Received: November 16, 2017
Accepted: January 09, 2018

Karınca Kolonisi Algoritması ile Yazılım Proje Takvimi Oluşturma

Nurhan GÜL[1*], Nursal ARICI [2]

[1,2] Gazi University, Faculty of Technology, Computer Engineering Department, 06500, Ankara, Turkey

Özet

Yazılım proje takvimi, yetkinlikleri göz önünde bulundurularak eldeki insan kaynaklarının görevlere tahsis edilmesi, görevlerin projenin ihtiyaçlarına göre uygun sırada işlenmesi, görev sürelerinin tahmin edilmesi, gerekli harcamaların görevlerle ilişkilendirilmesi süreçlerini kapsar. İnsanların görevlere atanması, bu problemin çözümünü karmaşık yapar. Problemin durum uzayı çok büyüktür. Bu gibi karmaşık problemlerde durum uzayını daraltmak için çeşitli yapay zeka optimizasyon algoritmaları kullanılabilmektedir. Bu çalışmada yazılım proje takviminin efektif bir şekilde oluşturulabilmesi için karınca kolonisi algoritması yöntemlerinden, maksimum minimum karınca sistemi algoritması uygulanmıştır. Uygulanan algoritma ile kaynakların görevleri tamamlama zamanı ve görevleri yapabilirliği dikkate alınarak, proje takvimindeki görevlerin en kısa sürede tamamlanması için uygun çözüm bulunmuştur. Algoritmanın stokastik özelliği nedeniyle her zaman en ideal çözüme ulaşmak mümkün olmamaktadır. Bununla birlikte, yapılan deneysel çalışma ile yazılım proje takvimindeki tüm aktiviteleri en kısa sürede gerçekleştirecek uygun insan kaynaklarını belirlemek için rastgele başlangıç noktasından ideal çözüme yakınsayacak şekilde karınca kolonisi algoritmasının kullanılabileceği gözlenmiştir. Deneysel sonuçlar, proje takvimi oluşturulması sürecinde karınca kolonisi algoritmasının beklenen çözümü ürettiğini göstermektedir.

Anahtar Kelimeler : Yazılım Proje Yönetimi, Yazılım Proje Takvimi, Karınca Kolonisi Algoritması, Yapay Zekâ Optimizasyon Algoritması.

Constitution of Software Project Schedule with Ant Colony Algorithm

Nurhan GÜL[1], Nursal ARICI [2]

Abstract

Software project scheduling include the process of assigning human resources to tasks in consideration of their competencies, processing tasks according to the needs of the project in an order, estimating of the duration of tasks and associating required budget with tasks. Assigning human resources to tasks makes the solution space too complex. The state space of the problem is too large. In such complex problems, artificial intelligence algorithms can be a solution to narrow the state space. In this study, max-min ant colony system algorithm is applied in order to create a software project schedule in an effective way. While taking into account time for human resources to complete tasks and the ability of human resources to perform tasks, it finds appropriate solution includes minimum time for the completion of tasks in the project schedule. Because of the stochastic nature of the algorithm, it is not always possible to achieve the best solution. However, starting from the random status, the solution converges to the effective status with KKA-YPTO algorithm. Experimental results show that the algorithm produces the expected results.

Keywords : Software Project Management, Software Project Schedule, Ant Colony Algorithm, AI Optimization Algorithm.

[*] *Corresponding Author*, e- mail: nurhangul13@gmail.com

1. Giriş

Proje genel olarak; belirli bir amacı olan, belirli bir zamanda belirli bir bütçeyle bu amacı gerçekleştirmeyi hedefleyen, bunu yaparken de kalite, risk, insan, iletişim gibi unsurları da kapsayan bir süreçtir. Her projenin yeni bir çıktısı vardır. Yeni bir fikir sonrası bir ürünün ortaya çıkması bir proje olarak tanımlanabileceği gibi, var olan bir ürüne yeni özellikler eklenmesi de bir projedir.

Bir projenin başarılı sayılması için çeşitli değerlendirmeler yapılmıştır. Proje Yönetimi Enstitüsü(PMI)'ne göre, bir projenin başarılı olabilmesi için; projenin zaman, bütçe, kapsam, risk, kalite ve kaynak parametrelerinin birlikte değerlendirilmesi gerekmektedir [1]. Standish Group 2015 raporuna göre başarılı bir proje, zamanında tamamlanmış, belirlenmiş bütçeyi aşmamış ve memnun edici sonuçların elde edildiği proje olarak tanımlanmıştır [2]. Lewis'e göre ise, projenin başarısını ölçmek için dört kriter vardır: performans, maliyet, zaman ve kapsam [3].

Yazılım projeleri diğer projelere oranla daha karmaşıktır. Bu karmaşıklığı oluşturan faktörler şu şekilde sıralanabilir [4]:

- Yazılım projesi soyuttur: Yazılım projelerinin ifade edilmesi, anlaşılması ve denetlenmesi kolay değildir.
- Standart bir yazılım geliştirme süreci yoktur: Bir inşaat projesi, aynı özelliklerde farklı lokasyonlar için aynı süreçler kullanılarak gerçekleştirilebilir. Sonuçta aynı çıktı üretilir. Ancak yazılım geliştirme projeleri, tek bir parametresi farklı olsa bile farklı sonuçlar üretebileceğinden, her seferinde şartlara uygun olarak yönetilmelidir.
- Tek seferlik projelerdir: Proje yöneticisinin yeterli deneyimi olsa bile, her yeni projede yeni deneyimler elde edilir.

Yazılım projeleri, başlangıçta planlanan hedefe, ortaya konan bütçe ve proje planına göre ulaşmayı hedefler. Bu projeler, yeni bir fikrin hayata geçirilmesi için olabileceği gibi, var olan bir projeye ek özellikler kazandırılması için de başlatılabilir. Yazılım alanı dışındaki projelerde bu kriterlerle süreç başlatıldığında hedefe başarılı bir şekilde ulaşılma oranı daha yüksek olmaktadır. Yazılım projeleri ise bu noktada deyim yerindeyse sınıfta kalmaktadır. Ya istenen bütçe ve planlanan teslimat tarihi aşılmakta ya da müşterinin istediği özellikte bir ürün ortaya konulamamaktadır. Bu durumun, yönetim desteğinin eksikliği, müşterinin istediği ürünü net olarak tanımlayamaması, sürecin iyi yönetilememesi gibi birçok nedeni bulunmaktadır.

Field (1997) CIO dergisinde yer alan makalesinde, bir yazılım projesini başarısızlığa götüren on nedeni şu şekilde sıralamıştır [5]:

- Proje yöneticilerinin kullanıcı ihtiyaçlarını anlamaması
- Proje kapsamının net olmaması
- Proje değişikliklerinin iyi yönetilememesi
- Seçilen teknolojinin değişmesi
- İş gereksinimlerinin değişmesi
- Bitiş tarihinin gerçekçi olmaması
- Kullanıcıların değişikliğe direnmesi
- Sponsorluğun sona ermesi
- Proje ekibinin uygun beceriden yoksun kişilerden oluşması
- Yöneticilerin en iyi çözümü ve deneyimleri dikkate almaması.

Bu tespit, her ne kadar yirmi yıl kadar önce yapılmış olsa da bugün halen geçerliliğini sürdürmektedir. Proje yönetimi başarısında sorunun kaynağı bellidir, ancak çözümün ne şekilde gerçekleştirileceğine ilişkin somut ve genel bir metodoloji henüz geliştirilememiştir. Bunun nedeni, yazılım proje yönetiminin genel çerçeveye oturtulamayacak kadar çok faktörlü ve karmaşık süreçler içermesidir.

Reel (1999) çalışmasında, yazılım projelerinin başarılı olma sebebini beş temel başlık altında toplamıştır:

- Doğru adımla başlamak
- İvmeyi sürdürmek
- Süreci takip etmek
- Zekice kararlar almak
- Geçmişe yönelik yaşanmışlıklardan ders almayı kurumsallaştırmak.

Reel (1999)'e göre, müşteri ile proje ekibi arasında taahhütler imzalanmalı, proje ekibi müşteriye uygun bir yaklaşım sergilemeli, proje yöneticisi ekibini iyi oluşturmalı ve bu ekibin moral ve motivasyonunu yüksek tutmalıdır. Proje için uygun teknoloji belirlenmeli, önceki deneyimlerden ders çıkarılmalı ve kötü sonuçlanan bir projeyi, o noktaya götüren faktörler üzerinde düşünmek için vakit harcanmalıdır [6].

Liu, Kane ve Bambroo (2006), yazılım projelerinde başarısızlığın fark edildiği anın oldukça geç olduğunu, daha verimli projeler için erken uyarı sistemlerinin geliştirilmesinin gerekli olduğunu ifade etmektedirler. Liu vd.(2006)'ne göre projenin erken dönemlerinden itibaren risklerin belirlenmesi ve önceliklendirilmesi gerekmektedir.

Liu vd. (2006), yazılım geliştirme sürecini ürün, süreç ve organizasyon perspektifinden analiz etmişlerdir. Süreç boyutunda; proje, aşama, görev, ürün boyutunda; sistem, alt sistem, modül, organizasyon boyutunda da; şirket, grup ve birey olarak alt kırılımlar oluşturmuş, riskleri bu alt kırılımlar bazında bulanık mantık çerçevesinde değerlendirmişlerdir. Böylelikle, risklerin sebeplerine ulaşmaya çalışmışlardır [7].

McBride (2008), yazılım proje yöneticilerinin proje yönetimi sırasında kullandıkları yöntemi belirleyebilmek için deneysel bir çalışma gerçekleştirmiştir. Proje geliştirme aşamasında ortaya çıkan beklenmeyen olayların ve dış etkilerin önemli bir etken olduğundan yola çıkarak, süreç boyunca izleme, kontrol ve koordinasyon işlerinin nasıl gerçekleştirildiğini araştırmıştır. Çalışma sonucuna göre, birçok proje yöneticisinin proje izleme yöntemi olarak proje planını ve kilometre taşlarını takip etme yöntemini kullandığı, erken uyarı sistemi olarak yazılımlardan, log bilgilerinden ve iletişim sonucu elde edilen bilgilerden yararlandığını tespit etmiştir. Çalışma sonucunda elde edilen bir diğer bilgi de, proje yöneticilerinin %80'inin koordinasyon için proje planını ve şartnameleri kullandığıdır [8].

Wang, Jia ve Qu (2010), mevcut risk analizi metodlarının genel proje yaklaşımı üzerinden çalışmakta olduğunu, yazılım projelerinin karakteristik özelliklerine ve yazılım mühendisliği süreçlerine odaklanmadıklarını, bu sebeple de bu çalışmaların yazılım projeleri alanında iyi uygulanamadıklarını öne sürmektedirler. Wang vd. (2010) çalışmalarında, proje yaşam döngüsü ile risk yaşam döngüsünün iç içe modellendiği dünya-ay modelini geliştirmişlerdir. Bu modeldeki risk yönetim modeli risk belirleme, risk analizi, risk planlama, risk izleme ve risk kontrol süreçlerini içermektedir. Bu süreçler, hangi yazılım proje geliştirme sürecinde çalışılıyorsa, o sürece ait risk değerlendirmesini gerçekleştirmektedir. Risk yaşam döngüsü her bir proje yaşam döngüsü için bir alt süreç olarak

işletilmektedir. Modelde, her bir proje yönetim aşaması için yapılan risk değerlendirmesi sonucuna göre bir sonraki aşamaya geçilip geçilmeyeceğine karar verilmektedir [9].

Othman, Zain ve Hamdan (2010), geleneksel proje yönetimi yaklaşımının, beklenen süreçlerin düzgün ilerleyeceği varsayımı üzerine kurulduğunu belirtmektedir. Proje yönetimi ortamlarında süreç boyunca beklenmeyen durumların(kaotik durum) ortaya çıkabileceği ve bu durumların da geleneksel proje yönetimi yaklaşımı ile başarılı bir şekilde ele alınamayacağını savunmaktadır. Mevcut standartlar projenin başarısı için net yöntemler ve geniş dokümantasyon sunsa da, kaotik giden durumlar için nasıl bir aksiyon alınması gerektiğine dair fikirler sunmamaktadır. Diğer bir deyişle standartlar, değişikliğe ve kaosa yatkın durumlar için esnek ve uyumlaştırılabilir bir yaklaşım içermemektedir.

Othman vd. (2010)'ne göre, başarılı bir yazılım projesi için, başlangıçta güçlü bir planlama yapılmalıdır. Maliyet ve risk değerlendirmesi gerçekleştirilerek, olası kaotik durumlar ile yüzleşme sağlanmalıdır. Durağan ve kaotik durum karşısında dengeyi sağlayacak şekilde uygun geliştirme ortamına sahip sistemler hazırlanmalıdır [10].

Gharehchopogh ve Dizaji (2014), yazılım projelerinin maliyet tahminlerini yapak amacıyla, yazılım proje türüne göre ilk sınıflandırmayı yaptıktan sonra, eğitim verileri üzerinde yapay arı kolonisi, arı koloni optimizasyon ve kaos optimizasyon algoritmalarını kullanarak, COCOMO II modelindeki sabit değişkenlerin tahminini gerçekleştirmeye çalışmışlardır. Bu değerleri belirledikten sonra da, test verilerini kullanarak yazılım proje maliyet tahminlemesi gerçekleştirmişlerdir [11].

Xia, Ao, ve Tang (2013) çalışmalarında, yazılım proje takviminin oluşturulma zorluğuna, karınca kolonisi algoritması ile sezgisel yaklaşımı birleştirerek bir çözüm sunmuşlardır. Çalışmalarında, kaynak atamalarını bir grafta modelleyip problemi karınca kolonisi optimizasyon algoritması ile çözülebilecek şekilde graf tabanlı arama algoritması yaklaşımına dönüştürmüşlerdir. Sonrasında geliştirdikleri sezgisel algoritmalarla, görevlerin kaynaklara atanması problemine bir çözüm üretmişlerdir [12].

Chen ve Zhang (2013) çalışmalarında, olay tabanlı planlama ile karınca kolonisi optimizasyon algoritmasını birleştirerek yazılım proje takvimi oluşturulmasında kullanılmak üzere bir yöntem geliştirmişlerdir. Projenin başlangıcı, görevlerin sonunda kaynakların boşa çıktığı zaman, kaynakların projeye dahil olduğu ya da projeden ayrıldığı zaman, olay olarak değerlendirilip, bu olaylar oluştuğunda kaynakların tahsis edilmesini sağlayacak algoritmalarıyla kaynak çakışması, görevlerin önceliklendirilmesi gibi problemlere çözüm üretmişlerdir [13].

Singh, Kaur ve Suri (2010) ve Suri ve Singal (2011) çalışmalarında yazılım projelerinde regresyon testi için seçilecek test senaryolarının belirlenmesinde karınca kolonisi optimizasyon algoritmasını kullanmışlardır[14,15]. Suri ve Jajoria (2013), bu çalışmadan yola çıkarak benzer yaklaşımı proje yöneticilerinin proje maliyetini ve toplam proje süresini en aza indirgeyebilmeleri amacıyla, verimli kaynak atamalarını sağlayabilecek şekilde yazılım proje takvimi oluşturulması alanında da uygulamışlardır [16].

Yazılımın soyut bir kavram olması, müşterilerin istedikleri ürünü teslim edildiği zaman daha net tarif edebilmesi ve talep değişikliklerinde geri dönüşlerin zor olması yazılım geliştirme sürecini zorlu bir süreç yapan etkenlerdir. Bu zorluk, yazılım projelerinin başarıyla tamamlanmasını olumsuz etkilemektedir. Başarısız bir proje, yazılım şirketi ve müşteri için maddi manevi değer kayıpları oluştururken, projede yer alan proje yöneticileri ve yazılım ekibi için de başarısız iş deneyimi sonucunu ortaya koymaktadır.

Yazılım proje yönetimi, belirlenen bütçe ve zaman çerçevesinde istenen ürünün en iyi şekilde teslim edilmesini amaçlayan bir disiplindir. Proje yöneticisi de bu disiplin çerçevesinde, projenin başarı ile tamamlanabilmesi ve süreç boyunca projenin belirlenen hedefe ulaşabilmesi amacıyla planlama ve organize etme, izleme ve kontrol gibi faaliyetler yürütür. Bunu yaparken de kurum kaynaklarını etkin bir şekilde kullanmaya gayret eder [17].

2. Materyal ve Metot

2.1. Yazılım Projelerinde Proje Takvimi

Yazılım projelerinin insan kaynağı, zaman ve bütçe planlamalarının izlemesi proje takvimi aracılığı ile yapılır.

Proje takvimi, yetkinlikleri göz önünde bulundurularak eldeki insan kaynaklarının görevlere tahsis edilmesi, görevlerin projenin ihtiyaçlarına göre uygun sırada işlenmesi, görevlerin süresinin tahmin edilmesi, gerekli harcamaların görevlerle ilişkilendirilmesi süreçlerini kapsar.

İyi planlanmış bir proje takvimi, yazılım proje yöneticisinin proje için ortaya konan bütçe ve hedeflenen bitiş tarihini dikkate alarak projede sağlıklı izlemeler yapılabilmesine olanak sağlar. Bu izlemeler neticesinde proje yöneticisi projenin durumunu değerlendirip, neler yapması gerektiğine dair planlamalar yapabilir.

Karınca Kolonisi Optimizasyon Algoritması(KKA)

Yazılım proje takvimi oluşturulması süreci, belirleyici olmayan zor polinom yapıdadır(NP-hard). Kaynaklara görevlerin atanmasında problemin durum uzayı çok büyüktür ve çözümü zaman alıcıdır. Karınca kolonisi optimizasyon algoritması, durum uzayını daraltmak için kullanılan algoritmalardan birisidir [12,13].

Bu algoritma, karıncaların doğadaki davranışlarından esinlenilerek geliştirilmiştir. Algoritmada, karıncaların yiyeceğe ulaşma sürecinde salgıladıkları maddenin, diğer karıncalar tarafından yorumlanması ile hedefe en kısa yoldan ulaşmaya çalışmaları modellenmiştir.

Karıncalar yiyeceğe ulaşırken, feromon denilen bir sıvı salgılarlar. Arkadan gelen karıncalar, gidecekleri yol tercihini yaparken feromon miktarının fazla olduğu yolu tercih ederler. Feromon miktarı belirli bir süre sonra buharlaşmaya başlar. Bu durum önceden verilmiş kararların önem derecesini azaltmaya yarar. Yiyecek bulma sürecinde zaman geçtikçe, yiyeceğe giden en kısa yolda karıncaların yoğunlaştığı görülür. Bu davranıştan esinlenilerek karınca kolonisi optimizasyon algoritması geliştirilmiştir [18].

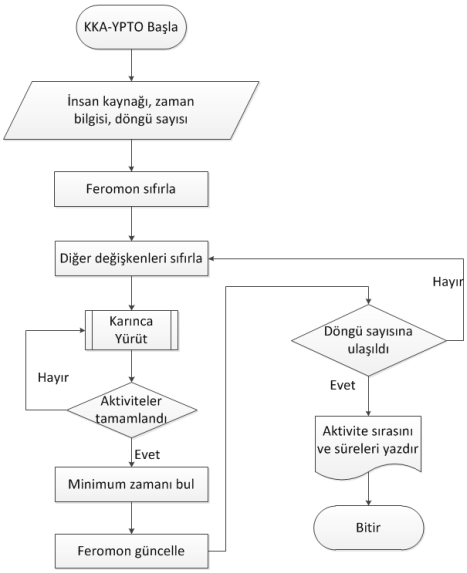
Bu çalışmada karınca kolonisi optimizasyon algoritmalarından maksimum-minimum karınca sistemi kullanılmıştır. Feromon güncellemeleri turun en iyi olan karıncası ve o ana kadar tüm turlardaki en iyi karınca tarafından belirli aralıklarda gerçekleştirilmektedir. Böylelikle arama işleminin belirli noktalarda durağanlaşması önlenmiş olmaktadır [19,20].

Karınca Kolonisi Algoritması ile Yazılım Proje Takvimi Oluşturma

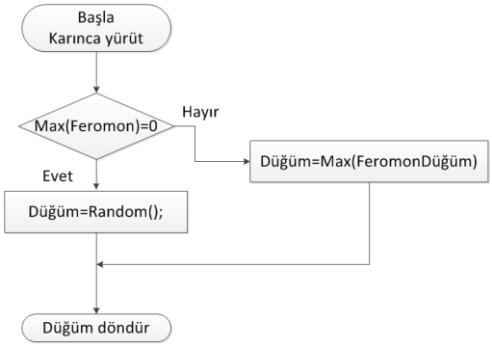
İyi bir yazılım proje takvimi oluşturulabilmesi için, projenin görev kırılımlarının iyi belirlenmesi, kaynakların görevlere yetenekleri ve deneyimleri doğrultusunda atanmaları, görevler arasındaki bağımlılıkların belirlenmesi, görevlerin yaklaşık tamamlanma süresinin tahmin edilebilmesi gerekmektedir.

İnsan kaynağı ihtiyaçlarının belirlenmesi, maliyet tahmini ve risk değerlendirmesinin yapılması ve proje takvimi oluşturulması gibi süreçlerde durum uzayını küçültmek amacıyla yapay zeka algoritmaları kullanılmaktadır [21].

Karınca kolonisi optimizasyon algoritması bu alanda kullanılabilen yapay zeka algoritmalarından birisidir. Şekil 1'de, yazılım proje takvimi oluşturulması amacıyla kullanılan karınca kolonisi optimizasyon algoritmasının(KKA-YPTO) akış şeması yer almaktadır.



(a) Ana fonksiyon



(b) Karınca yürütme fonksiyonu

Şekil 1. KKA-YPTO akış şeması.

Bu algoritma ile projedeki tüm görevlerin en kısa sürede yapılabilmesi için uygun yeterlilikte insan kaynağı atamasının gerçekleştirilebilmesi hedeflenmektedir. Algoritmadaki genel kurallar aşağıda belirtilmiştir:

- Her bir insan kaynağı için bir karınca oluşturulur.
- Her döngü için her bir karınca temsil ettiği insan kaynağını başlangıç düğümü olarak alır.
- Karıncanın ilerlediği düğümler insan kaynağına karşılık gelir.
- Karıncanın gittiği her düğüm, projeye yeni bir insan kaynağının dahil edilmesi anlamını taşır.
- Karıncalar, geçtikleri düğümlerden tekrar geçmezler.
- Tüm kenarlardaki feromon değeri başlangıçta sıfır alınır.
- Algoritmanın ilk döngüsünde ortamda feromon izi olmadığından karıncaların düğümler üzerindeki ilerlemeleri rastgele olur.

- Karıncanın ilerleyebileceği birden fazla düğüm varsa, o düğüme giden yollardaki feromon izi en yüksek olan tercih edilir.
- Her gidilen düğümde, o ana kadar seçilen insan kaynaklarının yapabildiği görevlerin, tüm yapılması gereken görevleri kapsayıp kapsamadığı kontrol edilir. Kapsıyorsa döngü tamamlanır, kapsamıyorsa yeni bir düğüm seçme süreci başlatılır.
- Döngü tamamlandığında, o döngü için en kısa sürede hedefe ulaşan karınca belirlenir. Bu karıncanın geçtiği yollar için minimum ve maksimum sınırlar içerisinde kalacak şekilde feromon ekleme işlemi yapılır. Başlangıç noktasından ilgili döngüye kadar olan en iyi yol için de benzer şekilde feromon eklemesi gerçekleştirilir.
- O ana kadar üzerinden geçilen tüm yollar için ise %10 oranında feromon buharlaştırma işlemi gerçekleştirilir [14-16].
- Feromon izi dışındaki değişkenler sıfırlanır ve karıncalar yeniden başlangıç noktasına alınarak aynı işlemler maksimum döngü sayısına ulaşılıncaya kadar tekrar edilir.

Algoritma belirli bir süre çalıştığında, feromon izine göre hedefe ulaşmaya çalışmaları nedeniyle, karıncaların uygun çözümü sağlayan düğümler üzerinde yoğunlaştığı görülebilmektedir.

3. Bulgular ve Tartışma

Bu çalışmada, yazılım proje takviminde görevlere kaynak atanması probleminde karınca kolonisi optimizasyon algoritması kullanılarak optimum çözümün üretilebildiğine dair sonuçlar gözlenmiştir.

Tablo 1'de yer alan örnek giriş verileri Şekil 1'deki algoritma ile işlenerek, projedeki görevlerin en kısa sürede ve uygun insan kaynakları ile tamamlanması için etkin çözümün bulunması araştırılmıştır.

Tablo1. KKA-YPTO algoritması giriş verisi

E/A	A0	T0	A1	T1	A2	T2	A3	T3	A4	T4	A5	T5	A6	T6	A7	T7	A8	T8	A9	T9	A10	T10	A11	T11
E0	0	0	1	4	0	0	1	12	0	0	0	0	1	3	0	0	1	5	0	0	0	0	0	0
E1	1	10	0	0	1	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	5	0	1
E2	0	0	0	0	0	0	0	0	1	5	0	0	1	5	1	7	0	0	0	0	0	0	1	0
E3	0	0	1	21	0	0	1	3	0	0	0	0	0	0	0	0	1	7	0	0	0	0	1	0
E4	0	0	0	0	1	2	0	0	0	0	1	6	0	0	0	0	0	0	1	5	0	0	0	0
E5	1	4	0	0	0	0	0	0	0	0	0	0	1	6	0	0	0	0	0	0	0	0	1	1
E6	0	0	0	0	1	3	0	0	0	0	1	8	0	0	1	5	0	0	0	0	1	4	0	0
E7	0	0	1	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	3	0	0	0	0
E8	1	12	0	0	0	0	0	0	1	2	0	0	1	8	0	0	0	0	1	6	0	0	0	1
E9	0	0	0	0	0	0	0	0	1	4	0	0	1	5	1	3	0	0	0	0	1	9	1	0
E10	0	0	1	14	0	0	1	8	0	0	0	0	0	0	0	0	1	3	0	0	0	0	1	0
E11	0	0	0	0	1	7	0	0	0	0	1	8	0	0	0	0	0	0	1	7	0	0	0	0
E12	1	9	0	0	0	0	0	0	0	0	0	0	1	9	0	0	0	0	0	0	1	6	0	1
E13	1	2	0	0	1	4	0	0	0	0	1	5	0	0	1	9	0	0	0	0	0	0	1	1
E14	0	0	1	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	0	0	1	0
E15	0	0	0	0	0	0	0	0	1	13	0	0	1	1	0	0	0	0	1	6	0	0	0	0

Giriş verilerinde bulunan satır değerleri insan kaynaklarını ifade etmektedir($E_0..E_k$). Sütun değerleri iki bölümden oluşmaktadır: ($A_0..A_n$) olarak ifade edilen sütunlar projedeki görevlere, ($T_0..T_m$) ile ifade edilen sütunlar, sütunların solundaki görevin tamamlanması için gereken süreye karşılık

gelmektedir. ($A_0..A_n$) ile belirtilen görevlerden 1 değerine sahip olanlar, kesiştiği satırdaki insan kaynağının o görevi yapabildiği anlamını taşımaktadır.

Tablo 1'de yer alan örnek veri dosyası için algoritma ayrı ayrı 50 kez her biri maksimum 500 döngü içerecek şekilde çalıştırıldığında ortaya çıkan sonuçlar Tablo 2'de gösterilmiştir.

Başlangıçta ortamda feromon izi olmadığından ilk döngüde karıncalar rastgele ilerlemektedirler. Bu rastgele başlangıç durumu, bazen sonucun ideale yaklaşamamasına neden olabilmektedir. Bu durum Tablo 2'de mimimum sürenin 77'den farklı olduğu sonuçlarda gözlenmektedir.

Tablo2. KKA-YPTO algoritması çıkış verisi

Sıra	Döngü	Karınca	Minimum Süre	Sıra	Döngü	Karınca	Minimum Süre
1	500	4	83	**26**	14	10	77
2	3	8	77	**27**	500	0	78
3	500	4	78	**28**	183	10	77
4	49	8	77	**29**	500	0	79
5	3	8	77	**30**	500	1	87
6	22	8	77	**31**	36	10	77
7	9	8	77	**32**	26	10	77
8	500	3	85	**33**	500	0	78
9	99	8	77	**34**	500	0	78
10	101	10	77	**35**	90	10	77
11	2	6	77	**36**	500	0	79
12	127	8	77	**37**	500	0	78
13	500	0	78	**38**	3	10	77
14	39	8	77	**39**	95	8	77
15	102	6	77	**40**	500	0	83
16	500	5	79	**41**	1	6	77
17	3	6	77	**42**	11	8	77
18	59	10	77	**43**	11	10	77
19	8	10	77	**44**	18	8	77
20	301	6	77	**45**	500	0	82
21	5	6	77	**46**	1	10	77
22	27	6	77	**47**	36	6	77
23	177	8	77	**48**	500	6	84
24	90	6	77	**49**	1	8	77
25	4	10	77	**50**	85	8	77

4. Sonuç

Yazılım proje yönetiminde, iyi bir proje yönetimi başarılı bir sonuç için önemli bir faktördür. Ancak tek başına yeterli değildir. Sürecin olması gerektiği gibi idare edilmesinin yanı sıra, ortaya çıkması muhtemel olumsuzluk oluşturacak durumların da belirlenmesi ve etkilerinin azaltılması gerekmektedir. Bunun için de proje yöneticisinin iyi bir izleme yapabilmesi gerekmektedir.

Proje takvimi, yazılım proje yöneticisinin projeye ait bütçe, zaman, insan kaynağı gibi bilgileri üst seviyeden izleyebilmesine olanak sağlar. Bu nedenle proje takvimi, yazılım proje yöneticisi için önemli kontrol araçlarından biridir.
Yazılım proje takvimi oluşturma süreci durum uzayı büyük olan karmaşık bir problemdir. İnsanların görevlere atanması, bu problemin çözümünü karmaşık(NP-hard) yapar. Bu durum yazılım proje yönetimi açısından önemli bir yeri olan proje takviminin oluşturulması sürecinin manuel olarak

yapılmasını yetersiz kılar ve yazılım proje yöneticilerinin yardımcı araç ve teknikler kullanmasını gerektirir.

KKA-YPTO yönteminde, proje takviminin efektif bir şekilde oluşturulabilmesi için karınca kolonisi optimizasyon algoritması kullanılmıştır. Tasarlanan karınca sisteminde başlangıçta ortamda feromon izi bulunmamaktadır. Bu nedenle ilk proje ekibi rastgele belirlenerek, minimum değeri üreten karıncanın gittiği yollar için feromon güncellemesi gerçekleştirilir. Sonraki turlarda projeye dahil edilecek insan kaynağının seçimi, karıncanın mevcut feromon izlerini yorumlaması ile gerçekleştirilir. Global feromon güncellemesinin de katkısıyla, zamanla karıncaların seçimini yaptıkları insan kaynaklarının, minimum süreyi üretebilecek insan kaynakları olduğu gözlenir. Algoritmanın stokastik özelliği nedeniyle her zaman en iyi çözüme ulaşmak mümkün olmamakla beraber, bu durumlarda ideal çözüme yakınsanmaya çalışıldığı gözlenir.

KKA-YPTO yönteminde, kaynakların görevleri tamamlama zamanı ve görevleri yapabilirliği dikkate alınarak, proje takvimindeki görevlerin en kısa sürede tamamlanması için uygun çözümün bulunması sağlanmıştır.

Bir sonraki çalışmada, proje takviminin daha fazla parametre içerecek şekilde benzer yapay zeka algoritmaları ile oluşturulması hedeflenmektedir. Ortaya çıkacak proje takvimi modeli ile yazılım projelerinde proje yöneticilerinin somut veriler doğrultusunda çıkarımlar yapabilmesine olanak sağlanacaktır.

5. Kaynaklar

[1] Project Management Institute (2013). *A Guide To The Project Management Body Of Knowledge* (5. basım), Project Management Institute Inc, Pennsylvania, ABD.
[2] Hastiwe S (2016). Standish Group 2015 Chaos Report - Q&A with Jennifer Lynch.InfoQ. http://www.webcitation.org/query?url=https%3A%2F%2Fwww.infoq.com%2Farticles%2Fstandish-chaos-2015&date=2016-11-07, (Erişim tarihi: 7 Kasım 2016).
[3] Lewis JP (2001). *Project Planning, Scheduling, and Control: A Hands-on Guide to Bringing Projects On Time and On Budget* (6. basım), The McGraw-Hill Companies, New York, ABD.
[4] Gül Z (2006). Yazılım geliştirme sürecinin iyileştirilmesi ve Türkiye uygulamaları. Yüksek lisans tezi, İstanbul Teknik Üniversitesi, İstanbul, Türkiye.
[5] Field T (1997). When bad things happen to good projects. *CIO (15 Eylul 1997)*: 55-62.
[6] Reel J (1999). Critical success factors in software projects. *IEEE Software*, 16(3):18-23.
[7] Liu X, Kane G, Bambroo M (2006). An intelligent early warning system for software quality improvement and project management. *The Journal of Systems and Software*, 79:1552-1564.
[8] McBride T (2008). The mechanisms of project management of software development. *The Journal of Systems and Software*, 81: 2386-2395.
[9] Wang Y, Jia J, Qu Y (2010). The "Earth-Moon" model on software project risk management. *Proceedings of the Ninth International Conference on Machine Learning and Cybernetics*, Cilt 4, s.1999-2003, Qingdao, Çin.
[10] Othman M, Zain AM, Hamdan AZ (2010). A review on project management and issues surrounding dynamic development environment of ICT project:Formation of research area. *International Journal of Digital Content Technology and its Applications*, 4(1): 96-105.
[11] Gharehchopogh FS, Dizaji ZA (2014). A new approach in software cost estimation with hybrid of bee colony and chaos optimizations algorithms. *MAGNT Research Report*, 2(6):1263-1271.
[12] Xiao J, Ao X, Tang Y (2013). Solving software project scheduling problems with ant colony optimization. *Computers & Operations Research*, 40:33-46.

[13] Chen W, Zhang J (2013). Ant colony optimization for software project scheduling and staffing with an event-based scheduler. *IEEE Transactions on Software Engineering*, 39(1):1-17.
[14] Singh Y, Kaur A, Suri B (2010). Test case prioritization using ant colony optimization. *ACM SIGNSOFT Software Engineering Notes*, 35(4):1-7.
[15] Suri B, Singal S (2011). Implementing ant colony optimization for test case selection and prioritization. *International Journal on Computer Science and Engineering (IJCSE)*, 3(5):1924-1932.
[16] Suri B, Jajoria P (2013). Using ant colony optimization in software development project scheduling. *2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, s.2101-2106.
[17] Esatoğlu N (2010). Bilgi teknolojileri proje yönetimi ve başarı koşulları. Yüksek lisans tezi, Ankara Üniversitesi, Ankara, Türkiye.
[18] Keskintürk T, Söyler H (2006). Global karınca kolonisi optimizasyon algoritması. *Gazi Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi*, 12(4):689-698.
[19] Mohammadrezapour, JavadZeynali M (2014). 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*, 1(2):55-66.
[20] Stützle T, Hoos HH (2000). Max min ant system, *Journal of Future Generation Computer Systems*, 8 (16):889–914.
[21] Meziane F, Vadera S (2010). *Artificial intelligence in software engineering current developments and future prospects*, (1. basım), Information Science Reference, New York, ABD.