılmıştır. Elde edilen analiz sonuçlarına göre, her iki yöntem minimum proje süresini elde etmede başarılı olmuştur. Yapay arı kolonisi algoritmasının işlem süresinin daha yavaş olmasına karşın, koloni/yiyecek kaynağı sayısı arttığında karınca kolonisi algoritmasına oranla sonuca daha hızlı yakınsadığı belirlenmiştir.

DOI: https://dx.doi.org/10.30855/GJES.2018.04.02.006

Software Project Scheduling Using Ant Colony and Artificial Bee Colony Algorithm

ARTICLE INFO

Received: 08.06.2018
Accepted: 24.07.2018

Keywords:
Software project scheduling, Ant colony optimization, Artificial bee colony algorithm, Optimization algorithm.

[*]***Corresponding Authors***
nurhangul13@gmail.com

ABSTRACT

The success rate of software projects is still not at the desired level despite of the technological advances. The vast majority of software projects can not be delivered at the desired specifications or can be delivered beyond the planned budget and time. The software project schedule is one of the important factors that influence this success. Due to it includes parameters such as human resource, time, cost, and process sequence of activities, state space is too big for software project scheduling. So it is difficult to create a software project schedule for software project managers. In this study, using human resource and activities these resource can do, it is tried to obtain minimum project completion time while creating software project schedule using ant colony and artificial bee colony optimization algorithm and results are analyzed. According the results obtained, both methods are successful in software project scheduling. Although the processing time of the artificial bee colony algorithm is slower than ant colony optimization algorithm, it has been determined that when the number of colony / food source is increased, it is converged to minimum project completion time faster than the ant colony algorithm.

DOI: https://dx.doi.org/10.30855/GJES.2018.04.02.006

1. GİRİŞ *(INTRODUCTION)*

Günümüzde yazılım projelerinin neredeyse sadece üçte biri başarı ile tamamlanabilmektedir. Teknolojik gelişmelerle kıyaslandığında bu başarı oranı henüz istenilen seviyeye ulaşamamıştır [1]. İhtiyaç duyulan yazılım projelerinin karmaşıklığının artması, yazılım projelerinin genelleştirilebilir bir sürece oturtulamaması gibi durumlar bu sonuca etki eden önemli faktörlerdir [2].

Yazılım proje takvimi, yazılım projelerinin başarı ile gerçekleştirilebilmesi için proje yöneticisi tarafından kullanılan önemli bir kontrol aracıdır. Ancak proje için belirlenen bütçenin ve tamamlanma zamanının aşılmaması koşuluyla; görevlerin uygun sırada işlenmesi, görev bağımlılıklarının yönetimi, insan kaynağı görev ataması gibi alt problemlerin varlığı yazılım proje takvimi oluşturma probleminin çözümü için gereken süreyi oldukça arttırmaktadır. Bu durum, problemin belirleyici olmayan zor polinom (NP-hard) sınıfına dahil olmasına neden olmaktadır [3-4].

NP-hard problemler için durum uzayında geleneksel yöntemlerle gerçekleştirilen arama işlemleri oldukça zor ve zaman alıcıdır [5]. Yapay zeka optimizasyon yöntemleri kullanılarak durum uzayındaki aramalar belirli bir düzen çerçevesinde rastgele ve zamanla optimum çözüme yakınsayacak şekilde gerçekleştirilebilmektedir. Böylelikle problemin çözümü kolaylaşmakta ve çözüm için gerekli işlem süresinde de önemli derecede iyileşmeler elde edilmektedir.

Xia, Ao, ve Tang (2013) çalışmalarında, yazılım proje takvimi oluşturulması probleminde kaynak atamalarını graf ile modelleyip, karınca kolonisi optimizasyon algoritmasını kullanarak görevlerin kaynaklara atanması problemine çözüm üretmişlerdir [6].

Chen ve Zhang (2013) çalışmalarında olay tabanlı yaklaşım kullanmışlardır. Proje takvimine doğrudan etkisi olan insan kaynaklarının hareketlerini (yeni personel eklenmesi, var olan personelin ayrılması vb.) olay olarak değerlendirip, karınca kolonisi optimizasyon algoritması ile kaynak atama problemini çözmeye çalışmışlardır [7].

Singh vd., (2010) ve Suri ve Singal (2011)'nin regresyen testi için senaryo seçiminde kullanıldıkları karınca kolonisi yöntemini [8,9], Suri ve Jajoria (2013) proje takviminin minimum sürede oluşturulması problemine uygulamışlardır [10].

Crawford vd. (2014) yazılım proje takvimi oluşturma problemini, maksimum minimum karınca sistemine feromon güncellemelerinin 0-1 aralığında olacak şekilde çok boyutlu olarak gerçekleştirildiği Hyper-Cube çatısını ekleyerek çözümlemişlerdir [11].

Akbari vd. (2011) çalışmalarında, kaynak kısıtlamalı proje takvimlendirme problemini (RCPSP), yapay arı kolonisi kullanarak modellemiş ve yapay arı kolonisinin bu problemdeki performansı üzerine çalışmalar gerçekleştirmişlerdir [12].

Nayak vd. (2012) çalışmalarında yapay arı kolonisi yöntemini, Rastrigin, Rosenbrock, Sphere ve Schwefel foksiyonlarında uygulayarak elde ettikleri sonuçları Parçacık Sürü Optimizasyon (PSO) yöntemi ile karşılaştırmışlardır [13].

Bansal vd. (2013) çalışmalarında, yapay arı kolonisi algoritmasındaki kontrol parametrelerini incelemiş ve benzer optimizasyon algoritmaları ile performans karşılaştırması yapmışlardır [14].

Pauzi (2015) yapay arı kolonisi yaklaşımını kullanarak iş akışı takvimlendirme problemine çözüm aramış ve yapay arı kolonisi için gözcü arı yaklaşımını değiştirerek yeni bir öneri sunmuştur [15].

Pauzi vd. (2016) çalışmalarında, yapay arı kolonisi algoritmasında gözcü arıların yiyecek kaynaklarına dağıtılması üzerine yaptıkları incelemede, gözcü arıların her birinin farklı kaynaklara dağıtılması yerine en iyi kaynaklarda yoğunlaşmasına dayalı bir yöntem üzerinde çalışmışlardır [16].

Bu çalışmada insan kaynağı ve insan kaynağının aktiviteleri gerçekleştirme süreleri ele alınarak yazılım proje takvimi, minimum proje tamamlanma süresini sağlayacak şekilde oluşturulmaya çalışılmıştır. Yöntem olarak yapay zeka optimizasyon algoritmalarından Karınca Kolonisi Algoritması (KKA) ve Yapay Arı Kolonisi Algoritması (YAKA) kullanılmış ve sonuçlar analiz edilmiştir. Elde edilen sonuçlara göre, her iki yöntem minimum proje süresini elde edecek ekibi belirlemede başarılı olmuştur. YAKA'nın yanıt zamanının daha yavaş olmasına karşın, koloni/yiyecek kaynağı sayısı arttığında karınca

kolonisi algoritmasına oranla sonuca daha hızlı yakınsadığı belirlenmiştir.

Çalışmanın bölümleri şu şekildedir: İkinci bölüm yazılım proje takvimi oluşturmak için kullanılan KKA ve YAKA'nın detaylarını, üçüncü bölüm elde edilen deneysel sonuçları, dördüncü bölüm ise sonuç kısmını içermektedir.

2. YÖNTEM *(METHOD)*

Yazılım proje takvimi oluşturma problemi NP-hard yapıdadır [6,7,10]. Aktivitelerin belirli sırada işlenmesi, insan kaynaklarının aktivitelere atanması, boşta kalan insan kaynaklarının verimli kullanılması, proje süresinin ve maliyetinin minimum düzeyde tutulması gibi durumlar çözüm uzayını çok büyük yapmaktadır. Bu nedenle çözüm uzayını daraltmak amacıyla çeşitli optimizasyon yöntemleri kullanılmaktadır [17].

Bu çalışmada çözüm uzayını daraltmak için yapay zeka optimizasyon yöntemlerinden KKA ve YAKA kullanılmıştır. Çalışmada insan kaynakları, insan kaynaklarının yapabildiği proje aktiviteleri ve yapabildikleri proje aktivitelerini gerçekleştirme süreleri kullanılarak yazılım proje takviminin minimum süreyi içerecek şekilde oluşturulması amaçlanmaktadır. Giriş verisinin örnek formatı Tablo 1'de, açıklamaları ise Tablo 2'de belirtilmiştir.

Tablo 1. Örnek Giriş Verisi (*Input Data Sample*)

E/A	A0	T0	A1	T1	A2	T2	A3	T3
E0	0	0	1	5	1	2	0	0
E1	1	2	0	0	1	5	0	0
E2	0	0	0	0	0	0	0	0
E3	0	0	0	0	0	0	1	4
E4	1	3	1	3	0	0	0	0

Projenin tamamlanması için projeye ait tüm aktivitelerin seçilen insan kaynakları tarafından yapılabiliyor olması gerekmektedir.

$$_{\text{PrjTeam}}\{A_{n_1}, A_{n_2}, \dots A_{n_m}\} \supseteq Prj\{A\} \quad (1)$$

$$_{\text{PrjTeam}} \subseteq \{E_1, E_2, \dots E_n\} \quad (2)$$

Tablo 2. KKA Parametreleri (*KKA parameters*)

Parametre	Açıklama
E=$\{E_0..E_k\}$	Projede görevli insan kaynağı
A=$\{A_0..A_l\}$	Projedeki aktivite
T=$\{T_0..T_l\}$	İnsan kaynağının aktiviteyi gerçekleştirme süresi
Prj	Takvimi oluşturulacak proje
PrjTeam	Döngü sonunda oluşturulan proje ekibi
n	Döngü sonunda belirlenen proje ekibindeki insan kaynağı sayısı
M	Döngü sonunda belirlenen proje ekibinin yapabildiği aktivite sayısı

2.1 KKA ile Yazılım Proje Takvimi Oluşturma (*Software Project Scheduling Using KKA*)

KKA, karıncaların doğadaki davranışlarından yararlanılarak 1990'lı yılların başında modellenmiştir [18-20].

Karıncalar yiyeceğe ulaşırken zamanla buharlaşma özelliği gösteren feromon denilen bir sıvı salgılarlar. Kolonideki bir karınca yiyecek ararken ortamdaki feromon maddesini kontrol eder. Feromonun yoğun olduğu yöne doğru ilerleyerek önceki karıncaların bulduğu yiyecek kaynaklarına doğru yol alır.

Feromon maddesi zamanla buharlaşma özelliği gösterir. Kolonideki bir karıncanın bulduğu yiyecek kaynağı diğer bir karıncanın bulduğu yiyecek kaynağından uzakta ise bu karıncanın ilerlediği yoldaki feromon miktarı diğer karıncanın ilerlediği yoldaki feromon miktarına göre daha az olur. Bu durum kolonideki karıncaların yakın mesafedeki yiyecek kaynağında yoğunlaşmasını sağlar. Bu davranış şekli modellenerek KKA geliştirilmiştir ve yöntem durum uzayı büyük olan birçok probleme uygulanmıştır [6-10,21].

Projenin tüm aktivitelerini minimum sürede tamamlayacak uygun proje ekibinin belirlenmesi için bu çalışmada kullanılan KKA'ya ait detaylar aşağıda belirtilmiştir:

- Maksimum minimum karınca sistemi kullanılmıştır [21-23].
- Projede yer alan insan kaynakları, karıncaların ilerleyeceği düğümlerle temsil edilmektedir.
- Her karınca başlangıçta rastgele bir insan kaynağı seçerek projeyi başlatır.
- Ortamda feromon varsa sonraki ilerlemeler, feromon fazlalığı olan insan kaynaklarına doğru

olur. Feromon yoksa henüz seçilmemiş insan kaynaklarından biri seçilerek projeye dahil edilir.

- Karıncaların ilerlemesi, projenin tüm aktivitelerini tamamlayan proje ekibinin oluşturulmasına kadar devam eder.
- İlerlemeler tamamlandığında tüm karıncaların proje süreleri hesaplanarak minimum süreyi elde eden karınca belirlenir.
- Döngü sonunda minimum süreyi elde eden karıncaların ilerlediği yollar için lokal ve global feromon güncellemesi yapılır [11]. Feromon güncellemesi formüllerine ilişkin açıklamalar Tablo 3'te belirtilmiştir.

Tablo 3. Feromon Güncelleme Parametreleri (Parameters for Pheromone Update)

Parametre	Açıklama
$Phr_{i,j}$	Projede görevlendirilen insan kaynaklarının temsil ettiği i ↔ j düğümleri arasındaki feromon miktarı
PTime	Döngü sonunda karıncanın elde ettiği proje süresi
AntNode	Minimum süreyi elde eden karıncanın ilerlediği düğümler
GPTime	O ana kadar elde edilen minimum proje süresi
GAntNode	O ana kadarki minimum proje süresini elde eden karıncanın ilerlediği düğümler
n	Döngü sonunda belirlenen proje ekibindeki insan kaynağı sayısı
M	Döngü sonunda belirlenen proje ekibinin yapabildiği aktivite sayısı
α, β, μ	Kontrol parametreleri

Lokal feromon güncelleme:

$$PTime_{Ant}=\sum_{k=1}^{n}\sum_{l=1}^{m} T_l \quad (3)$$

$$Phr_{i,j}(t)=\begin{cases}\mu * Phr_{i,j}(t-1) i,j \in \{AntNode\}, 0<\mu<1\\ 0, \quad else\end{cases} \quad (4)$$

Global feromon güncelleme:

$$Phr_{i,j}(t)=\begin{cases}\frac{1-\alpha}{\beta * \mathrm{GPTime}}, i,j \in \{GAntNode\}, 0<\alpha,\beta<1\\ 0, \quad else\end{cases} \quad (5)$$

- Her bir döngü sonunda tüm karıncaların ilerlediği yollar için belirli bir oranda (£) feromon buharlaştırması yapılarak başarısız yolların tercih edilebilirliği azaltılır [8-10]:

$$\Delta(t) = £ * Phr_{i,j}(t-1),\ Phr_{i,j}(t-1) > 0 \quad (6)$$

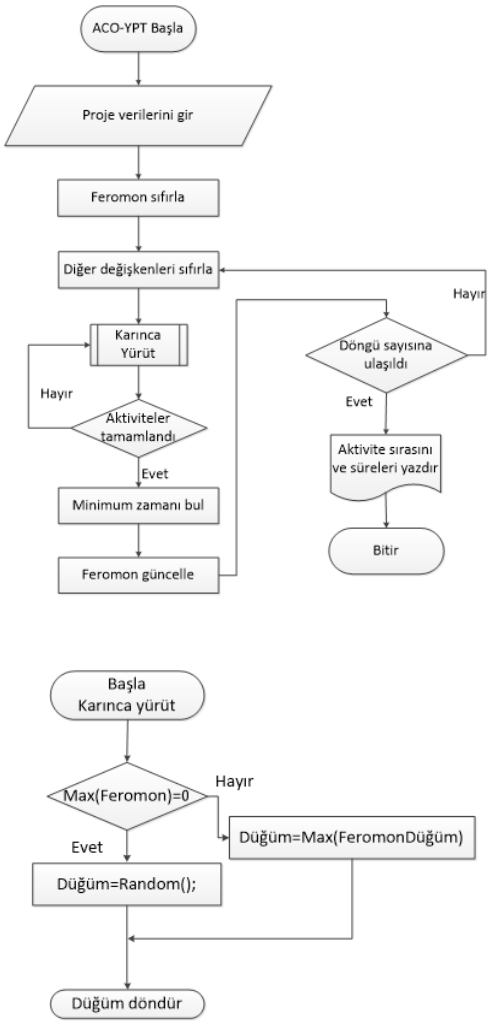
- Yapılan güncellemeler sonrası feromon miktarı maksimum ve minimum sınırları aşmışsa feromon miktarı bu sınırlara çekilir.

$$\max(t) = 1/(\partial * min\{PTime_{Ant}(t)\}) \quad (7)$$

$$\min(t) = \max(t) / (2 * n) \quad (8)$$

$$Phr_{i,j}(t)=\begin{cases}\max(t), & Phr_{i,j}(t) > max\\ \min(t), & Phr_{i,j}(t) < min\end{cases} \quad (9)$$

Başlangıç feromon miktarının sıfır alındığı yönteme ait akış diyagramı Şekil 1'de yer almaktadır.



Şekil 1. KKA ile yazılım proje takvimi oluşturma (Software project scheduling using KKA) [24].

2.2. YAKA ile Yazılım Proje Takvimi Oluşturma *(Software Project Scheduling Using YAKA)*

YAKA, çoğu optimizasyon tekniğinde olduğu gibi doğadan esinlenilerek arıların yiyecek bulma tekniğinin modellendiği bir optimizasyon tekniğidir.

2005 yılında Derviş Karaboğa tarafından geliştirilen yöntemde başlangıçta kaşif arılar rastgele yiyecek ararlar [25]. Yiyecek kaynağına ulaşan arı, artık işçi arı olur ve yiyecek kaynağını kovana taşımaya başlar. Kovana ulaşan işçi arı ya tekrar yiyecek taşımaya döner ya da bulduğu yiyecek kaynağını sergileyeceği dans ile kovanda yiyecek kaynağının lokasyonuna ilişkin bilgi bekleyen gözcü arılara iletir.

Gözcü arılar, yapılan dansı yorumlayarak yiyecek kaynağının kalitesi ve mesafesi ile ilgili bilgiyi alır ve kaynak tercihini bu bilgiye göre yaparlar. Arılar bu şekilde yiyecek kaynaklarından kovana yiyecek taşırlar.

Yiyecek kaynağı tükendiğinde de bu yiyecekleri kovana taşıyan işçi arılar kaşif arıya dönüşür ve yeni yiyecek kaynakları bulmaya çalışır [25-27].

Yapılan çalışmada YAKA'ya yazılım proje takvimi oluşturulması problemine özel olarak aşağıdaki yaklaşımlar eklenmiştir:

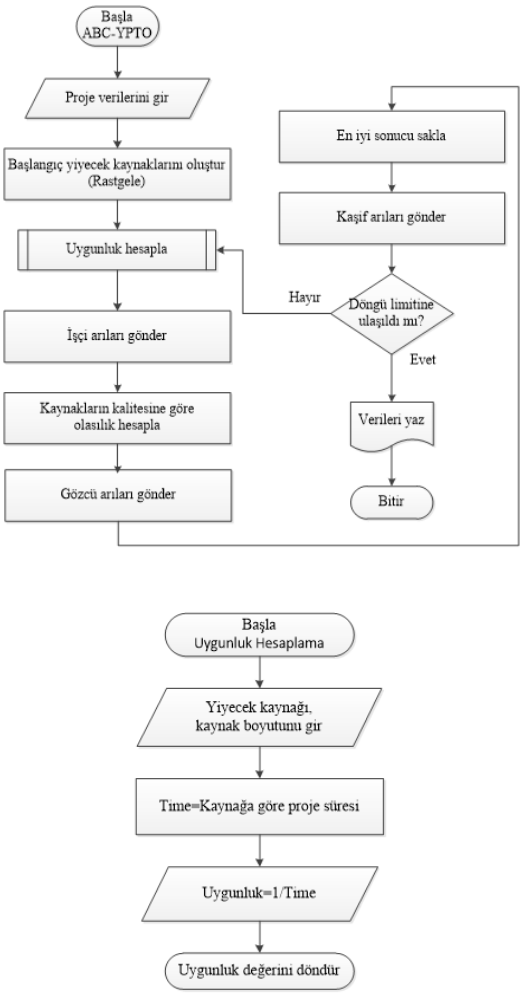
- **Dinamik boyut:** Mevcut problemde yiyecek kaynaklarının boyutu sabittir. Problemin yapısı itibariyle yiyecek kaynaklarının boyutu dinamik olarak oluşturulmuştur ve güncellenmiştir.

- **Dinamik çözüm uzayı:** İşçi ve gözcü arıların yeni yiyecek kaynağı belirleme süreci probleme özel olarak dinamik olarak belirlenmektedir. Yeni yiyecek kaynakları, insan kaynağı ekleme, güncelleme veya çıkarma yöntemleri ile üretilmektedir.

YAKA ile yazılım proje takvimi oluşturma yöntemine ait akış diyagramı Şekil 2'de yer almaktadır.

Yapay arı kolonisi ile yazılım proje takvimi oluşturulması amacıyla geliştirilen yönteme ilişkin temel kabuller ve algoritmaya ait detaylar aşağıda belirtilmiştir:

- İşçi arı sayısı, gözcü arı sayısına eşittir.
- Yiyecek kaynağı, projeyi tamamlaması için seçilen insan kaynaklarını temsil eder.
- Bir yiyecek kaynağı, projeyi tamamlamaya aday insan kaynakları grubundan oluşur. Örneğin başlangıçta boyutu rastgele 4 olarak belirlenen bir yiyecek kaynağında E2, E4, E6, E9 gibi 4 insan kaynağı bilgisi tutulur.



Şekil 2. YAKA ile yazılım proje takvimi oluşturma *(Software project scheduling using YAKA).*

- Her bir yiyecek kaynağı başlangıçta rastgele uzunlukta birbirinden farklı rastgele insan kaynağı içerecek şekilde dinamik boyutlu olarak oluşturulur.
- **İşçi arı fazı:**
    - Her bir yiyecek kaynağı için, rastgele belirlenen komşu yiyecek kaynakları kullanılarak yeni yiyecek kaynakları üretilir. Bu işlem projeye insan kaynağı ekleme, projeden insan kaynağı azaltma ve var olan insan kaynağını farklı bir insan kaynağı ile değiştirme şeklinde gerçekleştirilir.

$$PrjTeam=F_{Src} \subseteq \{E_1, E_2, \dots E_n\} \quad (10)$$

- Üretilen yeni yiyecek kaynağının kalitesi yani uygunluk değeri, yeni insan kaynaklarının projeyi tamamlayıp tamamlayamadığı kontrol edilerek, tamamlayabiliyorsa ne kadar sürede tamamlayabildiğine göre belirlenir.

$$PTime_{FSrc}=\sum_{k=1}^{n}\sum_{l=1}^{m} T_l \quad (11)$$

$$Fitness_{FoodSource}=1/PTime_{FSrc} \quad (12)$$

- Yeni yiyecek kaynağı mevcut yiyecek kaynağından kaliteli ise bu durum yeni ekibin projeyi mevcut ekipten daha kısa sürede tamamlayabildiği anlamına gelir. Bu durumda eski yiyecek kaynağı yeni yiyecek kaynağı ile değiştirilir, değilse bu yiyecek kaynağına ait iyileşememe sayacı 1 arttırılır.

- Tüm yiyecek kaynakları için temsil ettiği proje ekibinin proje tamamlama süresine göre hesaplanmış uygunluk değeri kullanılarak rulet tekerleği oluşturulur. Rulet tekerleği, her bir yiyecek kaynağının uygunluk değerinin o döngüdeki maksimum uygunluk değerine oranı ile belirlenir. Böylelikle, kaliteli yiyecek kaynağının uygunluk değeri daha fazla olduğundan rulet tekerleğinde temsil ettiği alan daha geniş olacak ve bu durumda da bu yiyecek kaynağının seçilme olasılığı diğer yiyecek kaynaklarına göre daha fazla olacaktır.

$$r_i(t) = Fitness_i(t) / \sum_{k=1}^{n} Fitness_k(t) \quad (13)$$

- **Gözcü arı fazı:**
  - Rulet tekerleği kullanılarak gözcü arıların daha kaliteli yiyecek kaynaklarına yönelmesi sağlanır.
  - Yiyecek kaynağına ulaşıldıktan sonra, işçi arı fazındaki işlemlere benzer şekilde kaliteli yiyecek kaynağı arama süreci gerçekleştirilir.

- **Kaşif arı süreci:**
  Yiyecek kaynağına ait iyileşememe sayacı limit değerine ulaşmışsa, bu yiyecek kaynağı başlangıçtaki sürece benzer şekilde rastgele uzunlukta rastgele insan kaynakları içerecek yapıda yeniden üretilir.

Tüm bu süreç, işçi arı fazından başlatılarak belirli sayıda tekrarlanıp en iyi sonuç elde edilmeye çalışılır.

Yöntemde kullanılan formüllerin açıklamaları Tablo 4'te yer almaktadır.

Tablo 4. YAKA Parametreleri (*Parameters in YAKA)*

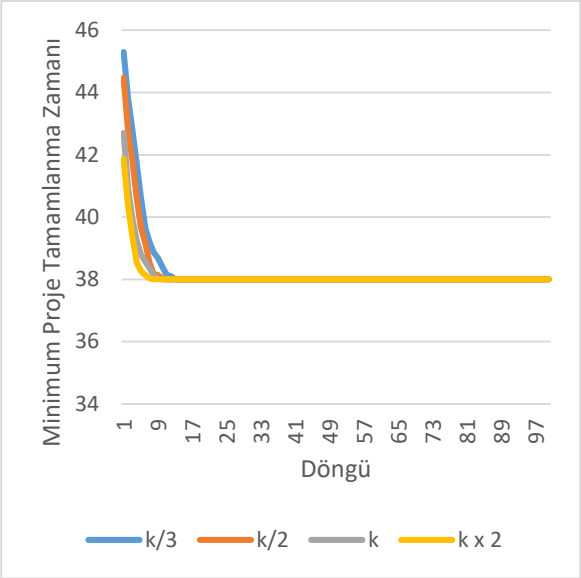
Parametre	Açıklama
FSrc	Yiyecek kaynağı
PTime	Döngü sonunda yiyecek kaynaklarından elde edilen proje süresi
Fitness	Yiyecek kaynağının kalitesi(uygunluk değeri)
n	Yiyecek kaynağı boyutu
m	Projedeki tamamlanması gereken aktivite sayısı

2.3. Yöntemlerin Karşılaştırması *(Comparison of Methods)*

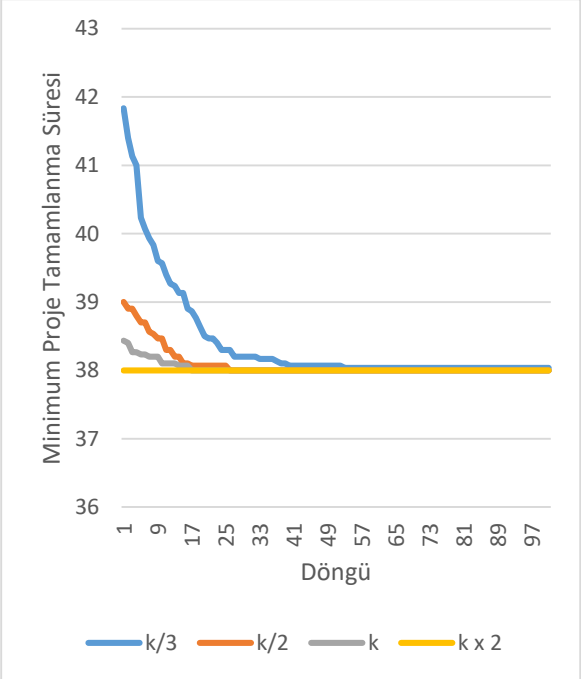
KKA ve YAKA ile yazılım proje takviminin minimum sürede tamamlanması amacıyla iki ayrı yazılım geliştirilmiştir. 18 insan kaynağı, 13 aktivite ve insan kaynaklarının bu aktivitelerden hangilerini ne kadar sürede gerçekleştirdiği bilgisini içeren test verisi kullanılarak, üretilen veriler analiz edilmiştir.

Her bir yazılım, her bir çalıştırma 100 döngü içerecek şekilde 30'ar kez birbirinden bağımsız olarak çalıştırılmıştır. Her iki yöntemin de her bir çalıştırma sonucunda minimum proje süresini elde ettiği görülmüştür.

Yöntemlerin minimum proje süresini elde etme davranışları, kullanılan koloni/yiyecek kaynağı sayılarının giriş verisindeki insan kaynağı sayısı ile oranı ele alınarak analiz edilmiştir. Her bir yöntem 100 iterasyon içerecek şekilde birer kez çalıştırılmıştır. k, projede görevlendirilen insan kaynağı sayısı olmak üzere, KKA yönteminde koloni sayısının k/3, k/2, k ve kx2 olması durumunda minimum süreye belirli bir döngü sonrasında yakınsanabildiği Şekil 3'te görülmektedir. YAKA yönteminde ise yiyecek kaynağı sayısının k/3, k/2 ve k olması durumunda belirli sayıda döngü sonrasında minimum proje süresine yakınsandığı, kx2 olması durumunda ise ilk döngüden itibaren minimum sürenin elde edilebildiği Şekil 4'te görülebilmektedir.



Şekil 3. KKA Yakınsama Oranı (KKA convergence rate)



Şekil 4. YAKA Yakınsama Oranı (YAKA convergence rate)

Her bir uygulamanın eşit sayıda parametrelerle (koloni/yiyecek kaynağı=18, döngü sayısı=100) 500'er kez çalıştırılması sonucu KKA yönteminde ortalama 260 ms., YAKA yönteminde ise ortalama 630 ms. yanıt süresi elde edilmiştir.

3. DENEYSEL SONUÇLAR *(EXPERIMENTAL RESULTS)*

Elde edilen sonuçlara göre, ACO yönteminin ABC yöntemine göre daha hızlı yanıt süresine sahip olduğu ortaya çıkmıştır. Bu hız farkı, KKA yöntemindeki koloni üretme ve yönetme sürecinin, YAKA yönteminde yiyecek kaynaklarının üretme ve yönetme sürecinden daha kolay olmasından kaynaklanmaktadır. KKA yönteminde veriler tek boyutlu olarak işlenmektedir. YAKA yönteminde ise yiyecek kaynakları değişken boyuttadır. Yeniden üretilmesi ve değerlendirilmesi zaman alıcıdır. Bu nedenle de YAKA yönteminin işlem süresi KKA yöntemine göre daha yavaş olmaktadır.

Kullanılan koloni/yiyecek kaynağı sayılarının giriş verisindeki insan kaynağı sayısı ile oranı ele alınarak yöntemlerin yakınsama hızları incelendiğinde aşağıdaki sonuçlar ortaya çıkmıştır:

- Her iki yöntemin de minimum proje tamamlanma süresini elde edecek ekibi belirlemede başarılı olduğu görülmüştür.
- Koloni/yiyecek kaynağı sayısının k/3, k/2 ve k olması durumunda KKA yönteminin sonuca daha erken ulaştığı görülmüştür. Koloni/yiyecek kaynağı sayısının kx2 durumunda ise, YAKA yönteminin daha başarılı olduğu belirlenmiştir.
- KKA yönteminde koloni sayısındaki artışın sonuca yakınsamayı pek fazla etkilemediği görülmüştür.
- YAKA yönteminde koloni boyutunun artması ile sonuca yakınsama hızında önemli derecede artış gözlenmiştir.
- Koloni/yiyecek kaynağı artışı göz önünde bulundurulduğunda YAKA yönteminin sonuç üretme hızının KKA yöntemine göre oldukça başarılı olduğu belirlenmiştir.

4. SONUÇ *(RESULT)*

Yazılım proje takvimi oluşturma NP-hard problemlerden birisidir. Çok sayıda parametre içermesi ve görevlerin birbirine bağımlılıkları problemin durum uzayını çok büyük yapar. Bu nedenle yazılım proje takviminin efektif bir şekilde insan çabası ile üretilmesi oldukça zordur. Bu çalışma ile yazılım proje yöneticilerinin karşılaştığı zorluklardan biri olan yazılım proje takvimi oluşturma sürecine katkı sağlanması amaçlanmıştır.

Yapılan çalışmada yazılım proje takvimi minimum proje süresini içerecek şekilde elde

edilmeye çalışılmıştır. Bu yapılırken de insan kaynakları, aktivitelerin yapılabilirliği ve hangi sürede yapılabildiği bilgisi kullanılmıştır.

Yöntem olarak yapay zeka optimizasyon tekniklerinden KKA ve YAKA kullanılmış ve bu iki algoritmanın sonuca yakınsama hızları incelenmiştir. Her iki yöntemin iterasyonlar sonucu minimum süreyi elde ettiği, koloni/yiyecek kaynağının projede görevlendirilen insan kaynağı sayısına eşit ve daha küçük olduğu durumlarda KKA yönteminin, büyük olduğu durumlarda ise YAKA yönteminin sonuca daha hızlı yakınsadığı belirlenmiştir.

Gelecek çalışmalarda aynı probleme proje yönetiminde kullanılan farklı parametreler dahil edilerek minimum proje süresinin elde edilmesine ilişkin çalışmalara devam edilecektir.

KAYNAKLAR *(REFERENCES)*

[1] S. Hastiwe, S. Wojewoda, Standish Group 2015 "Chaos Report Q&A with Jennifer Lynch.InfoQ.", *www.infoq.com*, Erişilebilir: https://www.infoq.com/articles/standish-chaos-2015, [Erişim Tarihi: 07.03.2018].

[2] Z. Gül, "Yazılım Geliştirme Sürecinin İyileştirilmesi ve Türkiye Uygulamaları", Yüksek Lisans Tezi, İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul, 2006.

[3] B. Crawford, R. Soto, G. Astorga, C. Castro, F. Paredes, S. Misra and J. Rubio, "Solving the Software Project Scheduling Problem Using Intelligent Water Drops", *Tehnički vjesnik*, vol. 25(2), 2018.

[4] A. Barreto, M. de O. Barros and C. M. L Werner, "Staffing a software project: A constraint satisfaction and optimization-based approach" *Computers & Operations Research*, vol. 35(10), pp. 3073-3089, 2008.

[5] M. Peker, B. Sen and S. Bayir, "Using Artificial Intelligence Techniques for Large Scale Set Partitioning Problems" *Procedia Technology*, vol. 1, pp. 44–49, 2012

[6] J. Xiao, X. Ao and Y. Tang, "Solving software project scheduling problems with ant colony optimization", *Computers & Operations Research*, vol. 40, pp. 33-46, 2013

[7] W. Chen and J. Zhang, "Ant colony optimization for software project scheduling and staffing with an event-based scheduler", *IEEE Transactions on Software Engineering*, vol. 39(1), pp. 1-17, 2013

[8] Y. Singh, A. Kaur and B. Suri, "Test case prioritization using ant colony optimization", *ACM SIGNSOFT Software Engineering Notes*, vol. 35(4), pp. 1-7, 2010.

[9] B. Suri and S. Singal, "Implementing ant colony optimization for test case selection and prioritization", *International Journal on Computer Science and Engineering (IJCSE)*, vol. 3(5), pp. 1924-1932, 2011

[10] B. Suri and P. Jajoria, "Using ant colony optimization in software development project scheduling", *2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Ağustos 22-25, 2013, Mysore, İndia,* 2013, pp. 2101-2106.

[11] B. Crawford, R. Soto, F. Johnson, E. Monfroye, E. And F. Paredes, "A Max–Min Ant System algorithm to solve the Software Project Scheduling Problem", *Expert Systems with Applications*, vol. 41, pp. 6634-6645, 2014.

[12] R. Akbaria, V. Zeighamib and K. Ziaratia, "Artificial Bee colony for resource constrained project scheduling problem", *International Journal of Industrial Engineering Computations*, vol. 2(2011), pp. 45-60, 2011.

[13] V. Nayak, H. Suthar and J. Gadit, "Implementation of Artificial Bee Colony Algorithm", *IAES International Journal of Artificial Intelligence*, vol. 1(3), pp. 112-120, 2012.

[14] J. C. Bansal, H. Sharma and S. S. Jadon, "Artificial bee colony algorithm: a survey", *Int. J. Advanced Intelligence Paradigms*, vol. 5, pp. 123-159, 2013.

[15] N. F. B. M. Pauzi, "Flowshop Scheduling Using Artificial Bee Colony (Abc) Algorithm With Varying Onlooker Bees Approaches", Yüksek Lisans Tezi, Faculty of Mechanical and Manufacturing

Engineering Universiti Tun Hussein Onn Malaysia, Johore, Malezya, 2015.

[16] N. F. B. M. Pauzi and S. A. Bareduan, "Scheduling Analysis For Flowship Using Artifical Bee Colony (Abc) Algorithm With Varying Onlooker Approaches", *ARPN Journal of Engineering and Applied Sciences*, vol. 11(10), pp. 6472- 6477, 2016.

[17] M. F. Amer, Optimization Algorithms in Project Scheduling, "Optimization Algorithms - Methods and Applications", Associate Prof. Ozgur Baskan (Ed.), *InTech*, DOI: 10.5772/63108, Available from: https://www.intechopen.com/books/optimization-algorithms-methods-and-applications/optimization-algorithms-in-project-scheduling, 2016

[18] M. Dorigo and G. Di Caro, "Ant colony optimization: A new meta-heuristic", *In Proceedings of the 1999 congress on evolutionary computation, July 6-9, 1999, Washington, DC, USA*, IEEE Press, pp. 1470–1477, 1999.

[19] M. Dorigo, V. Maniezzo and A. Colorni, "Ant system: Optimization by a colony of cooperating agents", *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 26(1), pp. 29–41, 1996.

[20] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem", *IEEE Transactions on Evolutionary Computation*, vol 1 (1), pp. 53-66, 1997.

[21] T. Keskintürk and H. Söyler, "Global karınca kolonisi optimizasyon algoritması", *Gazi Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi*, vol. 12(4), pp. 689-698, 2006.

[22] O. Mohammadrezapour and M. JavadZeynali, "Comparison of ant colony, elite ant system and maximum – minimum ant system algorithms for optimizing coefficients of sediment rating curve (case study: Sistan river)", *Journal of Applied Hydrology*, vol. 1(2), pp. 55-66, 2014.

[23] T. Stützle and H. H. Hoos, "Max min ant system", *Journal of Future Generation Computer Systems*, vol. 8(16), pp. 889–914, 2000

[24] N. Gül and N. Arıcı, "Constitution of Software Project Schedule with Ant Colony Algorithm", *Journal of New Results in Engineering and Natural Science*, vol. 8, pp. 38-47, 2018

[25] D. Karaboga, "An idea based on honey bee swarm for numerical optimization", Technical report, Computer Engineering Department, Engineering Faculty, Erciyes University, 2005.

[26] D. Karaboğa, *Yapay Zeka Optimizasyon Algoritmaları*, Dördüncü Baskı, Türkiye, Nobel Akademik Yayıncılık, 2017, pp. 201-222.

[27] D. Karaboga, B. Gorkemli, C. Ozturk and N. Karaboga, "A comprehensive survey: Artificial bee colony (ABC) algorithm and applications", *Artif. Intell. Rev.*, pp. 1-37, 2012.

Nurhan GÜL
Nurhan GÜL 2003 yılında Karadeniz Teknik Üniversitesi, Bilgisayar Mühendisliği Bölümü'nden mezun oldu. Karadeniz Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı'nda 2006 yılında yüksek lisansını tamamladı. 2013 yılında Gazi Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği (Teknoloji Fakültesi) Anabilim Dalında halen devam etmekte olan doktora eğitimine başladı. Kalkınma Bakanlığı'nda Bilgisayar Mühendisi olarak görev yapmaktadır. Çalışma konuları arasında kaos teorisi ve yapay zeka optimizasyon teknikleri yer almaktadır.

Nursal ARICI
Doç. Dr. / Gazi Üniversitesi
Lisans eğitimini 1985 yılında Gazi Üniversitesi Matematik Bölümü'nde tamamlamıştır. Yüksek Lisansını Gazi Üniversitesi Fen Bilimleri Enstitüsünde "Matematik Öğretiminde Bilgisayar Kullanım İmkânları ve MATEHOBİ Yazılımının Gerçekleştirilmesi" konulu tezi ile, doktorasını "Tarımsal İstatistik Analizlerinde Uzman Sistem Tasarımı ve Gerçekleştirilmesi" konulu tezi ile Ankara Üniversitesi Fen Bilimleri Enstitüsü'nde tamamlamıştır. Halen Gazi Üniversitesi Teknoloji

Fakültesi, Bilgisayar Mühendisliği bölümünde öğretim üyesi olarak görev yapmaktadır. Çalışma konuları arasında Bilgi Sistemleri Analiz, Tasarım ve Gerçeklemesi, Veri analizi, Veri Madenciliği ve Yapay Zekâ Teknikleri yer almaktadır.