T.C.
BAHÇEŞEHİR ÜNİVERSİTESİ

SARMAL VE ÇEVİK YAZILIM GELİŞTİRME ÇİZELGESİNİN SEZGİSEL YÖNTEMLERLE OPTİMİZASYONU VE KARŞILAŞTIRMASI

Yüksek Lisans Tezi

MAHMUT BAŞ

İSTANBUL, 2015

T.C.
BAHÇEŞEHİR ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİ TEKNOLOJİLERİ

SARMAL VE ÇEVİK YAZILIM GELİŞTİRME ÇİZELGESİNİN SEZGİSEL YÖNTEMLERLE OPTİMİZASYONU VE KARŞILAŞTIRMASI

Yüksek Lisans Tezi

MAHMUT BAŞ

Tez Danışmanı: PROF.ADEM KARAHOCA

İSTANBUL, 2015

T.C.
BAHÇEŞEHİR ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİ TEKNOLOJİLERİ

Tezin Adı : Sarmal ve Çevik Yazılım Geliştirme Çizelgesinin Sezgisel Yöntemlerle Optimizasyonu ve Karşılaştırması
Öğrencinin Adı Soyadı : Mahmut Baş
Tez Savunma Tarihi : 12.01.2015

Bu tezin Yüksek Lisans tezi olarak gerekli şartları yerine getirmiş olduğu Fen Bilimleri Enstitüsü tarafından onaylanmıştır.

Doç. Dr. Nafiz ARICA
Enstitü Müdürü
İmza

Bu tezin Yüksek Lisans tezi olarak gerekli şartları yerine getirmiş olduğunu onaylarım.

Doç. Dr. Mehmet Alper Tunga
Program Koordinatörü
İmza

Bu Tez tarafımızca okunmuş, nitelik ve içerik açısından bir Yüksek Lisans tezi olarak yeterli görülmüş ve kabul edilmiştir.

Jüri Üyeleri	İmzalar
Tez Danışmanı Prof. Dr. Adem Karahoca	----------
Ek Danışman Yrd. Doç. Dr. Yalçın Çekiç	----------
Üye Doç. Dr. Mehmet Alper Tunga	----------

ÖZET

SARMAL VE ÇEVİK YAZILIM GELİŞTİRME ÇİZELGESİNİN SEZGİSEL YÖNTEMLERLE OPTİMİZASYONU VE KARŞILAŞTIRMASI

Mahmut Baş

Bilgi Teknolojileri

Tez Danışmanı: Prof. Dr. Adem Karahoca

Ocak 2015, 72 sayfa

İlerleyen zaman ve teknoloji ile birlikte, yazılım geliştirme projeleri de büyümüş ve karmaşıklaşmıştır. Yazılım projelerindeki büyüklük ve karmaşıklığın artması insan kaynağı yönetimini zorlaştırmıştır. Bu çalışma, projenin maliyet ve verimini hesaplayarak optimizasyon algoritmalarıyla çizelgelenmesini ele almaktadır. Amaç en az maliyet ve performansı en iyi çalışanlar ile projeyi tamamlamaktır. Yazılım geliştirme süreçlerinden sarmal ve çevik model çalışma içerisinde anlatılmıştır. Ardından kısıtlı kaynaklar ile proje çizelgeleme ve sezgisel yaklaşımlar açıklanmıştır. Daha sonra sezgisel metotlar arasındaki Genetik Algoritma, Isıl İşlem Algoritması ve Karınca Koloni Algoritması için temel kavramlar anlatılmıştır.

Temel kavramlarından bahsedilen optimizasyon algoritmalarını kullanarak, sarmal ve çevik yazılım geliştirme süreci ile oluşturulmuş örnek aktiviteler kümesini proje kısıtlarını göz önünde bulundurarak çizelgeleyen bir c#.Net uygulaması geliştirilmiştir. Bu uygulama ile farklı aktivite ve insan kaynağı oluşturulmuş örnek sarmal ve çevik yazılım projeleri için optimizasyon algoritmaları aynı girdiler ile 5 kez çalıştırılmış ve sonuçları çalışma zamanı, maliyet düşüşü ve verim artışı bakımından karşılaştırılmıştır.

Yapılan testler sonucunda, sarmal yazılım geliştirme gibi aktivitesi fazla ve büyük yazılım geliştirme modellerinde genetik algoritma daha üstün gelmiştir. Genetik algoritmadan sonra karınca koloni algoritması ve sonrasında ısıl işlem algoritması gelmektedir. Çevik yazılım geliştirme gibi daha küçük parçalara bölünerek oluşturulan az aktiviteli modellerde ise karınca koloni algoritmasının daha etkin olduğu gözlemlenmiştir. Sonrasında sonuçları karınca koloni algoritmasına yakın olan genetik algoritma ve son olarak ısıl işlem algoritması sıralamadaki yerini almıştır.

Anahtar Kelimeler: Sarmal Yazılım Geliştirme, Çevik Yazılım Geliştirme, Genetik Algoritma, Isıl İşlem Algoritması, Karınca Koloni Algoritması

ABSTRACT

OPTIMIZATION AND COMPARING SPIRAL AND AGILE SOFTWARE DEVELOPMENT SCHEDULING BY USING HEURISTIC ALGORITHMS

Mahmut Baş

Information Technology

Thesis Supervisor: Prof. Dr. Adem Karahoca

October 2015, 72 pages

The software development projects are getting more complex by time and evolving technology. The increasing complexity in software projects brought difficulty in management of human resources. In this work, scheduling of staff that will contribute in such complex software development project with cost and productivity calculations is handled. The purpose is to complete the project with lowest cost and highest with performing staff. Spiral software development and agile software development processes are used in this work. And then, heuristic approach and project scheduling with limited resources are defined, then, genetic algorithm, simulated annealing, ant colony algorithm in between heuristic methods are mentioned.

A c# .Net application is developed which contains spiral software development and agile software development processes sample. In this application, tasks and human resources are defined, optimization algorithms for spiral and agile software development are run 5 times with same input, and results are compared in terms of total work time, cost reduction and productivity increase.

Conclusion of tests showed that genetic algorithm has seen more powerful in spiral software development with lots of tasks and complex software development models. After genetic algorithm secondarily ant colony algorithm and thirdly simulated annealing algorithm comes. Ant colony algorithm is observed as more useful for less task models like agile software development. And then, simulated annealing algorithm comes in the end, with results close to ant colony algorithm.

Keywords: Spiral Software Development, Agile Software Development, Genetic Algorithm, Simulated Annealing, Ant Colony Algorithm

İÇİNDEKİLER

TABLOLAR vii
ŞEKİLLER viii
KISALTMALAR ix
SEMBOLLER x
1. GİRİŞ 11
2. KAYNAK ARAŞTIRMASI 3
3. YAZILIM GELİŞTİRME MODELLERİ 5
3.1 SARMAL MODEL 5
3.2 ÇEVİK YAZILIM GELİŞTİRME 7
4. KISITLI KAYNAKLAR İLE PROJE ÇİZELGELEME ve SEZGİSEL YAKLAŞIMLAR 11
4.1 AKTİVİTE VE KAYNAK İLİŞKİSİ 12
4.2 ÇEŞİTLİ AMAÇ İŞLEVLERİ 13
4.3 KESİN ÇÖZÜM YÖNTEMLERİ 13
4.4 SEZGİSEL YÖNTEMLER 14
4.4.1 Öncelik Kuralına Dayalı Sezgisel Yöntemler 14
4.4.2 Çok Çözüm Türeten Sezgisel Yöntemler 16
4.4.3 Meta-sezgisel Yöntemler 16
5. OPTİMİZASYON ALGORİTMALARI 19
5.1 GENETİK ALGORİTMA 19
5.1.1 Evrimsel Doğal Seçicilik ve Evrimsel Hesaplama 19
5.1.2 Genetik Algoritmanın Tanımı 20
5.1.3 Temel Kavramlar 23
5.1.4 Bireyler ve Uygunluk 23
5.1.5 Uygun Bireyleri Seçme 24
5.1.6 Çaprazlama 25
5.1.7 Mutasyon 26
5.1.8 Kontrol Parametreleri 26
5.2 ISIL İŞLEM ALGORİTMASI 28
5.2.1 Doğal Isıl İşlem 28
5.2.2 Yapay Isıl İşlem Algoritması Tanımı 28
5.3 KARINCA KOLONİ ALGORİTMASI 31

5.3.1 Doğal Karınca Davranışları 32
5.3.2 Karınca Koloni Algoritması Tanımı 34
5.3.3 Modelleme ve Temel Parametreler 35
6. VERİ VE YÖNTEM 39
6.1 VERİLERİN OLUŞTURULMASI 39
6.1.1 Proje Uygulama Şablonu 39
6.1.2 Varsayımlar 40
6.2 OPTİMİZASYON ALGORİTMA YÖNTEMLERİ 41
6.2.1 Genetik Algoritma Optimizasyonu 41
6.2.2 Isıl İşlem Algoritması Optimizasyonu 42
6.2.3 Karınca Koloni Optimizasyonu 42
7. BULGULAR 44
8. TARTIŞMA 50
9. SONUÇ VE ÖNERİLER 53
KAYNAKÇA 55
EKLER 60
EK 1: Örnek Sarmal Proje İçeriği 61
Ek 1.1: Sarmal proje insan kaynağı ve verimlilik değerleri 61
Ek 1.2: Sarmal proje örnek aktivite adımları 62
Ek 1.3: Sarmal proje adımları 63
Ek 1.4: Sarmal proje çalışanın uygun olduğu zaman seçimi 64
EK 2: Örnek Çevik Proje İçeriği 65
Ek 2.1: Çevik proje insan kaynağı ve verimlilik değerleri 65
Ek 2.2: Çevik proje örnek aktivite adımları 66
Ek 2.3: Çevik proje adımları 67
Ek 2.4: Çevik proje çalışanın uygun olduğu zaman seçimi 68
EK 3: Yazılım Geliştirme Projesi Test Uygulaması 69
Ek 3.1: Test uygulaması ana ekranı 69
Ek 3.2: Genetik algoritma test ekranı 70
Ek 3.3: Isıl işlem algoritması test ekranı 71
Ek 3.4: Karınca koloni algoritması test ekranı 72

TABLOLAR

Tablo 5.1: GA için önerilen kontrol parametre değerleri..... 27
Tablo 7.1: Sarmal ve çevik yazılım geliştirme için optimizasyon algoritması karşılaştırmaları 48

ŞEKİLLER

Şekil 3.1: Sarmal model üretim süreci 6
Şekil 3.2: Çevik yazılım adımları 8
Şekil 5.1: GA akışı 22
Şekil 5.2: Gerçek karıncaların en kısa yolu bulma aşamaları 33
Şekil 7.1: GA ile sarmal ve çevik yazılım geliştirme 45
Şekil 7.2: İİA ile sarmal ve çevik yazılım geliştirme 46
Şekil 7.3: KKA ile sarmal ve çevik yazılım geliştirme 47
Ek 3.1: Test uygulaması ana ekranı 69
Ek 3.2: Genetik algoritma test ekranı 70
Ek 3.3: Isıl işlem algoritması test ekranı 71
Ek 3.4: Karınca koloni algoritması test ekranı 72

KISALTMALAR

ÇS : Çalışma Süresi

GA : Genetik Algoritma

İİA : Isıl İşlem Algoritması

KKA : Karınca Koloni Algoritması

SEMBOLLER

Boltzman sabiti	: k
En iyi çözümün sonraki yinelemelere aktarılma olasılığı	: Q_0
Feromon buharlaşma oranı	: ρ
Feromon kuvvetlendirme oranı	: α
(i,u) arasındaki feromon iz miktarı	: τ (i,u)
Karıncanın toplam tur uzunluğu	: L^k
Sezgisellik kuvvetlendirme oranı	: β
T sıcaklığında enerjideki olabilecek artış olasılığı	: AE

1. GİRİŞ

Dünyada kaynakların azaldığı son yıllarda, faaliyet çizelgesi yapılırken kaynakların sınırsız olarak kullanılması düşünülemez. Kaynaklar az, harcanacak zaman önemli olduğu için kaynakların doğru şekilde yönetilmesi ve yönlendirilmesi her zaman avantaj sağlayacaktır.

Proje, bir problemi çözmek veya yeni bir ürün geliştirmek için, başlangıcı ve bitişi beli olan, kalite, risk, kapsam, bütçe ve zaman yönetimi içerisinde, amaca yönelik planı başlatma aşamasından sonlandırma aşamasına kadar geçen süreçtir. Proje yapısı içerisinde zaman, maliyet ve kaynaklar arasında bütünleşik bir yapı bulunmaktadır. Bir projedeki kullanılabilecek kaynakların çoğu başka bir proje için kullanılmaktadır veya farklı problemler nedeniyle projede zamanında katılamayabilir. Proje yöneticileri, projelere kaynak atamalarını zaman içerisinde klasik optimizasyon yöntemleriyle yönetmişlerdir. Son zamanlarda, proje girdileri, projelerin büyüklüğü ve karmaşıklığı klasik yöntemlerin ihtiyacı karşılayamayacağı derecede artmıştır. İhtiyaç duyulan optimizasyon yöntemleri için aranacak çözüm uzayı çok büyük, uygun çözüm için değişken sayısı ve girdi sayısı çok fazla ve en uygun çözümün bulunması neredeyse imkansızlaşmıştır.

Bu çalışma, kaynak kısıtlı projelerde kaynakların işlere maliyet ve verimlilik çerçevesinde atanmasını ele almaktadır. Proje çizelgelemesi esnasında kaynakların özel durumlarının olması da kısıtlar içerisinde incelenmektedir. Proje yönetim metotlarından sarmal model ve çevik model süreçleri incelenmekte, kısıtlı kaynaklar ile proje planlama ve sezgisel yaklaşımlar hakkında genel bilgiler paylaşılmaktadır. Kısıtlı kaynakların projelere atanması genetik algoritma, ısıl işlem algoritması ve karınca koloni algoritması ile verimi arttırılmış çizelge oluşturularak yapılmaktadır. Olası çizelgeleme geliştirme örneği ile simüle edilen bu optimizasyon algoritmaları sarmal ve çevik proje yönetim modelleri için karşılaştırılmıştır.

Sonuçta, geliştirilen bu yöntem ve uygulama ile ticari veya kurumsal olarak çalışan bir proje yönetim ofisi veya danışmanlık firması kendi kaynaklarını ve kısıtlarını göz önüne alarak, ihalesine gireceği bir proje için ihtiyaç duyulacak maliyeti öngörebilecek ve bu doğrultuda gerekli tedbirleri alabilecektir.

2. KAYNAK ARAŞTIRMASI

Çizelgeleme, kaynaklar üzerindeki işlerin belirli bir kriter doğrultusunda, kısıtlamaları da göz önünde bulundurarak zamanlaması ve sıralanması olarak tanımlanabilir. Genel bir ifade olarak, planlama içermediği halde çizelgeleme işlerin sıralamasını da içermektedir denilebilir (Özdemir, 2006).

Kaynak kısıtlı proje çizelgeleme problemi Özdamar ve Ulusoy (1995) tarafından detaylıca araştırılmış ve problemi farklı yönlerle ele alan kapsamlı bir çalışma ortay acıkmıştır. Bu çalışmaya göre, maiyet temelli amaçların çözümünde karşılaşılan problemlerden dolayı araştırmacılar genellikle zaman temelli problemleri çözmeye yönelmiştir. Oysa, maliyet göz önünde bulundurulmadan yapılan çalışmalar yeterli olmayacaktır.

Eliiyi (2004) özdeş paralel makinalarda iş çizelgelemesi problemi üzerine çalışmalar yapmıştır. Çizelgelemede kısıtlayıcı girdiler, zamana bağlı kısıtlar ve makine iş yoğunlukları olarak tasarlanmıştır. Bütün kısıtlar göz önünde bulundurularak dal ve sınır yöntemi kullanılıp verimli sonuçlar elde edilmeye çalışılmıştır. Yapılan testlerin sonucunda 100 işe kadar olan çalışmalarda dal ve sınır yöntemi uygulanabilir zaman diliminde, en iyi çözümleri sağlamıştır.

Kesin çözüm yöntemleriyle çözülebilen optimizasyon problemlerinin girdilerinin büyüklüğünün belirli bir seviyeye kadar çözülebilmesi ve çözüm için ihtiyaç duyulan zamanın girdilere paralel artması, araştırmacıları en iyi çözümü sağlamasa da en iyiye yakın çözümler sağlayacak metotlar uygulamaya yöneltmiştir. Yapay zekanın optimizasyon yöntemleri aynı zamanda meta sezgisel yöntemler adıyla bilinmektedir. Bu yöntemler, iyi bir çözüm sağlamak için çeşitli alternatif çözümlerden amaca en yakın olanı tanımlayıp bulmayı sağlayan bilgisayar tabanlı çözümlerdir. Yapay zeka optimizasyon algoritmalarının uygulama sonuçları aynı zamanda kesin çözüm metotları için bir alt veya üst sınır oluşturmaları nedeni ile de önemlidir. Bu tür çözüm algoritmaları kesin çözümü garanti etmezler ancak amaca uygun bir çözümü bulmayı garanti edebilirler (COŞKUN, 2007).

Santos ve Zhong (2001), sabit iş çizelgeleme kısıtlı bir yapıda minimum maliyeti bulan takviyeli öğrenme sağlayan genetik algoritma tasarlamışlardır. Tasarladıkları algoritma, minimum maliyet için uygunluk değerlerini en iyi seviyelere çıkartmıştır.

Verme ve Singhal (2009) akış tip iş çizelgeleme problemi için genetik algoritma kullanımı ile klasik yöntemi incelemişlerdir. Elde edilen sonuçlar toplam tamamlanma süresi ve makinelerin bekleme sürelerinin minimize edilmesinde genetik algoritmanın klasik yöntemden daha iyi sonuçlar ürettiğini göstermiştir.

Isıl işlem algoritması, genetik algoritmalar ve karınca koloni algoritması (KKA) kompleks optimizasyon çalışmaları için tasarlanmış sezgisel yöntemlere örnektir (Pham & Karaboğa, 2000).

Karınca kolonisi algoritması kombinasyonel optimizasyon problemlerinin çözülmesi için tasarımlanmış meta sezgisel bir algoritmadır. Algoritma ilk olarak 1991 yılında çalışılmıştır, sonrasında ise literatürde birçok uygulaması rapor edilmiştir (Dorigo & Gambardella, 1997).

Adewole ve arkadaşları (2012) genetik algoritmanın ısıl işlem algoritmasına göre daha uzun süre çalışmasına karşın daha iyi sonuçlar getirdiği üzerine çalışmalar yapmıştır.

Purian ve arkadaşları (2013) dinamik girdilere karşılık genetik algoritma ve karınca koloni algoritmasını karşılaştıran çalışma yapmışlar ve bu çalışmanın sonucunda karınca koloni algoritmasının yakın sonuçlar ile genetik algoritmaya göre daha iyi ve daha hızlı sonuçlar verdiğini belirtmişlerdir.

3. YAZILIM GELİŞTİRME MODELLERİ

Bilgisayarların 1940'lı yıllarda kullanılmaya başlanması ile birlikte yazılım geliştirme süreçleri de kullanılmaya başlamıştır. Yazılımın ilk yıllarında projelerin zamanında bitirilememesi ve müşterinin ihtiyaç duyduğu kalitede geliştirilememesi başlıca sorunlar olmuştur. Teknolojinin hızla değişmesi, birçok yazılımın hız ve fonksiyonellik eksikliklerinden dolayı devre dışı kalmasına ve işin yeniden projelendirilmesine neden olmuştur (Beydağlı, et al., 2009). Teknolojinin gelişimi ile birlikte daha büyük problemleri çözebilecek yazılımlar geliştirilmeye başlamıştır. Yazılım girdilerinin sürekli artması daha gelişmiş yeni yazılım yöntemlerini ve bu yöntemlerin uygulandığı süreç modellerini ortaya çıkarmıştır. Yazılım projesi için en uygun modelin seçilmesi, projenin daha güvenli, tutarlı, anlaşılabilir, analiz edilebilir ve kolayca uygulanıp test edilebilir olmasını sağlar. Kalitesi yüksek yazılım projelerinin az maliyet ile kısa sürede gerçekleştirilebilmesi için devamlı olarak yeni teknolojiler ve bu yeni teknolojiler ile sürekli gelişmiş yeni modeller kullanılmalı, var olan modeller sürekli geliştirilmelidir.

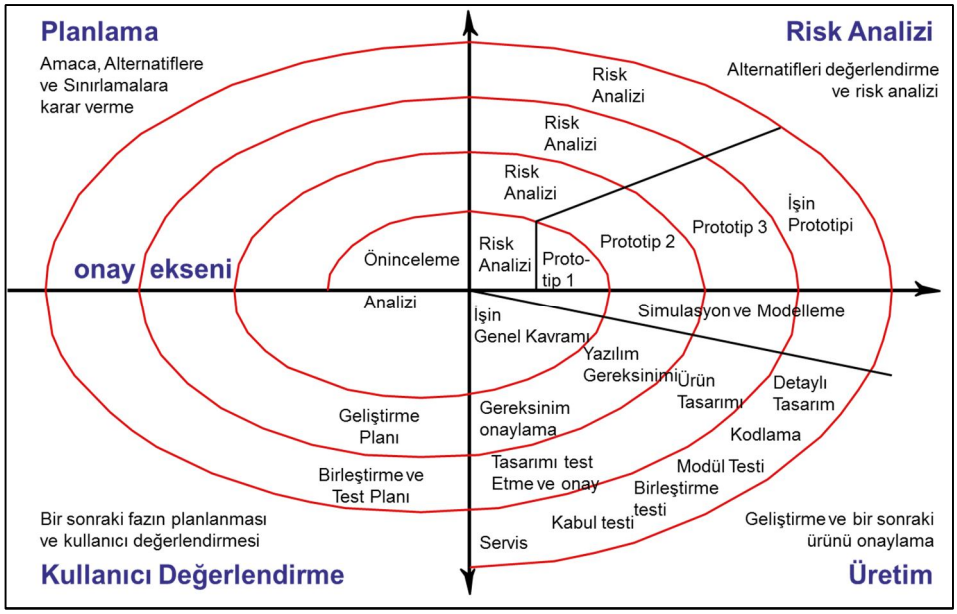
Proje yönetiminin ilk ve en önemli kararı yazılım sürecinde kullanılacak modelin seçimidir. Bir modelin seçilmesi, seçilmemesine göre daha iyi sonuçlar getirirken, yanlış bir modelin seçimi bazen ek problemler ortaya çıkartmaktadır. Çevik yazılım, şelale metodu, barok metodu, artırımsal ve gelişigüzel geliştirme belli başlı yazılım geliştirme modelleri olarak sıralanabilir.

3.1 SARMAL MODEL

Sarmal model temeli Barry Boehm tarafından 1988 yılında oluşturulmuştur. Spiral modeli, şelale modelinden farklı olarak belli ve düz bir akıştan ziyade kademeli olarak geliştirilen bir yazılım modelidir. Şekil 4.1' de görüldüğü gibi proje çevrimlere ayrılır. Her çevrimin riskleri ayrı ayrı ele alınır. İlk aşama olan planlama aşamasında üretilecek prototip için iş paketlerinin oluşturulması, gereksinim, kısıtlar ve alternatif çözümler üretilir. Ayrıca önceki çevrim sonucu oluşan prototip ile yeni oluşturulacak geliştirmelerin birleştirilmesi aşamasıdır. Risk analizi aşamasında projeye ve ürüne ait riskler çıkartılır ve risk planı oluşturulur, risk cevapları tasarlanır, alternatifler

değerlendirilir. Üretim aşaması, prototipin geliştirmesinin yapıldığı aşamadır. Kullanıcı değerlendirme aşamasında ise prototipin özelliklerinin son kullanıcı tarafından test edilmesi ve müşterinin ihtiyacının karşılanması durumunda prototipin kabulü yapılır.

Şekil 3.1: Sarmal model üretim süreci



Kaynak: (Yılmaz, 2007)

Sarmal yazılım geliştirme modeli, her çevrimde risk analizleri ve cevapları oluşturularak bir ara ürün yapmaya dayanmaktadır. Her çevrimin çıktısı, bir sonraki çevrimin planlama aşamasında girdi olarak kullanılır. Sarmal model, bir önceki prototipin üzerine geliştirme temelli olduğu için bir kere yapılmış ama yeniden kullanılabilen uygulamalar için entegrasyona çok uygundur. Daha önceden yapılmış uygulamalar prototip üzerine eklenerek ihtiyaçlar karşılanabilir.

Sarmal model, zaman, kalite ve maliyet değerleri önceden kolaylıkla tahmin etmede başarı bir yöntemdir. Bunu sağlayan temel öğe ise risk analizlerinin her çevrimin başında yapılmasıdır. Yazılımın geliştirilmesi ve test edilmesi, ilk ara ürün ile birlikte projenin erken safhalarında başlar ve ürünü kullanacak personelin projeye erken aşamalarda katılması durumunda ileride yaşanabilecek problemlerin önüne geçilir. Sarmal yazılım geliştirme modelinin dezavantajları;

a. Küçük çaplı, basit seviyedeki projelerde kullanılması durumunda yeterli verim alınamayabilir.

b. Bu modeli kullanacak proje yöneticisin sarmal model ile yazılım geliştirme sürecinde tecrübeli olması, adımların tam olarak uygulanması için gereklidir.

c. Risk analizleri nedeniyle oluşturulan risk cevapları, sözleşmeler ve yaptırımlar ile alt yüklenicilerin işini veya alt yüklenici bulunmasını zorlaştırabilir.

3.2 ÇEVİK YAZILIM GELİŞTİRME

Yeni bir yazılım geliştirmek yeni bir ürün yaratmak olduğu için yazılım projelerinin gereksinimleri proje sürecinde değişerek şekillenir. Projelerdeki birçok problem, başlangıçta öngörülemeyebilir. Bu yüzden, yazılım geliştirirken mevcut klasik yöntemlerin dışında daha esnek ve dinamik yöntemlere ihtiyaç duyulmaktadır. Yazılım projelerinin gerçek hayattaki durumları incelendiğinde;

i. Projeni ilk adımlarında müşteriler ne istediğini veya nasıl isteyeceğini bilemez

ii. Müşteriler isteklerini açık bir şekilde ifade edemeyebilirler.

iii. Müşterilerin asıl istekleriyle ilgili detaylar proje geliştirildikçe ortaya çıkar.

iv. Müşterilerin talepleri gerçekleştirildikçe ve projede ilerlendikçe, müşterinin taleplerinde değişiklikler ortaya çıkabilir.

v. Dış koşullar değişiminden dolayı projelerde değişiklik meydana gelebilir.

Bu nedenlerden dolayı yazılım projeleri müşteri odaklı olmalı ve değişikliklere uyum sağlayabilmelidir. Eski yöntemlerin kullanıldığı yazılım geliştirme projeleri, değişimler karşısında güncellenmekte zorlandığı için projelerde hedeflenen sonuca ulaşmak zorlaşmaktadır.

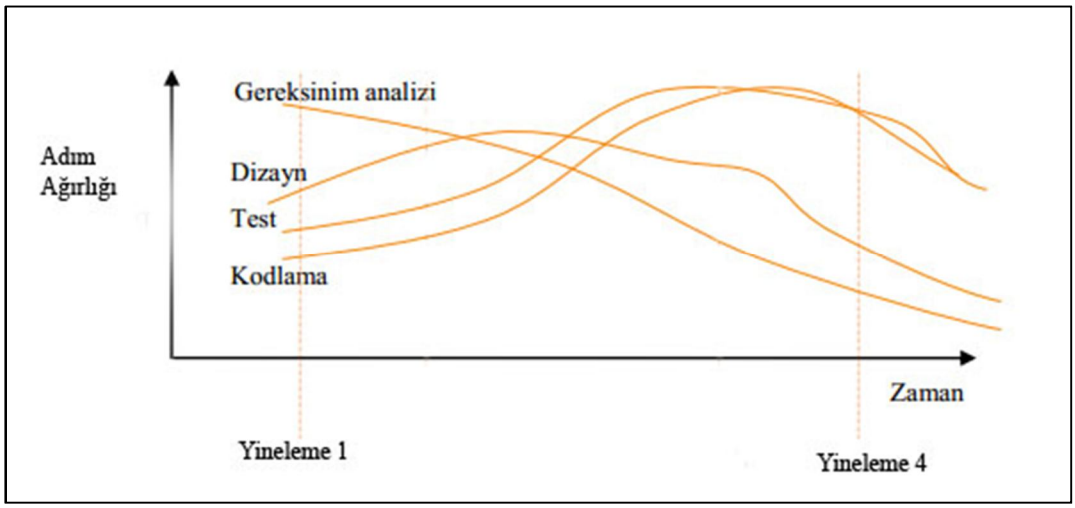
Çevik yazılım yöntemleri, 20. Yüzyılın ortalarında üretim aşamasında kullanılan yalın yöntemlerin, yazılım projelerinde de kullanılmasıyla doğmuştur. Yazılım sektöründe çevik metotların kullanımı 1970 yılı sonrasında şekillenmeye başlamıştır. Çevik yazılım metotları son 25 yıl içerisinde artarak dünya genelinde, başarılı projelerde kullanılmıştır. Günümüzde çevik yazılım yöntemleri çoğu yazılım projesi geliştirme aşamasında kullanılmaktadır.

Çevik yazılım geliştirme, sarmal yöntemlerde olduğu gibi tekrarlama-temeline dayanır. Kısa zaman aralıklarında, adım adım ürün teslimatının yapılmasını ve üründe müşterinin taleplerine göre değişiklik yapılmasını sağlar. Çevik yazılım geliştirme değişikliği en ileri seviyede destekleyen bir yöntemdir. İletişim, özellikle takım içi iletişim ön planda tutulur. Müşteri aynı zamanda geliştirmenin bir parçası haline getirilerek ile birlikte testler yapılarak, test odaklı çalışılır.

Tekrarlanan yazılım geliştirme metodu, projenin adımlarının sıralı olarak yinelendiği ve geri bildirimler oluşturularak adım adım tamamlandığı bir yazılım geliştirme modelidir. Proje içerisindeki iş adımları küçük parçalara bölünerek yazılımın geliştirmesi yapılır. Küçük parçalara bölünme sonucu oluşan her bir yineleme ayrı bir yazılım projesi olarak değerlendirilebilir. Nihai yazılım projesi, bu küçük yineleme parçalarının birleştirilmesiyle oluşur.

Yinelemeli modelde her bir yineleme analiz, tasarım, yazılım geliştirme, test işlemlerini içerir. Her yinelemede, yapılan bu işlem adımlarının kapsamı ve büyüklüğü değişebilir. İlk yinelemelerde analiz işleminin ağırlığı daha fazla iken ileriki yinelemelerde kodlamanın ağırlığı daha fazladır. Bu adımlar içerisindeki ağırlıkların zaman içerisindeki değişimi aşağıdaki şekilde gösterilmektedir:

Şekil 3.2: Çevik yazılım adımları



Kaynak: ACM YAZILIM ÇÖZÜMLERİ,ÇEVİK YAZILIM GELİŞTİRME, http://www.acm-software.com/Pdf/AboutAgile.pdf [erişim tarihi 10 Aralık 2014], s.7

Bir projedeki yineleme sayısı proje içeriğinin büyüklüğüne, müşterinin talebinin netliğine ve uygulanan metotlara göre değişkenlik gösterir. Genel olarak kullanılan modeller incelendiğinde yineleme zamanı 1 ile 6 hafta aralığında sürmektedir.

Yazılım projelerinin kapsamı arttıkça proje kompleks bir yapıya dönüşmekte ve projelerin müşterinin ihtiyacını karşılaması zorlaşmaktadır. Bu yüzden yapılacak çalışmaları karmaşıklıktan kurtarmak ve riski azaltmak için projeler daha küçük yinelemeli yazılım parçalarına ayrılır. Bu şekilde uygulanan kullanım verimliliği ve projenin başarıyla sonuçlanma olasılığını artırmaktadır. Yinelemeler arasındaki müşteriden alınan geri bildirimler, ihtiyacın tam anlaşılması ve doğru adımların atılması için büyük öneme sahiptir.

Ortak bir standart oluşturulması için 2001' de bir araya gelerek çevik yazılım geliştirme manifestosunu ve prensiplerini yayınlamışlardır. Bu manifesto ile çevik yöntemlerin özellikleri çevik yazılım geliştiricileri tarafından ortak olarak açıklanmıştır. Bu manifestoda bireyler ve aralarındaki etkileşimlerin, kullanılan araç ve süreçlerden; çalışan yazılımın, detaylı dokümantasyondan; müşteri ile işbirliğinin, sözleşmedeki kesin kurallardan; değişikliklere uyum sağlayabilmenin, mevcut planı takip etmekten daha önemli ve öncelikli olduğu belirtilmektedir. Çevik yazılımın prensipleri 12 madde olarak aşağıda belirtilmiştir:[1]

i. Projede en önemli öğe, müşteri memnuniyetini kaliteli bir çıktıyla kazanmaktır.

ii. Projede her zaman değişiklikler olabilir.

iii. Sık zaman periyodunda kaliteli ve çalışan çıktılar oluşturulur.

iv. Ekip içi iletişim önemlidir ve ekibin tüm üyeleri iletişim halindedir.

v. Projede çalışan ekibe güvenilmeli ve moral yükseltici çalışmalar yapılmalıdır.

vi. Ekip elemanlarının yüz yüze iletişim kurması projedeki iletişim kanallarının doğru kullanılması için önemlidir.

vii. Çalışan uygulama, çevik yazılım sürecinde, yazılım projesinin temel ölçütüdür.

[1] *Çevik Yazılım Geliştirme Manifestosu.* 2001. http://agilemanifesto.org/iso/tr/ [erişim tarihi 13.06.2012].

viii. Sabit hızlı, devamlılığı sağlanan yazılım geliştirme ön plandadır.

ix. Alt yapının sağlam olması ve tasarımın doğru yapılması hızı ve atikliği artırır.

x. Sadelik her zaman önemlidir.

xi. Kendiliğinden organize olan ekip üyeleri tarafından doğru tasarlanmış, gereksinim analizleri doğru yapılmış projeler çıkar.

xii. Ekip üyeleri tarafından sık periyotta gözden geçirme işlemi yapılır ve daha iyi sonuca ulaşmak için düzenlemeler yapılır.

Çevik yöntemlerin avantajları aşağıdaki gibi sıralanır:

a. Projenin ilk adımlarında öngörülemeyen riskler büyük sorunlara neden olmadan önlenebilmekte ve projenin şekillenmesine göre düşük risk oluşturulmaktadır.

b. Müşterinin taleplerine göre değişim, sürecin bir parçası haline getirilir ve her adımın başlangıcında değişimler fark edilerek mevcut yapıya adapte edilmesi sağlanır.

c. Büyük projeler küçük parçalara bölünerek karmaşıklık en düşük seviyeye indirilir.

d. Her bir yinelemenin sonunda çalışır bir programcık meydana getirilmesi müşterinin memnuniyetini arttırmaktadır.

e. Test süreçlerinin yinelemelerin içerisinde yapılmasıyla uygulamadaki sorunlar büyümeden ortaya çıkartılıp hızlıca çözümler sağlanmaktadır.

f. Yinelemeler esnasında müşteri ile yapılan fikir alış verişleri ile müşterinin ihtiyaçlarını en iyi derecede karşılayabilecek programlar ortaya çıkartılmaktadır.

4. KISITLI KAYNAKLAR İLE PROJE ÇİZELGELEME ve SEZGİSEL YAKLAŞIMLAR

Çizelgeleme, kaynaklar üzerindeki işlerin belirli bir kriter doğrultusunda, kısıtlamaları da göz önünde bulundurarak zamanlaması ve sıralanması olarak tanımlanabilir. Genel bir ifade olarak, planlama içermediği halde çizelgeleme işlerin sıralamasını da içermektedir denilebilir (Özdemir, 2006).

Kapasite planlama, iş yükünün belirlenmesi ve kaynaklar üzerinde yapılacak olan işlemlerde ortaya çıkan kapasite problemlerinin çözülmesi olarak tanımlanabilir. Kapasite planlama problemleri, farklı türde yöneylem araştırması teknikleri kullanılarak modellenebilir ve çözülebilir; ama bu yaklaşımlarda detaylar noktasına gelindiğinde iki güçlükle karşılaşılır:

1. Karmaşıklık artar,
2. Detay arttıkça, o detayla ilgili ihtiyaç duyulacak bilginin elde edilme olasılığı azalır.

Genellikle çıkış yolu, problemi olduğu gibi sezgisel şekilde modelleme ve çözmeye tepeden başlamaktır. Her bir kademede çözüm bir alt kademedeki çözüm tarafından sınırlandırılır ve problem bütünü için verimli bir sonuç elde edileceği umulur. Sonuçta geçerli bir yargı oluşması için, bu sınırlar kayıt edilerek eldeki verilerin mümkün olduğu kadar detaylandırılması gerekmektedir.

Blazewicz ve arkadaşları (1983) tarafından açıklandığı şekilde, kısıtlı kaynaklarla proje çizelgelemesi problemleri, zor sayısal optimizasyon problemleri sınıfına ait olan klasik iş yeri çizelgeleme probleminin bir genellemesidir. Bu nedenle sezgisel çözüm prosedürleri, genellikle uygulamada meydana gelen büyük problem örneklerinin çözülmesi sırasında vazgeçilmezdir. 1960'larda Kritik Yol Metodu geliştirildiğinden bu yana, literatürde çok sayıda farklı sezgisel algoritmaların önerildiği görülmektedir.

Kaynağın kısıtlı olduğu projelerde çizelgeleme, kısıtlı kaynaklarla projenin iş paketlerinin, bir biri ile olan öncelik ilişkilerini göz ardı etmeden, proje planını en verimli haliyle oluşturmaktır.

Problemin tanımına temel teşkil eden varsayımları aşağıdaki gibidir:

a. Aktivite süreleri belirsizlik içermez.

b. Aktivitelerin birim zaman kaynak kullanımı sabittir.

c. Bir aktiviteye atanan kaynak, aktivite süresince o aktivite tarafından kullanılır.

d. Başlayan aktivite zinciri durmaksızın bitirilmek zorundadır; ara verilemez.

e. Aktiviteler iptal edilemez. Proje planındaki her aktivite gerçekleştirilmek zorundadır.

Bu varsayımların bazıları kaldırılarak daha farklı problem tanımlarına gitmek mümkündür. Örneğin, aktivite sürelerinin rasgele olması durumunda araştırma ve geliştirme projeleri gibi aktivitelerinin içeriği tam olarak bilinemeyen projeler için daha doğru yaklaşımlar yapılabilmektedir. Bir kaynağın eş zamanlı olarak birkaç aktivite de olduğu durumlar olabilir.

4.1 AKTİVİTE VE KAYNAK İLİŞKİSİ

Bir aktivitenin gerçekleştirilme süresi ile kullanılan kaynaklar arasında bir ilişki vardır. Genel olarak, birim zaman içinde daha çok kaynak kullanımının maliyeti yükseltmesi, faaliyet süresini ise azaltması beklenir. Proje aktivitelerinin tamamlanması bu maliyet ve süre etkileşimine göre modellenebilir. Kullanılan kaynağın parçalı veya sürekli olmasına bağımlı olarak maliyet ve süre etkileşimi parçalı veya sürekli bir işlev olarak ifade edilir. Parçalı işlev durumunda, işlevin her bir maliyet ve süre çiftine karşı gelen noktası bir model olarak nitelendirilir (Ulusoy, 2000).

Diğer bir bütünleştirme türü kaynaklar arasındaki bütünleştirmedir. Burada bir aktivitenin süresi sabittir ancak değişik kaynakların değişik kullanımları söz konusudur. Örneğin bir bina yapımı projesinde temel atma aktivitesinde bir iş makinası, bir kamyon ve iki operatör yerine aynı süre içinde otuz düz işçi tarafından yapılması verilebilir.

Proje yönetiminin uygulamalarında genellikle birden fazla projenin aynı kaynaklar ile yürütülmesi sağlanır. Bu tür problemler, çok projeli çizelgeleme problemleri olarak isimlendirilirler.

4.2 ÇEŞİTLİ AMAÇ İŞLEVLERİ

Kaynak kısıtlı proje çizelgeleme probleminde çeşitli amaç fonksiyonları uygulanmıştır. Proje yöneticisi olarak incelendiğinde en önemli başarı kriterlerinden birisi projenin tamamlanma süresidir. Aktivitelerin eldeki kaynaklarla en az sürede tamamlanması hedeflenir. Kaynak kısıtlarının olmadığı düşünülen projelerde amaç fonksiyonu sürenin en aza indirilmesi olacaktır. Kaynak kısıtlı proje çizelgeleme problemlerinde en yaygın kullanılan amaç fonksiyonu proje süresinin en aza indirilmesidir. Projenin gecikmesinin en aza indirilmesi, erken bitirme, gecikme toplamının en aza indirilmesi gibi çeşitli amaç işlevleri de geliştirilmiştir.

Sık kullanılan bir diğer amaç fonksiyonu da toplam maliyetin en aza indirilmesi veya toplam karın en verimli olduğu seviyeye getirilmesidir. Bu amaç fonksiyonu kullanılırken, nakit akışına paranın zaman değeri uygulanarak bugünkü değeri bulunur ve maliyet veya kar, net bugünkü değer olarak ifade edilir.

4.3 KESİN ÇÖZÜM YÖNTEMLERİ

Proje süresinin en aza indirildiği kaynak kısıtlı proje çizelgeleme problemi için kesin çözüm yöntemleri ancak belirli problem büyüklükleri için geçerli olabilmektedir.

Çok sayıda çalışmada dal sınır yöntemi kullanılmaktadır. Bir düğümdeki kısmi çizelgede devam etmekte olan faaliyetlerin en erken biteninin bitiş zamanı, o düğümden çıkacak düğümlerin çizelgeleme zamanını belirler. Yeni yaratılan düğümler henüz atanmamış faaliyetlerin olurlu altkümeleridir. Düğüm seçimi hiyerarşik olarak uygulanan altı seçim kuralı ile gerçekleştirilir. Da-sınır ağacını budamak için iki baskın kuralı kullanılmıştır (Stinson, et al., 1978).

4.4 SEZGİSEL YÖNTEMLER

Kesin çözüm yöntemleri ile kısıtlı büyüklükteki problemler için çözüm sağlanabilir. Üstelik bu çözümlerin sağlanması da uzun süreler almaktadır. Bu yüzden araştırmacılar daha büyük proje çizelgelemeleri hızlı çözüm üretebilen yapılara ihtiyaç duymuştur. Hızlı bir şekilde en ideal çözüme yakın bir çözüm bulunmasını sağlayan sezgisel yöntemler üzerinde çalışmalar artmıştır.

Sezgisel yöntemleri üç ayrı başlık halinde incelenmektedir. Birinci grup sezgisel yöntemler, öncelik kuralları kullanarak çizelge oluşturan yöntemlerdir. Bir defa uygulanarak ulaşılan çözüm nihai çözüm olarak kabul edilir. İkinci grupta ise, öncelik kuralları kullanarak veya örnekleme yolu ile birden çok uygulama yapılır ve birden çok çözüm elde edilir. En iyi çözüm nihai çözüm olarak kabul edilir. Meta-sezgisel yöntemlerin son senelerde öne çıkışı ve yaygın kullanımı nedeni ile bu sezgisel yöntemleri ayrı bir başlık altında üçüncü grup olarak incelenmektedir (Köksalan & Erkip, 2000).

4.4.1 Öncelik Kuralına Dayalı Sezgisel Yöntemler

Öncelik kurallarına dayalı sezgisel yöntemlerin iki boyutu vardır: Kullanılan öncelik kuralı ve çizelgenin oluşturulma yöntemi. Tüm aktiviteler öncelik kuralları kullanılarak atandıktan ve böylece öncüllük ilişkileri ve kaynak kısıtları açısından anlamlı bir proje planı hazırlandıktan sonra algoritmanın işletilmesi durdurulur.

Performans açısından değerlendirmeyi sağlayabilecek bazı sezgisel öncelik kural önerileri aşağıdaki gibi olabilir:

a. En kısa işlem süresi

b. En kısa boşluk süresi

c. En çok sayıda toplam ardıl

d. En büyük kaynak talebi

e. En geç bitirme zamanı

f. En geç başlama zamanı

g. Kaynak çizelgeleme metodu

h. En büyük yerel ağırlık

i. En kötü alternatif boşluk

Çizelgenin oluşturulmasında ikiye ayrılır: Paralel çizelgeleme ve seri çizelgeleme.

Paralel çizelgeleme en fazla aktivite adedi kadar aşamadan oluşur. Her aşamada, bir veya daha fazla faaliyet kısmi çizelgeye atanır. Uygulama esnasında aktiviteler dört küme içinde yer alırlar. Bitmiş aktiviteler kümesi, atanmış fakat henüz tamamlanmamış süregiden aktiviteler kümesi, kaynak kısıtlarını ihlâl etmeyen ve öncülleri bitmiş aktivitelerin oluşturduğu karar kümesi ve bunların dışında kalan aktivitelerin oluşturduğu kalan aktiviteler kümesi. Her aşamada, çizelgeleme anı süregiden aktiviteler kümesindeki faaliyetlerden en erken biten aktivite tarafından belirlenir. Bu aktivite bitmiş aktiviteler kümesine eklenir. Karar kümesi mevcut duruma göre yeniden belirlenir. Kullanılan öncelik kuralı uygulanarak karar kümesinden yeni atanacak aktivite seçilir ve atanır. Atama döngüsü, karar kümesinde atanacak faaliyet kalmayana veya mevcut kaynak kullanımına göre kaynak sınırlamasını ihlal etmeyen aktivite kalmayana kadar sürdürülür.

Seri çizelgeleme üç küme tanımlanmıştır. Bitmiş aktiviteler kümesi, öncülleri bitmiş aktiviteler kümesinde yer alan aktivitelerin oluşturduğu karar kümesi ve bunların dışında kalan aktivitelerin oluşturduğu kalan aktiviteler kümesi. Her aşamada bir aktivite atanır. Buna göre algoritma aktivite adedi kadar aşamada sona erer. Her aşamada karar kümesi içinden bir aktivite öncelik kuralı kullanılarak seçilir ve öncüllük ve kaynak kısıtları bakımından olurlu olduğu en erken anda atanır. Adı geçen üç küme güncellendikten sonra yeni bir aşamaya geçilerek algoritma atanacak aktivite kalmayana kadar sürdürülür (Kelley, 1961).

Paralel çizelgelemenin ertelemesiz çizelgeler türettiği; seri çizelgelemenin ise aktif çizelgeler türettiği gösterilmiştir. Düzgün bağımlı performans ölçütleri için aktif çözümler kümesi mutlaka en iyi çözümü içermekle birlikte ertelemesiz çözümler

kümesi en iyi çözümü içermeyebilmektedir. Bu açıdan bakıldığında seri çizelgeleme yöntemi en iyi çözümü bulmaya daha yakın durmaktadır (Kolisch, 1996).

4.4.2 Çok Çözüm Türeten Sezgisel Yöntemler

Çok çözüm türeten sezgisel yöntemlerde önceden belirlenmiş sayıda çizelgeleme türetilerek aralarından en iyisi seçilir. Bir çözüm elde etmenin çözüm süresinin azlığı, algoritmayı çok kere uygulamayı hesap süresi bakımından olurlu hale getirmiştir. Çok çözüm türeten sezgisel yöntemleri üç başlık altında incelenebilir: (a) Örnekleme, (b) öncelik kuralları, (c) başlangıç bitiş çıkışlı çizelgeleme

a. Örnekleme kullanılan yöntemde, rasgele örnekler oluşturularak bir çözüm kümesi içinde en iyi çözüm aranır. Atanacak aktivitelerin rasgele olarak seçildiği bu yöntemlerde seri veya paralel çizelgeleme kullanılır. Seri çizelgeleme kullanıldığında çözüm kümesi aktif çizelgeler kümesi; paralel çizelgelemede ise ertelemesiz çizelgeler kümesidir

b. Öncelik kuralları kullanılarak birçok çizelge oluşturulur ve uygunluğu test edilir. Bunlardan bir tanesi, türetilen her çizelge için ayrı bir öncelik kuralı uygulamaktır.

c. Başlangıç çıkışlı çizelgelemede, çizelgeleme proje başlangıcından itibaren oluşturulur ve faaliyetler atanır. Bitiş çıkışlı çizelgelemede ise, zaman yine başlangıç noktası olarak kabul edilir ama faaliyetler sondan başlanarak çizelgeye atanır.

4.4.3 Meta-sezgisel Yöntemler

Meta-sezgisel yöntemlerde, öncelikle, çözümün gösterimi ve bu gösterimi çizelgeye dönüştüren çizelgeleme yöntemi üzerinde durulacaktır. Bu gösterimlerden bir tanesi, rasgele anahtar gösterimidir. Rasgele anahtar gösteriminde, gösterimin her bir elemanı bir faaliyete ilişkin bir değere karşı gelir. Aktivite listesi gösterimi olarak nitelendirilen gösterimde, gösterimin her bir elemanı bir aktivitenin proje serimi üzerinde atanmış gösterim sayısına karşı gelir. Üçüncü gösterimde ise, gösterimin elemanlarını öncelik kuralları oluşturur: Bu gösterime öncelik kuralları gösterimi denir. Meta-sezgisel yöntemlere örnek olarak genetik algoritmalar (GA), ısıl işlem algoritması (İİA), karınca koloni algoritması (KKA) verilebilir.

Çözüm gösteriminin çizelgeye dönüştürülmesinde seri ve paralel çizelgeleme kullanılır. Paralel çizelgelemenin ertelemesiz çizelge türetmesi ve en iyi çözümün bunlar arasında olmaması olasılığı nedeni ile bazı paralel çizelgeleme uygulamalarında çizelgelenebilir aktivitelerin ertelenebilmesine olanak sağlanmıştır. Bu şekilde uygulanan paralel çizelgeleme, uyarlanmış paralel çizelgeleme olarak nitelendirilmektedir (Ulusoy, 2000).

Meta-sezgisel yöntemlerin bitiş kriteri olarak belirli bir süre, belirli bir sayıda alternatif çözüm veya amaç tanımlanarak bu amaca ulaşılması kullanılabilir.

Meta-sezgisel algoritmaların en iyileri, sezgisel karar kuralları kullanan algoritmaların en iyilerinden daha iyi sonuçlar vermektedir. Meta-sezgisel algoritmaların 2000 çözüm seviyesinde görülen çözüm seviyesinin 10000 çözüm seviyesinde daha da artması en büyük özelliğidir. Bunun nedeni, sezgisel algoritmalardan farklı olarak, meta-sezgisel algoritmaların yeni çözüm türetirken daha önce türetmiş olduğu çözümlerin bilgi birikiminden öğrenerek ilerlemesidir.

Meta-sezgisel algoritmaların en iyileri sezgisel öncelik kuralları kullanan algoritmaların en iyilerinden daha iyi sonuçlar türetiyor olmakla birlikte sezgisel öncelik kuralları yine de önemlidir. Sezgisel öncelik kuralları, çok büyük problemlerin çözümünde kullanıldıkları gibi, meta-sezgisel algoritmaların başlangıç çözümlerinin oluşturulmasında da kullanılmaktadırlar. Sezgisel öncelik kuralı kullanan algoritmalar arasında genellikle örneklemeye dayalı algoritmalar öne çıkmaktadır. En iyi sonuçların adaptif algoritmalar (Schirmer, 1998) ile alındığı görülmektedir.

Seri ve paralel çizelgeleme uygulamalarından; seri çizelgelemenin nispeten küçük problemlerde (30 aktivite gibi) daha iyi sonuçlara yol açtığı, paralel çizelgelemenin ise nispeten büyük problemlerde (120 aktivite ve üstü gibi) daha iyi sonuçlar verdiği gözlenmiştir. Bilindiği gibi, paralel çizelgeleme ertelemesiz çizelgeler türetir ve ertelemesiz çizelgeler kümesi en iyi çözümü içermeyebilir. Buna karşın, seri çizelgeleme aktif çizelgeler türetir ve aktif çizelgeler kümesi en iyi çözümü içerir. Kolisch (1995) bu gözlem ve tespitten hareketle; daha küçük problemlerde paralel çizelgelemenin en iyi çözümü dışlayabileceği, buna karşın seri çizelgelemenin en iyi çözümü daha kolaylıkla saptayabileceğini, daha büyük problemlerde ise paralel

çizelgelemenin daha küçük bir çözüm kümesi üzerinde çalıştığından daha iyi sonuçlara ulaşabileceğini belirtmektedir.

5. OPTİMİZASYON ALGORİTMALARI

5.1 GENETİK ALGORİTMA

Genetik algoritmalar evrime dayalı algoritmaların bir çeşididir. Bu bölümde öncelikle evrimsel algoritmalar hakkında genel bilgiler verilerek ardından genetik algoritma temel elemanları tanıtılacaktır. Genetik algoritma sürecinde uygulanabilecek değişik yöntemlerden bahsedilerek uygulama bölümünde genetik algoritma ile ilgili bir uygulama yapılacaktır.

5.1.1 Evrimsel Doğal Seçicilik ve Evrimsel Hesaplama

Bir toplumdaki bütün bireyler birbirlerinden farklıdır. Her bireyin kendi fiziksel ve davranışsal tapısı kendi gen yapısında saklıdır. Her bir birey, türünün güçlü yapısını, güçlü genlerini bir sonraki nesile aktarır. Böylece bireyleri zamanla gelişen bir toplum oluşur. Doğanın yapısı gereği, güçlü olan ve değişikliklere çabuk uyum sağlayan bireyler daha uzun yaşarlar. Bu hayatta kalma mücadelesi, doğal seçme yöntemi olarak tanımlanır. Genetik algoritmada, hayatta kalmak için uyum sağlayan ve güçlü yönlerini bir sonraki nesile aktaran bireyler üzerine kurulmuştur.

Seçici adaptasyon doğal seçme yöntemlerinden türetildiği halde karmaşık problemleri çözmekte çok güçlü ve etkilidir. Özellikle sadece bir tane en iyi çözümü olmayan veya en iyi çözümün devamlı olarak değiştiği problemlerde bu yöntem kullanılır. Seçici adaptasyon bir problemi çözmek için, önce bir toplumluk aday çözüm kümesi oluşturur. Bunlar içlerinde çeşitli bilgiler gömülü kromozomlardır. Bir kromozom, genler dizisidir. Bir kromozomun bir strateji belirlemesi için, problemin çözümünde karşılaşılabilecek her durumda izlenecek yol için bir reçeteye ihtiyacı vardır. Genler, verilen herhangi bir senaryo için, bu senaryoyu işlerlerse nelerin yapılacağını anlatan bir değer içerirler. Böylece bir genin ihtiva ettiği değerler kümesi, genin uygulayacağı hareketler kümesine karşılık gelir. Seçici adaptasyon bir problemi deneme yoluyla

çözmeye çalışır. Her denemede, bütün stratejiler problemi çözmeye çalışır ve daha başarılı stratejiler üreme yoluyla birleşirler ve ortaya çıkan yeni nesil stratejiler, kendilerini oluşturan stratejilerin parçalarını taşırlar. Bu işlem daha başarılı stratejilerin bilgilerinin nesiller boyunca var olmasını, başarısız stratejilerin ise giderek yok olmasına sağlar. Bu mekanizma, problem çözmede çok güçlü ve çok basit bir yöntemdir. Seçici adaptasyon vasıtasıyla problem çözmede tek gereken, çözümün kalitesinin bir ölçümüdür. Bu verilen stratejinin problemi ne kadar iyi çözdüğünün ölçüsüdür (Özdemir, 2006).

Son yıllarda evrim temeline dayalı algoritmalara ilgili oldukça artmıştır. Bu temel ile oluşturulmuş hesaplamalara genel olarak evrimsel hesaplama denmektedir. Bir problemi evrimsel algoritma aşağıdaki elemanlara ihtiyaç duymaktadır:

a. Çözümlerin genetik temsili

b. Başlangıç toplumu oluşturulması

c. Çevre, yani toplumu değerlendirecek uygunluk fonksiyonu

d. Popülasyonun genetiğini değiştirecek operatörler

e. Kontrol parametrelerinin değeri

5.1.2 Genetik Algoritmanın Tanımı

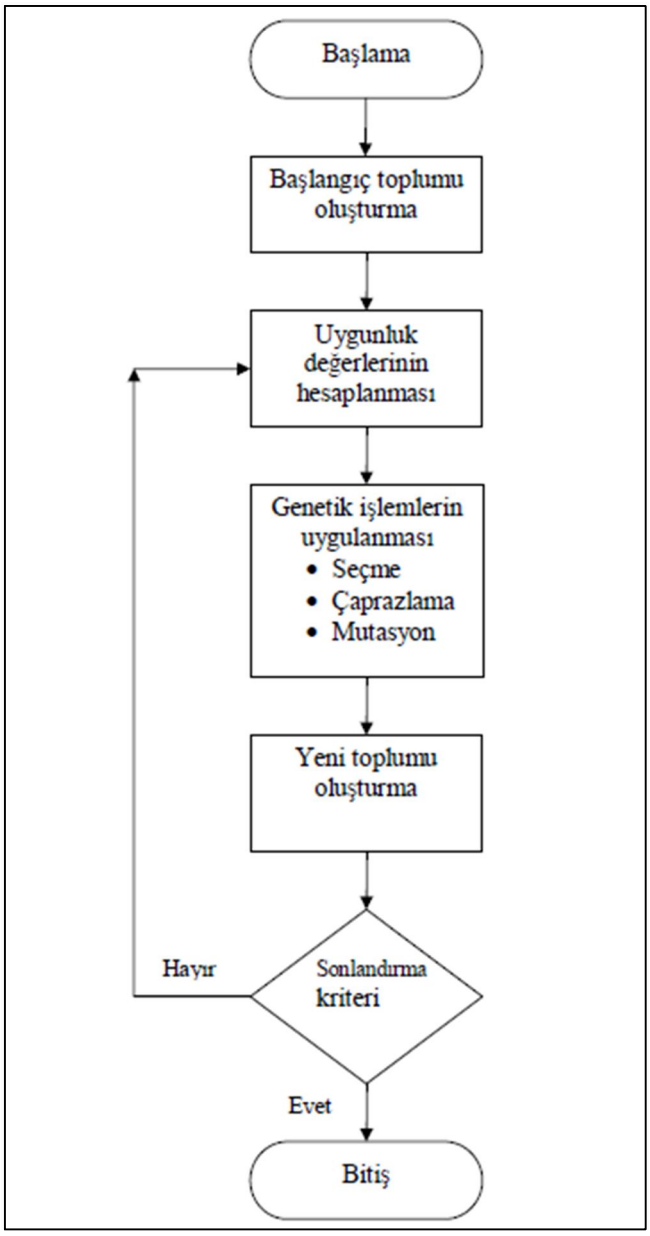
Genetik algoritma, Holland tarafından oluşturulmuştur (1975). Holland ve arkadaşları bu hesaplama yöntemini geliştirmiştir, sonrasında kitap olarak yayınlamıştır (Holland, 1975).

Genetik algoritma, yönlendirilmiş rasgele araştırma algoritmalarının bir türüdür. Genetik Algoritma, her yinelemede daha iyi çözümler elde edebilmek için sürekli gelişen nesilleri taklit eden bir yapay zeka sistemidir. Bu evrimsel işlemler çaprazlama, mutasyon ve seçme adımlarını içerir. Genetik Algoritma bir problemi çözerken çaprazlama operatörünü kullanarak iki ya da daha fazla bireyin özellikleri birleştirilerek, daha iyi çözüm sağlayan bireyler oluşturulabilir. Mutasyon değişik adımların rastgele sıralanması ve birleştirilmesi işlemini sağlar. Son olarak seçme işlemi ile daha iyi sonuç

veren bireyler bir sonraki nesile aktarılırken, diğer bireyler toplumdan yani çözüm havuzundan çıkartılır. Sona kalan toplumun uygunluk değeri ve kalitesi yüksek bireyi, hesaplama içerisindeki en iyi çözüm veya en iyi çözüme yakın bir çözüm olacaktır.

Genetik Algoritmalar binlerce veya milyonlarca girdisi bulunan büyük kapsamlı ve karmaşık problemlerde iyi sonuçlar sağlar. Bir çok karar verme veya karar değiştirme aşamasında genetik algoritma kullanılarak sayısız çözümler araştırılarak en uygun çözüm aranır. Genetik Algoritma tam olarak belli bir çözüm yolu olmayan veya olsa bile bunun çok zaman aldığı problemlerde kullanılır. Bunlar yerel olarak çoklu ve karmaşık, hatta çelişen kısıtlamaları olan ve bu kısıtlamaların hepsine aynı anda uyulması gereken problemler olarak karakterize edilebilir (Engin, et al., 1996). Bu problemlere örnek olarak iş gücü planlaması, dağıtım şebekeleri, ambar ve depolar için en uygun yerlerin bulunması, istatistiki modellemelerin yapılması vb. gösterilebilir.

Şekil 5.1: GA akışı



Kaynak: (Paksoy, 2007)

Genetik Algoritmalarda ilk olarak rasgele bir başlangıç toplumu üretir. İlk toplumdaki bireyler rasgele oluşturulduğu için uygunluk değerleri yüksek olmayabilir. Daha sonra bu bireyler türlerin evrimi teorisine göre çeşitli işlemlerden geçirilirler. Bu evrim teorisini taklit eden mekanizma, toplumdaki çözüm metotlarını genetik kromozom zincirleri olarak kabul edip bunları birbirleriyle eşleştirerek (çaprazlama) ve genleriyle oynayarak (mutasyon) yeni kromozom zincirleri elde eder (Kandemir, et al., 1996).

Üretilen yeni nesillerden bazıları, tamamen rastgele üretilmiş olan ilk toplumdakilerden daha iyi çözümler üretirler. En iyi çözümler toplumda kalır ve diğerleri atılır (seçme). Bu işlem toplumdaki bireylere uygulanmaya devam edildikçe, çözümler devamlı olarak iyileşir. Üretilen en iyi çözümleri tutup, kötüleri atarak toplumu iyileştirme işlemi sağlanır.

5.1.3 Temel Kavramlar

Kromozom, GA problemin çözüm alternatiflerini içeren çözüm kümesidir ve simgelediği çözümün adımlarından oluşur. Kromozom üzerindeki işlem adımları, bir dizi yapısı oluşturularak, her bir dizi adımı bir işleme karşılık gelecek şekilde düzenlenir. Kromozomlar, işlem adımına karşılık gelen genlerin, oluşturulan dizi içerisinde sıralanarak bir çözüm oluşturmasıyla ortaya çıkar.

Gen, kalıtımdaki en küçük öğedir. Bireyin genetik özelliklerini üzerinde barındıran parçadır. Her bir işlem, bir geni, her bir gen proje içerisindeki sayısal bir karar mekanizmasını işaret eder.

Toplum büyüklüğü, kromozom ya da birey sayısını ifade eder. Toplum küçük olursa, GA daha küçük bir çözüm uzayında arama yapacaktır. Bu durumda çözüm uzayı, en iyi çözümden uzak olabilir. Toplum büyük olursa, GA sonuçları geç getirecek ve yavaş çalışacaktır.

5.1.4 Bireyler ve Uygunluk

Genetik algoritmanın uygulanmasının ilk aşaması başlangıç toplumunun oluşturulmasıdır. Başlangıç toplumu, başlangıç kromozomlarını, başlangıç kromozomları da başlangıç genlerini içerir. Bilgilerin anlam bütünlüğü için, genlerin kodlaması sistematik şekilde yapılmalıdır. GA ile ilgili yapılan ilk uygulamalarda, kromozomlar ikilik sayı tabanına göre oluşturulmuşlardır. İkili tabanda oluşturulan her bit, bir çözümü(geni) temsil eder. Kromozomların kodlanmasında ikili sayı sistemine ek olarak tam sayı ve reel sayılarda kullanılmaktadır. İkili sistemin yetersiz kaldığı girdisi fazla olan büyük uygulamalarda onluk, on altılık sistemler kullanılabilir.

Toplumdaki kaliteli bireylerin sonraki nesillere aktarılması, önceden oluşturulan bireyin uygunluk durumu kontrolüyle yapılır. Bireylerin uygunluk seviyesi, önceden hazırlanan ve değerlendirmelerin yapılarak kalitenin ölçüldüğü uygunluk fonksiyonu ile hesaplanır. Uygunluk fonksiyonu değeri iyi olan çözümler kaliteli ve en iyi çözüme yakın olan kromozomlardır. Çalışmanın en başında belirlenecek uygunluk fonksiyonu ile toplumdaki bütün bireylerin uygunluk değeri hesaplanır. Uygunluk değeri en yüksek bireylerin seçilerek sonraki nesillere aktarılması GA' nın gücünün temelidir.

5.1.5 Uygun Bireyleri Seçme

Seçme işlemi, uygunluk değeri yüksek olan bireylerin yaşaması prensibine göre işler. İlk nesilde bulunan çözümlerden en uygun olan kromozomların bazıları seçilerek bir sonraki nesile aktarılır. Yeni oluşturulan nesilde bir önceki nesilden gelen kromozomlar haricindeki diğer kromozomlar, genetik uygulamalar kullanılarak yaratılır. Uygunluk fonksiyonu ile ölçüldüğünde, değeri yüksek çıkan kromozomların, bir sonraki nesile seçilerek aktarılma olasılıkları yüksek olacaktır. Bu özellik sayesinde, yeni oluşacak nesile kötü bireylerin taşınması önlenirken iyi bireylerin taşınması ve bu bireylerden başka bireyler üretilmesi en iyi çözüme hızla yaklaşmayı sağlayacaktır.

Uygunluk fonksiyonu ile ölçümü sonrası kalitesi iyi çıkan kromozom, bir sonraki nesilde yeni bireylerin oluşumunda kullanılabilir. Bu yöntem ile kromozomlar üzerindeki iyi özellikler nesilden nesile aktarılabilir.

Bir sonraki nesile aktarılacak kromozomların seçimi için genel olarak *elitist*, turnuva yöntemi, rulet çarkı kullanılır.

Elitist yöntemde uygunluk fonksiyonu sonucu kalitesi en yüksek çıkan kromozom, en düşük kromozomu ezerek yerini alır. Böylece en kötü bireyler toplumdan çıkartılırken en uygun birey bir sonraki kuşağa aktarılır.

Rulet çarkında toplumdaki bütün kromozomların kalitesi belirlenir. Kalitesi en yüksek olan kromozom, seçilme olasılığı en yüksek olan kromozomdur. Bu olasılıklar referans alınarak bütün kromozomlar olasılık değeri büyüklüğünce rulet çemberine yerleştirilirler. En yüksek olasılıklı bireyler rulet çarkında en geniş alanı kaplarlar ve

seçilme olasılıkları yükselir. Yeni toplumun birey sayısı tamamlanıncaya kadar bu çark üzerinden olasılıklar kapsamında yeni birey seçimi yapılır.

En yaygın uygun birey seçimi yöntemi kolay kullanımı nedeniyle turnuva yöntemidir. Bu yöntemde, önceden belirlenen kriterlere göre nesil içerisinden belirlenen kromozomlar bir turnuvaya tabi tutulur. Turnuvada uygunluk değeri ve kalitesi iyi olan birey sonraki nesile aktarılır. Böylece oluşturulan yeni toplum, bir önceki toplumun kötü bireylerinden arındırılmaya çalışılmaktadır. Optimum çözüme hızlı erişim sağlaması açısından önemli bir yöntemdir (Paksoy & Uzun, 2008).

5.1.6 Çaprazlama

Çaprazlama operatörü, doğal ortamda meydana gelen melez yapıların üretilmesine eş değer bir özelliği genetik algoritmaya kazandırır. Seçme işlemleri ile bir sonraki topluma aktarılması için bulunan uygun kromozomların birer çifti rasgele seçilir ve çaprazlama operatörü bu iki yapıdan yeni iki yapı meydana getirmek için kullanılır. Bu genetik şifre değiştirme algoritmasında üç değişik çaprazlama değişkeni uygulanmaktadır.

a. Bir nokta çaprazlaması, rasgele seçilen bir tam sayının seçilen birinci ve ikinci kromozom için hangi gene denk geldiği bulunur. Bu iki kromozom arasındaki genler, belirtilen tam sayıya denk gelen genden itibaren değiştirilerek yeni çocuklar üretilir.

b. İki nokta çaprazlaması, rasgele seçilen iki tam sayının birinci ve ikinci kromozom için hangi gene denk geldikleri bulunur. Bu iki kromozom arasındaki birinci ve ikinci tam sayının aralığına düşen genler, karşılıklı olarak yer değiştirilerek yeni çocuklar üretilir.

c. Düzgün çaprazlama, 1 veya 0 olması koşuluyla rasgele bir sayı üretilir ve bu 1 ise birinci kromozomdan, 0 ise ikinci kromozomdan gen alınarak yeni çocuklar üretilir. Bu uygulama iki defa yapılırsa, iki nokta çaprazlamayı kapsar.

5.1.7 Mutasyon

Mutasyon operatörü, doğadaki mutasyonun benzerini yaparak genetik algoritmada çeşitliliği sağlar ve doğru sonuçlara ulaşmada büyük görev sahibidir. Yeni toplumda bulunan çözümlere ait her bit tek tek kontrol edilir ve mutasyon oranına göre 1 ise 0'a, 0'sa 1'e çevrilir, yani bitlerin değeri tersine döndürülür. Mutasyon operatörü olmayan bir genetik algoritma, bireylerin tekrar üreme ve çaprazlama işlemlerinde değişmediği varsayılırsa en iyi çözümü ancak çözüm bireylerinin başlangıç toplumunda olması halinde bulabilir. Mutasyon yeni, görülmemiş ve araştırılmamış çözüm elemanlarının bulunmasını sağlar. Mutasyon işlemiyle, genetik algoritmanın yerel iyi çözümlere takılması önlenerek genel iyi çözüm aranması sağlanır. Çünkü mutasyon, daha önceden atılmış iyi çözüm elemanlarının tekrar üretilmesini sağlar.

Toplumlar arasında araştırma yapılırken gelişme sağlanamadığı durumlarda mutasyon sabit tutulmayıp, adaptif olarak değiştirilebilir. Bu değişiklikler, belirli bir adımda gelişim sağlanamaması, mutasyon oranının kromozom uygunluk değerine göre dinamik adaptasyonu, kromozomun farklı kısımları için farklı mutasyon olasılıklarının uygulanması şeklinde olabilir. Optimizasyon işleminin başlangıcında genellikle toplumda çok az sayıda iyi çözüm mevcuttur. Bundan dolayı başlangıçta yüksek mutasyon oranının araştırmayı hızlandırmak için kullanılması ve yavaş yavaş azaltılması daha uygun olacaktır.

5.1.8 Kontrol Parametreleri

Genetik algoritmanın girdisi olan ve kullanıcı tarafından seçilen kontrol parametrelerinin değerlerinin seçimi algoritmanın performansı üzerinde büyük öneme sahiptir. Basit bir genetik algoritmanın temel kontrol parametreleri şunlardır:

a. Toplum büyüklüğünün küçük veya büyük olması algoritmanın performansını etkiler. Toplumun küçük olması, araştırma uzayının küçülmesine neden olacaktır ve araştırma en iyi çözüm yerine yerel iyi çözüme yönelecektir. Toplum büyük seçilirse, bir toplumu araştırmak uzun zaman alacağı için algoritma süresi uzayacaktır. Optimizasyon yavaş çalışacak, geç sonuç getirecektir.

b. Çaprazlama oranının düşük seçilmesi durumunda yeni kuşağa çok az sayıda yeni yapının girmesine neden olunur. Araştırmanın yakınsama hızı düşmektedir. Yüksek çaprazlama oranı, hızlı bir şekilde çözüm uzayındaki çözümlerin incelenmesine neden olur. Oran çok yüksek olduğunda benzer veya daha iyi yapılar üretilemeden yapılar çok hızlı olarak bozulabilir.

c. Yüksek mutasyon değeri, algoritmada çok yüksek rasgeleliğe neden olacak ve araştırmayı çok hızlı tahrip edecektir. Çok düşük mutasyon oranının kullanılması, araştırma uzayının tamamen araştırılmasını engelleyecektir ve algoritmanın yerel iyi çözüm bulmasına neden olacaktır.

De Jong (1975), genetik algoritmanın performansı üzerine kontrol parametrelerinin etkisini incelemek amacıyla çeşitli test problemleri kullanarak çalışmalar yapmış ve bu çalışmalar sonucunda iyi bir gerçek zaman ve gerçek olmayan zaman performansı elde etmek için kontrol parametrelerine uygun değerler önermiştir. Benzer olarak, Schaffer ve arkadaşları (1989) GA' nın performansı üzerinde kontrol parametrelerinin etkisini incelemek amacıyla ayrıntılı bir çalışma yapmış ve iyi bir gerçek zaman performansı elde edilmesi için gerekli olan kontrol parametre değerlerini araştırmıştır. Grefenstette (1986), genetik algoritmanın kontrol parametrelerinin en uygun değer olarak ayarlanması için ikinci bir genetik algoritmanın kullanılmasını araştırmıştır.

Tablo 5.1: GA için önerilen kontrol parametre değerleri

Kontrol Parametreleri	De Jong	Schaffer	Grefenstette
Popülasyon büyüklüğü	50 - 100	20 - 30	30
Çaprazlama oranı	0,6	0,75 - 0,95	0,95
Mutasyon oranı	0,001	0,005 - 0,01	0,01

Kaynak: (Karaboğa, 2011)

İkili kodlu kromozomların kullanıldığı durumlarda en iyi toplum büyüklüyle ilgili teorik araştırma Goldberg (1985) tarafından yapılmıştır ve toplum büyüklüğü ile kromozom uzunluğu arasındaki bağıntı aşağıda verilen denklem 6.1 ile tanımlanmıştır:

$$\text{toplum büyüklüğü} = 1{,}65 \times 2^{0{,}2 \times \text{uzunluk}} \quad \textbf{(5.1)}$$

5.2 ISIL İŞLEM ALGORİTMASI

Isıl işlem algoritması (İİA), aynı zamanda tavlama benzetimi olarak isimlendirilir. Kimyada kullanılan ısıtma ve soğutma işlemleri ile faz değişimlerinden faydalanmıştır.

5.2.1 Doğal Isıl İşlem

Metal malzemelerin belirli bir sıcaklığa kadar sıcaklığının artırılması ve belirli bir süre bu sıcaklıkta kaldıktan sonra soğutma işlemine tabi tutulmasıyla ısıl işlem uygulanmış olur. Isıl işlem ile metallerin mekanik özellikleri artırılır.

Isıtma işlemi, malzemenin sıcaklığını oda sıcaklığından ısıl işlemde belirtilen sıcaklık değerine yükseltme işlemidir. Isıl işlemde ulaşılması öngörülen en yüksek sıcaklık, ısıtma sıcaklığı olarak adlandırılan işlem sıcaklığıdır. Ancak ısıl işlemin çeşidine göre bu kavram yerine, tavlama sıcaklığı da kullanılabilir.

Isıl işlemin son aşaması soğutmadır. Soğutma açık havada kendi haline bırakarak veya belirli bir sıcaklıkta hava verilerek yapılabilir. Belirli bir sıcaklığa kadar sürekli soğutup, bu sıcaklıkta sıcaklık dengesi sağlanacak kadar beklendikten sonra soğutmaya devam edilip kesintili soğutma sağlanabilir. Başka bir uygulamada yüksek sıcaklıktan belirli bir sıcaklığa ısıtılmış banyo içerisine daldırıp, sonra bu sıcaklıkta yapısal dönüşme yaptıracak şekilde bekleme ve tekrar sürekli soğutmaya devam etme olarak eş ısıl dönüşümlü soğutma tarzında olabilir. Çeliklerin sertleştirilmesinde yapılan hızlı soğutma işlemi de, ani soğutma olarak tanımlanır. Soğutma işlemi, ısıl işleme en uygun şekilde seçilmelidir.

5.2.2 Yapay Isıl İşlem Algoritması Tanımı

Yapay ısıl işlem algoritması temelini yukarıda bahsedilen temel prensiplerden almaktadır. Bu algoritmanın optimizasyon problemlerine uygulaması ile ilgili çalışmalar, 1983 yılında Kirkpatrick ve arkadaşları tarafından yapılan bir çalışma ile başladı (Kirkpatric, et al., 1983). Isıl işlem belli bir sıcaklık derecesine kadar katının sıcaklığının arttırılması ve sonrasında bu sıcaklıktan azaltılması işlemini tanımlar. Çıkartıldığı yüksek sıcaklık değerinde katı molekülleri, sıvı moleküllerine dönüşür ve

moleküller rasgele, düzensiz olarak yerleşir. Sonra sıvı haldeki moleküller kristal bir yapıya dönüşene kadar yani kararlı hale gelinceye kadar sıcaklık düşürülüp soğutma işlemi uygulanır. Sıcaklığın düşürülmesi doğru şekilde yapılırsa kristal yapı çok düzenli olur, yani süper kafes yapı elde edilir. Soğutma işlemi, dış sıcaklığı ani olarak sıfıra düşürmek suretiyle yapılırsa kristal yapıda geniş dağılımlı düzensizlikler ve yapısal bozukluklar meydana gelir. Yani kafes yapısında düzensizlikler oluşur ve süper kafes meydana gelmez. Bu olay, hızlı soğutma olarak tanımlanır.

Yapay ısıl işlem algoritmasında optimizasyon probleminin olası çözümleri, katının belli bir enerji seviyesinde moleküllerin hız, pozisyon, oryantasyon gibi özellikleri tarafından belirlenen durumlarına, söz konusu çözümler için hedef fonksiyonunun aldığı değerler ise enerji seviyelerine karşılık gelmektedir. Atomların mümkün olan en düşük enerji seviyesine ulaşması, problemin global optimum noktasının bulunması anlamına gelmektedir (Şimşek & Mergen, tarih yok).

Dış sıcaklık sıfır olduğunda daha yüksek enerjili bir duruma geçiş mümkün olmaz. Bu durumda araştırma bölgesel minimuma takılır. Bu yüzden sıcaklık yavaş ve dikkatlice düşürülür ve her sıcaklık azaltılmasında dengeye erişmek için bir süre sıcaklık sabit tutulur. Sıcaklık sıfır olmadığı durumlarda yukarıya doğru bir hareket olabilir. Mevcut enerji seviyesinden uzaklaşmayı sağlamak için mevcut sıcaklığı belirli bir süre muhafaza ederek bu adım adım soğutma işlemi tekrarlanır. Böylece yerel minimum değerden kurtularak tüm uzayda minimum değer araması yapılır. Yukarı doğru olan bu hareketlerin kabulü belirli bir olasılık tabanlı kontrol stratejisi ile takip edilir. Bu kontrol işlemini tanımlamada Kirkpatrick ve arkadaşlarına ilham kaynağı olan fikir, Metropolis ve arkadaşlarının istatiksel termodinamikle ilgili yapmış olduğu bir çalışmadan çıkmıştır (Metropolis, et al., 1953). Termodinamik kanunları, T sıcaklığında enerjide AE genlikli bir artışın olma olasılığını

$$P(AE) = \exp(-AE/kT) \quad \textbf{(5.2)}$$

İfadesi ile tanımlamaktadır; burada k Boltzman sabiti olarak bilinen bir fiziksel sabittir. T sıcaklığının azaltılması bir bozulma oluşturmakta ve bu bozulmadan dolayı ortaya enerji çıkmaktadır. Bu enerji hesaplanarak azalma olmuşsa sistem bu yeni duruma kayar, artmışsa yeni durum yukarıda bulunan olasılığa göre kabul edilir. Uygulanan algoritma çözüme karşılık gelen bir dizi kristal durumlar üretmektedir. Kristalin mevcut

durumu (S) verildikten sonra rasgele seçilmiş bir molekülün yer değiştirmesiyle kristalin durumunda küçük bir bozulma sağlanır. Mevcut durum (S) ile yeni üretilmiş durumun (S') enerji seviyeleri arasındaki fark (AE) negatif ise yeni durum daha düşük enerji seviyesinde demektir ve S' yeni durum olarak kabul edilir. S' kabul edildikten sonra işleme bu durumdan devam edilir. AE >= 0 olduğunda rasgele bir sayı 0 ve 1 arasında üretilir ve bu sayı Denklem 6.2' de tanımlı olasılık değerinden küçük ise o zaman S' yeni durum olarak kabul edilir. Yoksa mevcut durum (S) yeni çözüm olarak saklanır. Bu kabul etme kuralı, Metropolis kriteri olarak adlandırılır (Karaboğa, 2011).

Algoritmanın adımlarını sırasıyla aşağıdaki şekilde ifade edebiliriz:

1. Rasgele ya da doğrudan seçilen bir çözüm kümesi, başlangıç kümesi S olarak atanır
2. Başlangıç çözümü en iyi çözüm (S*) olarak atanır, S*=S
3. Başlangıç çözümü için maliyet (amaç) fonksiyonu hesaplanır, S: C(S)
4. Başlangıç sıcaklığı T_0 belirlenir
5. Başlangıç sıcaklığı, güncel sıcaklık (T) değerine atanır, $T=T_0$
6. Durma koşulu sağlanamadı ise aşağıdaki adımlar sırasıyla uygulanır:
   a. Mevcut S çözümünde rasgele bir komşuluk aralığını S' olarak belirle
   b. S' için maliyet fonksiyonunu hesapla S': C(S')
   c. Bir önceki çözüm ile mevcut çözümün maliyet fonksiyonları farkı alınır fark = C(S') - C(S)
   d. fark <= 0 ise f. Adıma dön ve S=S' kabul et, C(S)<C(S*) den dolayı S*=S atanır
   e. Fark>0 ise 0 ve 1 aralığında değer üret ve Denklem 6.2' deki P(AE)>üretilen değer ise S=S' atanır ve a. Adıma dönülür
   f. T sıcaklığı azaltılır ve adım 6'ya dönülür
7. En iyi çizelge S* oluşturulur ve durulur

Herhangi bir bölgesel optimizasyon algoritması, komşuluğu rasgele örnekleme yaparak yeni çözümler üretmek ve bu çözümlerden kötü olanları bile Denklem 6.2 ile tanımlı kabul kriterine göre kabul etmek suretiyle ısıl işlem algoritmasına dönüştürülebilir. Isıl işlem algoritması, özel algoritma olmaktan daha ziyade bir yaklaşımdır. Bundan dolayı optimizasyon problemlerine uygulama aşamasında çeşitli seçeneklerle ilgili kararlar alınmalıdır. Bu seçenekler iki gruba ayrılabilir:

a. Probleme yani probleme özel seçenekler:

i. Tüm muhtemel çözümler uzayını tanımlayabilecek şekilde çözümler için uygun bir gösterim yönteminin seçilmesi, minimize edilecek amaç fonksiyonunun tanımlanması ve bir başlangıç çözümünün üretilmesi

ii. Komşu çözümlerin üretilmesini sağlayan komşu üretme mekanizmasının tanımlanması

iii. Komşuluğun nasıl seçileceğinin ve hareket seçimleri için uygun stratejilerin tanımlanması

b. Algoritmanın kendisine ait seçenekler:

i. Sıcaklık parametresi T için uygun bir başlangıç değerinin tayin edilmesi

ii. Sıcaklık değiştirme koşulunun oluşturulması ve soğutma oranı belirlenmesi

iii. Ulaşılan her bir sıcaklık değerinde uygulanacak tekrarlama miktarının belirlenmesi

iv. Araştırmanın durdurulması için durdurma kriterinin atanması

Isıl işlem algoritmasının performansı seçilecek soğutma tarifesiyle doğrudan orantılıdır.

5.3 KARINCA KOLONİ ALGORİTMASI

Bu alanda karıncaların doğadaki davranışlarından esinlenerek geliştirilen karınca kolonisi optimizasyonu aktarılacaktır. Karınca kolonilerinin yiyecek bulmak için doğada gösterdiği davranışlar, matematiksel olarak ifadesi ve uygulamaları aktarılacaktır.

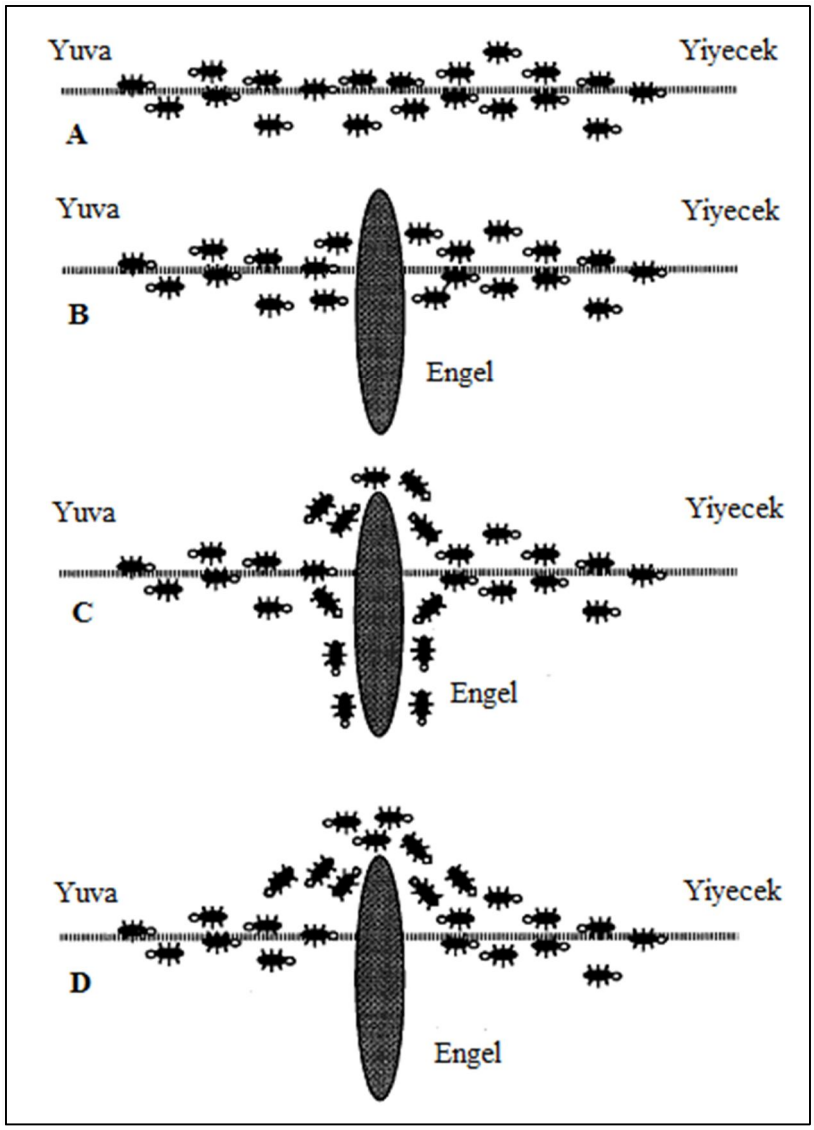
5.3.1 Doğal Karınca Davranışları

Tekil olarak bakıldığında karıncalar küçük ve karmaşık bir yapısı olmayan basit canlılardır. Tekil olarak basit varsayılan karıncalar topluluk (koloni) olarak güçlü, karmaşık problemleri çözebilecek yeteneğe sahip olmaktadırlar (Erdoğan, 2008). Bir karınca doğası gereği yuva yapmak, yiyecek bulmak gibi temel görevleri yerine getirebilir. Karıncalar yuva ile yiyecek arasındaki en kısa yolu çift taraflı olarak bulabilmektedirler. Bu özellik ezberlenmiş bir davranış değildir. Bunun anlamı, karıncaların kullandığı en kısa yol doğal nedenler veya fiziksel müdahale ile kullanılamaz duruma gelse bile, karıncalar yine yeni en kısa yolu bulacaktır. Karıncalar buldukları bu en kısa yol bilgisini diğer karıncalar ile paylaşır. Paylaşma işlemi bir iletişim formu ile yapılır. Bu iletişim formu, karıncalar tarafından izlenen yola bırakılan feromon sıvısıdır. Şekil 5.2 'de gösterilmiş yuva ve yiyecek arasında gidip gelen karıncaların davranışlarını inceleyelim.

Karıncalar, izledikleri yola belirli bir miktarda feromon maddesi bırakırlar. Aralarındaki iletişim bu feromon sıvısının yoğunlu ile sağlanır. Karıncanın izleceyeceği yönün belirlenmesinde bu maddenin fazla olduğu yolun tercih edilme olasılığı daha yüksektir. Olasılık dememizin nedeni, feromon maddesinin daha az yoğun olduğu yolun seçilme ihtimali de vardır. Bu yöntemi uygulayarak yuvaya veya yiyeceğe doğru giden karıncalar en kısa yolu hızlıca tespit ederler. Şekil 5.2 A'da belirtilen yol, varsayılanda karıncaların yiyecek için kullandığı yol olsun ve bu yolun dışarıdan bir müdahale ile bertaraf edildiğini düşünelim. Diğer bir deyişle, kullanılan en kısa yol bozulmuştur ve farklı bir kısa yol izlenmesi gereklidir. Bu durum şekilde 5.2 B' de gösterilmektedir.

Cisim yola konulduğunda alternatif yollar üzerinde feromon miktarı olmayacağı için karıncalar rasgele yönler belirleyerek o yönlere yönelirler. Koloni halindeki karıncaların bir kısmının sağ tarafa, benzer bir kısmının sol tarafa dönmesi beklenir. Bu durum şekil 5.2 C' de gösterilmektedir.

Şekil 5.2: Gerçek karıncaların en kısa yolu bulma aşamaları



Kaynak: (ATEŞ, 2012)

Karıncaların mevcutta kullandığı en kısa yolun üzerine konulan cismin etrafını dolaşan karıncalar, diğer yönlere yönelen karıncalara göre feromon yoğunluğunu kullandıkları yol üzerinde daha hızlı artırırlar. Bunun nedeni ise, karıncaların aynı hızla ilerlediği düşünülürse, kısa yol üzerinden geçecek olan karınca sayısı, uzun yol üzerinden geçecek olan karınca sayısına göre daha fazla olacağı için daha fazla feromon maddesi bırakırlar. Daha fazla feromon maddesinin bırakılması ise, daha fazla karıncanın bu yolu kullanmasını sağlayacaktır. Cismin karıncaların kullandığı yola konulması karıncaların yiyecek arayışındaki kısa yol kullanımını engellemeyecektir. Değişen ortam şartlarına karıncalar hızla uyum sağlayarak yine en kısa yolu bulurlar. Pozitif geri

besleme ile feromon maddesinin kısa yol üzerinde daha hızlı toplanmasını sağlayan, karıncaların yön seçerken daha yüksek feromon maddesine sahip yönleri tercih etmeleridir (Karaboğa, 2011).

5.3.2 Karınca Koloni Algoritması Tanımı

Karınca koloni algoritması (KKA) doğadaki karıncaların yiyecek arayışlarının matematik yöntemlerine uyarlanması ile geliştirilmiş bir algoritmadır. Bu algoritma ile ilgili ilk çalışmalar Dorigo ve arkadaşları tarafından (1991) yapılmıştır. Yapının adını "karınca sistemi", ürettikleri algoritmaya ise "karınca algoritması" olarak tanımlamışlardır. Karınca kolonilerinin yapısı tam olarak modellenmeyip yapay ajan karıncalar algoritma içerisinde üretilmiştir. Bu karıncalar topoloji üzerinde hareket ederler ve topoloji hakkında bilgileri toplarlar. Belirli bir hafızaya sahiptirler ve gerçek karıncaların aksine tamamen kör değildirler. Algoritmada karıncalar bireysel olarak hareket edebilirler. Herhangi bir topolojinin oluşturulmasında en önemli nokta karıncalar arasında iletişim sisteminin nasıl yapıldığının ifade edilmesidir.

Dorigo ve arkadaşları (1991), çok sayıda bölgesel olarak etkileşen basit bireylerin davranışlarının ortak özelliğini temel alan karınca algoritmasını zor problemlerin dağıtılmış çözümüne bir yaklaşım olarak tanıtmışlardır. Yani, bu yaklaşımda karıncalar basit etkileşen bireylerdir.

Düğümler arasında ilerleyen bir karınca aşağıdaki işlemleri gerçekleştirir:

i. Karınca, ilerleyeceği bir sonraki noktayı yani düğümü, noktanın uzaklığının olasılık değeri ve o yol üzerindeki feromon miktarı değerlerine göre belirler.

ii. Karıncanın sürekli aynı noktalar üzerinden geçmemesi için geçtiği düğümlerin bilgisi bir listede tutulur.

iii. Karınca bir noktadan başka bir noktaya ilerlerken kullandığı yola feromon miktarını da bırakır.

5.3.3 Modelleme ve Temel Parametreler

Yapay karıncalar, karınca koloni optimizasyonunu modellerken ilk gidilecek noktanın seçimi, feromon bırakma, feromon güncellemesi ve feromon buharlaşmasıyla ilgili bazı kurallara uymak zorundadırlar. Karınca topluluğu içerisinde bulunan tüm karıncalar bir çözüm yolunu temsil eder. Karınca koloni optimizasyonunda sonucun iyileştirilmesi için gerekli parametreler aşağıdaki gibidir:

a. Karınca sayısı, kolonide kan tane karıncanın olacağını belirleyen parametredir. Karınca sayısının arttırılması çözümde iyileşmeyi sağlarken, hesaplama artacağı için işlemin geç sonuçlanmasına neden olur.

b. Yineleme sayısı, arama işleminin kaç yineleme (adım) gerçekleşeceğini belirleyen parametredir.

c. Başlangıç feromon miktarı(T_0), mevcutta topolojide bulunan, algoritmanın başlangıç aşamasındaki feromon miktarlarını ifade eder.

d. Feromon kuvvetlendirme oranı(α), düğümler arasındaki feromon miktarının önem derecesini belirleyen parametredir. Bu değerin yüksek seçilmesi, feromun yoğun olduğu yolların seçilme olasılığını arttırmaktadır. Yüksek seçilmesi tesadüf oluşumu azaltır, önceki yinelemedeki sonuçları sonraki yinelemelere aktarmayı temin eder.

e. Sezgisellik kuvvetlendirme oranı(β), düğümler arasındaki mesafenin önem derecesini belirleyen parametredir. Bir sonraki noktanın seçimindeki etkisini belirtir. Bu değer arttıkça tesadüf oluşumlar artacaktır. Düşük seçilmesi ise alternatif çözümlerin araştırılması ihtimalini azaltır.

f. Feromon buharlaşma oranı(ρ), her yineleme sonunda düğümler arasındaki feromonların hangi oranda buharlaşacağını belirleyen parametredir. 0 ile 1 arasında bir değer alır.

g. Q_0 değeri en iyi çözümün sonraki yinelemelere aktarılması olasılığını belirleyen parametredir.

Karınca koloni algoritmasının adımları aşağıdaki gibidir;

1. Karınca sayısı yani bir toplulukta araştırılacak çözüm sayısı belirlenir.

2. Başlangıç feromon sıvısı değerleri verilir.

3. Karıncalar her farklı noktaya rasgele yerleştirilir.

4. Karıncalar, sonraki hedeflerine olasılık denklemlerine bağlı olacak şekilde ilerlerler.

5. Karıncaların üzerinden geçtiği yollar ve buna ait olan feromon sıvı miktarları hesaplanır, sonrasında yeni yerel feromon sıvı miktarları oluşturularak ilgili bilgi güncellenir.

6. En iyi çözüme ait yol hesaplanır ve *global* feromon güncellemesinde kullanılır.

7. Feromon buharlaşma değerine göre bütün yollardaki feromon değeri azaltılır.

8. Belirtilen nesil sayısına ulaşılana kadar veya hedef değere ulaşılana kadar adım 3'e gidilir.

KKA' da bir tur esnasında, i noktasında bulunan k karıncası için, sonraki j noktasını seçer iken iki alternatif yol söz konusudur. İlk alternatif, gidebileceği yollar içerisinden feromon miktarlarına bağlı olarak hesaplanan seçim değerlerinden maksimum olanını seçmesidir. Genellikle bu yolla tercih yapma olasılığı (q_0) yüzde 90 olarak belirlenmektedir. İkinci alternatifte ise yollardaki feromon miktarı göz önüne alınarak oluşturulan olasılık dağılımına bağlı olarak yollar seçilir. Aşağıda bu geçiş kuralına göre i noktasında bulunan k karıncasının u adet alternatif noktadan hangisine gideceğinin belirlendiği formül görülmektedir:

$$j = \max_{u \in J_k(i)} \left\{ \left[\tau(i,u) \right]^{\alpha} \times \left[\eta(i,u) \right]^{\beta} \right\} \quad \text{eğer} \quad q \le q_0 \qquad \textbf{(5.3)}$$

Burada τ (i,u) , (i,u) hattındaki feromon izidir. $\eta(i,u) = 1/\delta(i,u)$ i noktasından u noktasına uzaklığın tersidir. J_k (i) i noktasındaki k karıncası tarafından henüz gidilmemiş şehirleri temsil etmektedir. q_0 ($0<q_0< 1$) çözüm uzayını araştırmanın göreli önemliliğini gösteren parametredir.

$q<q_0$ olması durumunda ise ikinci geçiş kuralı uygulanır. Bu kurala göre gidilecek bir sonraki nokta hesaplanan seçim değerlerine bağlı olarak rasgele seçilmektedir. Dolayısıyla feromon miktarının daha yoğun olduğu yolların seçilme olasılığı daha fazla olacaktır. Gidilebilecek yolların seçilme olasılıkları aşağıdaki formüle göre hesaplanmaktadır:

$$p_k(i,j)=\begin{cases}\dfrac{\left[\tau(i,j)\right]^{\alpha}\times\left[\eta(i,j)\right]^{\beta}}{\sum_{u\in j_k(i)}\left[\tau(i,u)\right]^{\alpha}\times\left[\eta(i,u)\right]^{\beta}} & \text{eğer } j\in J_k(i)\\ 0 & \text{Diğer durumlarda}\end{cases} \quad \textbf{(5.4)}$$

Feromon güncellemesi, en iyi çözümünde içinde bulunduğu çözüm uzayının araştırılması için yapılır. Bütün karıncalar gidecekleri noktaya ulaştığında feromon güncellemesi toplu olarak yapılır. İki şekilde güncelleme yapılır:

a) Ağda bulunan bütün noktadan noktaya kenarların feromonlarının, algoritmaya girdi olarak verilen buharlaşma oranı kadar azaltılması yöntemiyle güncelleme yapılabilir.

b) Karıncaların geçiş yapmış oldukları yollardaki feromon miktarlarının, o yolu kullanan karıncanın yol uzunluğuyla ters orantılı olarak arttırılması (Stützle & Hoos, 2000).

Buharlaşma oranı daha önceki çözümlerin öneminin azaltılmasını sağlamaktadır. Yol uzunluğuyla ters orantılı olarak feromon artışı ise, iyi çözümlerin öneminin arttırılmasını temin eder (Dorigo & Gambardella, 1997).

KKA' nın farklı uygulamalarında farklı feromon yenileme kuralları kullanılmıştır. Karınca Koloni Algoritmasında feromon yenilemesi yerel ve *global* olmak üzere iki düzeyde gerçekleşmekte ve bir yoldaki toplam feromon düzeyi; lokal ve global feromon düzeyinin toplamından oluşmaktadır.

$\tau_{ij}(t)$, t yinelemesine kadar biriken feromon düzeyi, $\Delta\ \tau_{ij}^{k}(t+1)$ iterasyondaki feromon düzeyi ve ρ ($0<=\rho<=1$), feromon buharlaşma parametresi olmak üzere yerel feromon düzeyi aşağıdaki formülle hesaplanır:

$$\tau_{ij}(t+1) = (1-\rho)\,\tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t+1) \qquad \textbf{(5.5)}$$

$\Delta\,\tau_{ij}^{k}(t+1)$, aşağıdaki formülle hesaplanır:

$$\Delta\tau_{ij}^{k}(t+1) = \begin{cases} 1/L^{k}(t+1) & \text{k karıncası (i, j)} \\ & \text{yolunu kullanmışsa,} \\ 0 & \text{diğer durumlarda} \end{cases} \qquad \textbf{(5.6)}$$

$L^{k}(t+1)$, k karıncasının toplam tur uzunluğudur. Yerel feromon güncellemesi, turları dinamik olarak değişerek geçiş yapılan yolları cazip hale getirir. Karıncalar değişen feromon miktarlarına bağlı olarak her yinelemede turlarını da değiştirmektedirler. Böylelikle sürekli olarak daha kısa turları bulmak amaçlanmaktadır.

KKA'da, *global* feromon güncellemesi, geçerli yinelemedeki en iyi sonuca sahip karıncanın izlediği yolun feromon düzeyinin artırılmasından oluşur ve yinelemede bulunan en iyi sonuçların belli bir oranda ileriki yinelemelere aktarılmasını sağlar. *Global* feromon güncellemesi aşağıdaki formül ile yapılır:

$$\tau_{ij}(t+1) = (1-\rho)\,\tau_{ij}(t) + \Delta\tau_{ij}^{k}(t+1) \qquad \textbf{(5.7)}$$

$$\Delta\tau_{ij}(t+1) = \begin{cases} \dfrac{1}{L_{best}(t+1)} & (i, j) \text{ en iyi tura ait ise} \\ 0 & \text{diğer durumlarda} \end{cases} \qquad \textbf{(5.8)}$$

$L_{best}(t+1)$ geçerli yinelemede bulunan en iyi turun uzunluğudur.

6. VERİ VE YÖNTEM

Faaliyet sayısı çok fazla olan projelerin çizelgelemesinde matematiksel yöntemlerin performanslarının yeterli olmadığı yapılan bazı çalışmalarda vurgulanmış ve özellikle faaliyet sayısı fazla olan projelerde sezgisel veya ileri sezgisel yöntemlerin kullanılması gerekliliği ortaya çıkmıştır. Bu bağlamda, en yaygın proje yönetim metotlarından olan sarmal ve çevik proje yönetim metotlarının çizelgelemelerini, GA, İİA, KKA' sına göre belirli yineleme sayıları ile en iyi çözüme yakın çizelge üreten "Proje Programlama" adında bir paket program hazırlanmıştır. Microsoft Excel üzerinden alınan proje kaynak bilgileri, proje büyüklüğü ve diğer kısıtlar kapsamında c# .Net 3.5 ile tasarlanan uygulamanın kullanım kılavuzu EK-1' de verilmiştir.

6.1 VERİLERİN OLUŞTURULMASI

Optimizasyon işlemleri öncesi şirket çalışanlarının bilgilerinin ve proje adımlarının tutulduğu bir MS Excel veri tabanı oluşturulur. Oluşturulan veri tabanı özel bir tasarım olup rasgele bilgiler ile düzenlenmiştir. Bu veri tabanı oluşturulurken aşağıdaki adımlarda açıklanan şablonlar kullanılır:

6.1.1 Proje Uygulama Şablonu

Her bir proje ve optimizasyon algoritmasına özel veri tabanları oluşturulur. Bu veri tabanları her bir çalışanın bilgilerini, projenin adımlarını ve kısıtları içerir:

a. "resourceMembers" sayfasında, çizelgelemede kullanılabilecek mevcut insan kaynağı listesi bulunmaktadır. Her bir çalışanın sicil numarası, sorumluluğu, günlük çalışma ücreti, moral ve uzmanlığa bağlı verimlilik değeri bulunmaktadır. Verimlilik değeri 0 ile 1 aralısında bir değer almaktadır. Verimlilik değeri yüksek çalışanlar, bir işi diğer çalışanlara göre daha hızlı bitirebilmektedir. Bu sayfada bulunan bir bilgi de değer bilgisidir. Çizelgelemeyi oluşturacak proje yöneticisi

özellikle bazı çalışanlar ile çalışmak isteyebilir. Bu durumda çalışmak istediği personelin değer alanını 0 ile 1 alanında bir bilgi girmelidir. Bu değer arttıkça, bu personel çizelgede daha fazla yer alacaktır.

b. "projectSize" sayfasında proje çizelgesinin oluşturulacağı detay planlanmaktadır. Bu alanda bir gün dilimlere ayrılmaktadır. Proje yöneticisi bu dilimlerin sayısını ve her bir parçanın kaç saat olacağını belirleyebilir. Her bir parça, optimizasyon algoritmalarında kullanılacak bir düğüm teşkil edecektir. Her bir zaman diliminde çalışabilecek personel sayısı sarmal metotta 5 sabit, çevik metotta ise dinamik olarak "steps" sayfasında verilmektedir. Sarmal metotta 264 zaman dilimi, çevik metotta 25 zaman dilimi vardır. Sarmal proje yönetimi için zaman dilimi 2 saat, proje büyüklüğü 2.640 adam/saattir, çevik proje yönetimi için zaman dilimi 8 saat, proje büyüklüğü 1480 adam/saattir.

c. "steps" sayfasında projenin adımları ve yineleme durumları verilmiştir. Sarmal proje yönetiminde her bir faz, diğerinden ayrıdır ve konusunun uzmanı personeller kendi içerisinde birlikte çalışırlar. Aynı anda sarmal projede konusunun uzmanı 5 personel bulunur. Çevik proje yönetiminde ise, her bir yineleme için değer alanında yazan kadar personel, o hafta için birlikte çalışır. Değer alanını proje yöneticisi düzenleyebilir.

d. "valueOf" sayfasında, personelin özellikle çalışmak istediği veya istemediği zaman dilimleri, izinli olduğu zaman dilimleri, raporlu olduğu zaman dilimleri 0 ile (-1,1) arasında pozitif veya negatif girilebilir. Proje çizelgesi oluşurken bu değerler hesaplamaya katılır.

6.1.2 Varsayımlar

Optimizasyon algoritmalarına proje yönetim çizelgesini uyarlayabilmek için bazı kabuller yapılmıştır. Bu kabuller aşağıdaki gibidir:

a. Faaliyetler yarıda bırakılamaz, kesintiye uğramaz, başlanılan iş bitirilmek zorunadır.

b. Her faaliyetin başlangıç zamanı, belirli bir zaman dilimine yerleştirilmiştir, öne çekilemez, yer değiştirilemez.

c. Proje başlangıç ve bitişi sabit çizelgeleme yapılmaktadır.

d. Çalışanların verimliliğinin artması, projenin zamanına olumlu etki yapar.

6.2 OPTİMİZASYON ALGORİTMA YÖNTEMLERİ

Sezgisel optimizasyon algoritmalar probleme bağımlı algoritmalardır. Farklı problemlerde farklı sonuçlar ortaya çıkabilir. Bir problemde başarılı olurken diğer bir problem için başarı elde edilemeyebilir. Yapılan uygulamada bir çizelgenin uygun olma koşulu, o çizelgedeki faaliyetlere atanan çalışanların maaşlarının toplam gideri ile ters orantılı, çalışanların toplam verimlilik değeri ile doğru orantılıdır. Çalışanın verimliliği arttıkça, projedeki performansı artar ve projenin zamanını azaltır.

Sarmal proje yönetimi ve çevik proje yönetimi için yapılan optimizasyon çalışmalarında seçilen yöntemler her bir algoritmanın altında, aşağıda listelenmiştir:

6.2.1 Genetik Algoritma Optimizasyonu

Evrime dayalı oluşturulan GA adımlarında uygulama için seçilen metotlar aşağıdaki gibidir:

a. Her bir zaman dilimi, bir gen ile ifade edilmiştir. Buna göre her bir çizelge, bir kromozoma karşılık gelmektedir.

b. Gen kodlamada permütasyon kodlama kullanılmıştır. Her bir personelin sicil numarası gen kodu olarak dizilmiştir.

c. Toplum yineleme sayısı ve toplum büyüklüğü ara yüz üzerinden kullanıcıdan alınmaktadır.

d. Amaç fonksiyonu, maliyet, verimlilik, proje yöneticisinin seçimleri ve personel seçimlerini hesaplayacak şekilde oluşturulmuştur.

e. Seçim yöntemi olarak turnuva seçim yöntemi uygulanmıştır. Böylece gen çeşitliliği kısıtlanmadan, iyi bireyin yeni topluma aktarılması sağlanır.

f. İki noktalı çaprazlama kullanılmıştır. Çaprazlama olasılığı, uygulama ara yüzünden, kullanıcıdan alınmaktadır.

g. Kullanıcı ara yüzünden alınan mutasyon olasılığı ile çaprazlama sonraki genlerde mutasyon işlemi yapılır. Gen permütasyon kodlama olduğu için, aynı özellikte (aynı uzmanlığa sahip başka bir personel) başka bir gen ile değiştirilir. Mutasyon, algoritmada çeşitliliği sağlar.

6.2.2 Isıl İşlem Algoritması Optimizasyonu

Metal malzemelerin sıcaklık değişimine dayalı oluşturulan İİA adımlarında uygulama için seçilen metotlar aşağıdaki gibidir:

a. Her bir zaman dilimine bir personel atanarak zaman çizelgesi oluşturulur.

b. Yineleme sayısı, başlangıç sıcaklığı ve soğutma oranı ara yüz üzerinden kullanıcıdan alınır.

c. Soğutma kuralı olarak geometrik soğutma uygulanır. Yani soğutma oranı ile mevcut sıcaklık çarpılarak soğutulmuş yeni sıcaklık bulur.

d. Amaç fonksiyonu, maliyet, verimlilik, proje yöneticisinin seçimleri ve personel seçimlerini hesaplayacak şekilde oluşturulmuştur.

6.2.3 Karınca Koloni Optimizasyonu

Karıncaların yiyecek arayışına dayalı oluşturulan KKA adımlarında uygulama için seçilen metotlar aşağıdaki gibidir:

a. Her bir karıncanın izlediği yol bir çizelgeyi ifade eder.

b. Karınca sayısı, yineleme sayısı, feromon oranının önemi, sezgisellik değeri, başlangıç feromon değeri, buharlaşma oranı, feromon artış değeri ara yüz üzerinden kullanıcıdan alınır.

c. Her bir zaman dilimi bir durak noktasıdır ve başlangıçta karıncalar bu noktalara rasgele yerleştirilmiştir.

d. Karınca bir sonraki durağa gitmek istediğinde, uygulama o karıncanın gidebileceği durakları ve bu durakların uzaklıklarını hesaplamaktadır.

e. Karıncanın her durağında, bir sonraki yol haritası belirlenerek ışın araması şeklinde bir ağaç yapısıyla ilerlemesi sağlanmaktadır

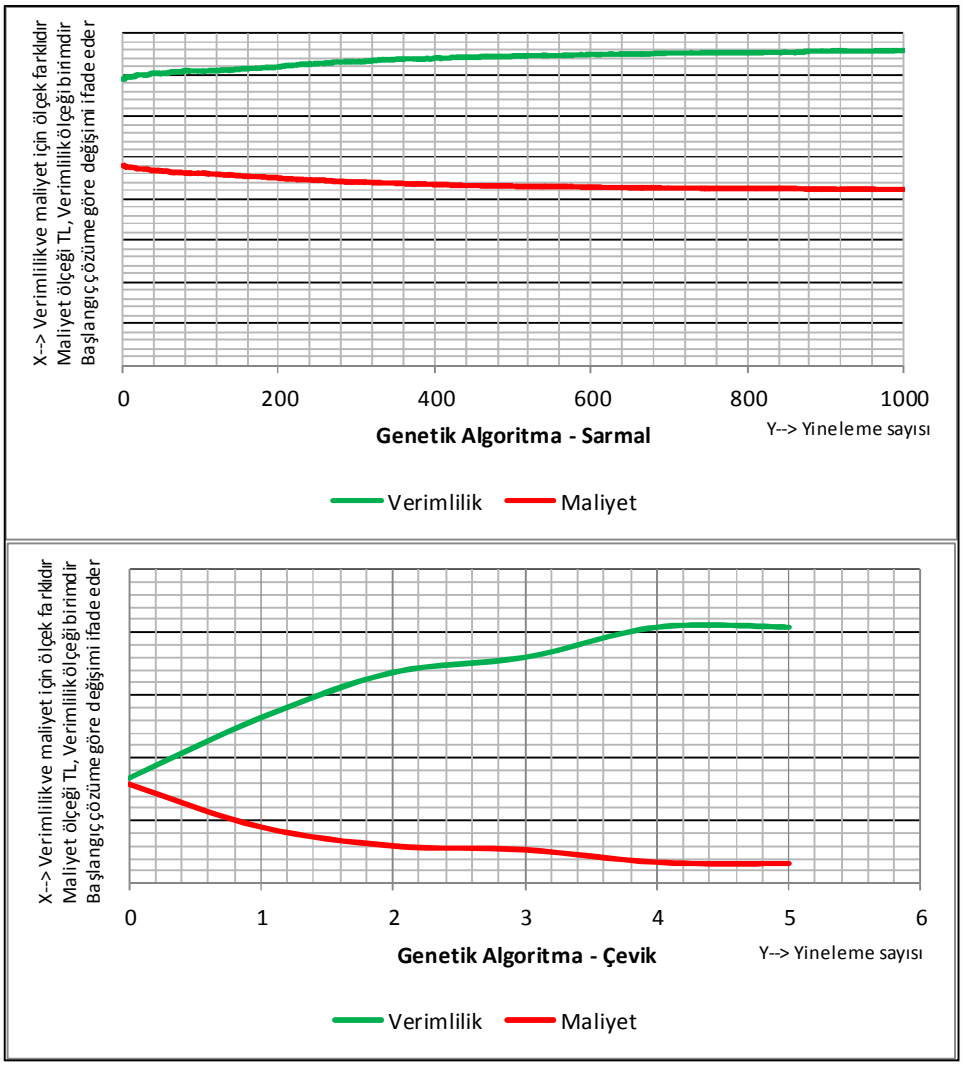
7. BULGULAR

Uygulama üzerinde deneyler intel i7 2.0GHz işlemci ve 8 GB belleği olan 64bit bir kişisel diz üstü bilgisayarda yapılmıştır. Programın kodlanması visual studio 2008 programı üzerinde c# .Net 3.5 *framework* ile yapılmış ve Windows 7 işletim sisteminde denenmiştir. Sarmal proje 2640 adam/saat, 1320 zaman dilimi ve 32 çalışandan oluşmaktadır. Çevik proje 1480 adam/saat, 25 zaman dilimi ve 32 çalışandan oluşmaktadır.

MS Excel ile bilgileri uygulama tarafından alınan sarmal ve çevik proje yönetim metotlarına göre çizelgeleme oluşturulmak için ayrı ayrı 5 kez çalıştırılan GA, İİA, KKA' a ait bulgular aşağıda paylaşılmıştır.

Genetik algoritma toplumdaki birey sayısı 30, toplum yineleme sayısı 1000, mutasyon olasılığı 0,01, çaprazlama olasılığı 0,9 olarak alınmıştır. Maliyet ve verimlilik dengede tutulduğunda, toplumun yenilenmesi sağlandıkça Şekil 7.1' deki değişim oluşmuştur.

Şekil 7.1: GA ile sarmal ve çevik yazılım geliştirme



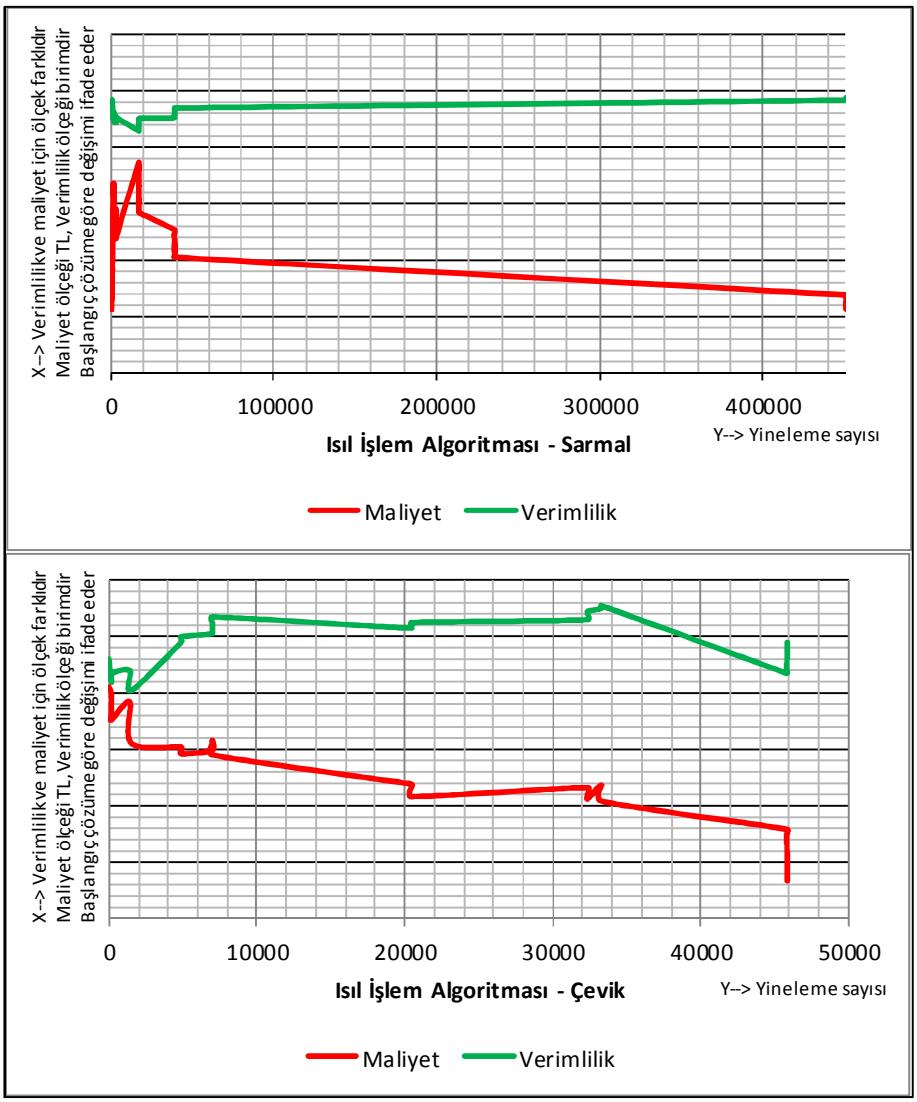
Kaynak: Bu şekil Mahmut Baş tarafından hazırlanmıştır (2014).

Genetik algoritma tek bir çözüm yerine eş zamanlı olarak toplum içerisindeki bütün çözümleri değerlendirir. Genetik algoritmadaki bu paralel arama özelliği, sonuçlara daha hızlı ulaşmayı sağlar. Sarmal yazılım geliştirme metodunda genetik algoritma uygulanması ile birlikte başlangıç çözümüne göre verimlilikte ortalama yüzde 4,5, maliyette ise ortalama yüzde 6,1 optimizasyon sağlanmıştır. Çevik yazılım geliştirme metodunda ise başlangıç çözümüne göre verimlilikte ortalama yüzde 2,4, maliyette ise ortalama yüzde 4,5 optimizasyon sağlanmıştır.

Isıl işlem algoritması 100.000 yineleme sayısı, mevcut sıcaklık 10.000, soğutma oranı 0,02 olarak alınmıştır. Daha öncesinde uygulamanın soğutma oranı daha büyük seçilmiş (0,995) ama yapının bozulup daha kötü çözüme yönelmesine neden olmuştur. Bu

yüzden soğutma oranı, ilk girilen değerlere göre azaltılmıştır. Şekil 7.2' de görüldüğü gibi yerel minimumlar elde edilmektedir. Yapılan denemelerde sarmal yazılım geliştirme metodu için başlangıç çözümüne göre maliyette en fazla yüzde 1,2, verimlilikte ise yüzde 0,7 optimizasyon sağlandığı görülmüştür. Çevik yazılım geliştirme modeli için ise başlangıç çözümüne göre maliyette yüzde 4,2, verimlilikte ise yüzde 2,4 optimizasyon sağladığı görülmüştür.

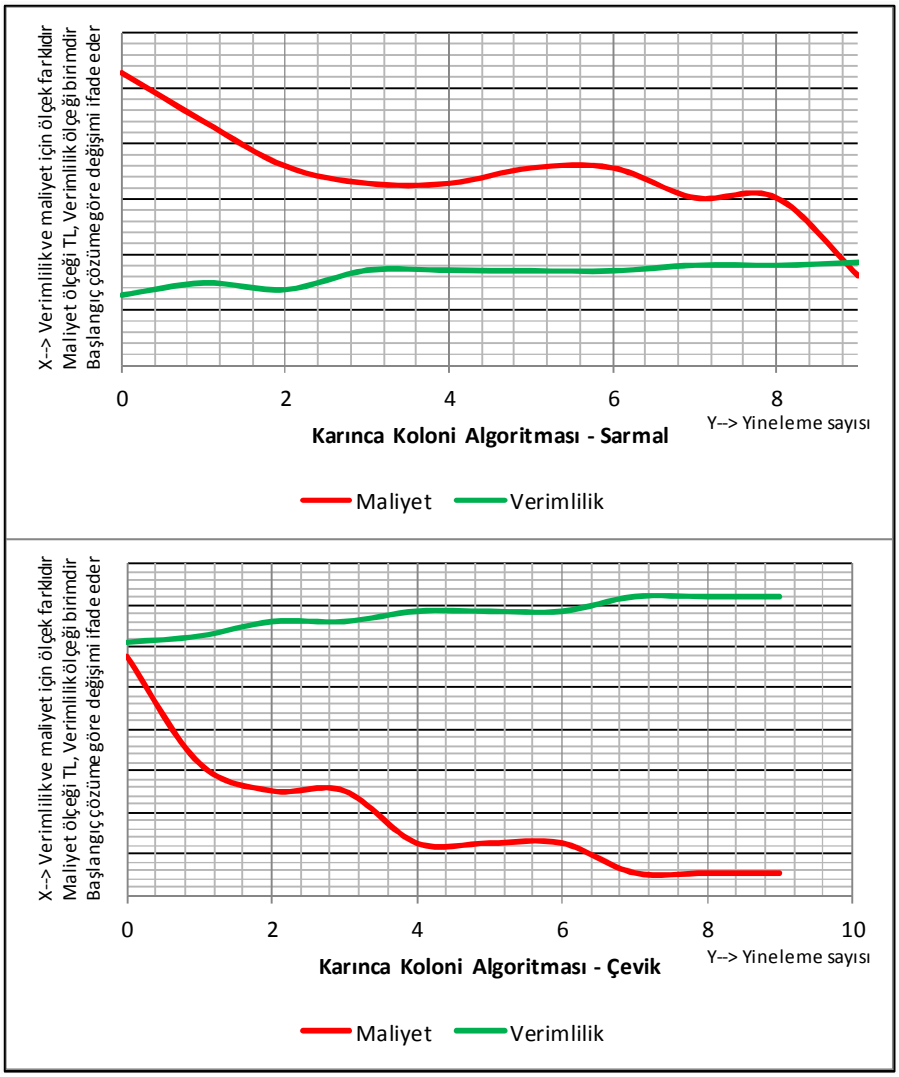
Şekil 7.2: İİA ile sarmal ve çevik yazılım geliştirme



Kaynak: Bu şekil Mahmut Baş tarafından hazırlanmıştır (2014).

Karınca koloni algoritması 10 yineleme, 10 karınca, 3 alpha, 2 beta, feromon buharlaşma katsayısı 0,01, sabit feromon artırma miktarı 2, başlangıç feromon miktarı 0,0001 olarak denemeler yapılmıştır. Çizelgeleme çalışmalarında KKA ağaç modeline göre tasarlanmaktadır. Her bir ağaç basamağında bir sonraki adımlar hesaplanmakta ve en uygun olan seçilmektedir. Şekil 7.3' de görüldüğü gibi maliyet hızlı bir şekilde azalmakta, verimlilik ise artmaktadır. Yapılan denemelerde sarmal yazılım geliştirme metodu için başlangıç çözümüne göre maliyette en fazla yüzde 1,1, verimlilikte ise yüzde 0,6 optimizasyon sağlandığı görülmüştür. Çevik yazılım geliştirme modeli için ise başlangıç çözümüne göre maliyette yüzde 2,1, verimlilikte ise yüzde 1,3 optimizasyon sağladığı görülmüştür.

Şekil 7.3: KKA ile sarmal ve çevik yazılım geliştirme



Kaynak: Bu şekil Mahmut Baş tarafından hazırlanmıştır (2014).

Spiral ve çevik yazılım geliştirme metotları için her bir optimizasyon algoritması 3 kez çalıştırılmış, en iyi değerler çözüm olarak kabul edilmiştir. Bu değerler ve ayrıca bu değerlerin ortalaması Tablo 7.1' de paylaşılmıştır. Buna göre sarmal projeler için zaman, maliyet ve verimlilik bakımından iyi değerler GA' da bulunmuştur, çevik projeler için ise en iyi değerler KKA' da bulunmuştur.

Tablo 7.1: Sarmal ve çevik yazılım geliştirme için optimizasyon algoritması karşılaştırmaları

		Verim/Maliyet = 20/80		Verim/Maliyet = 50/50		Verim/Maliyet = 80/20		ORTALAMA	
		Sarmal	Çevik	Sarmal	Çevik	Sarmal	Çevik	Sarmal	Çevik
GA	En İyi Maliyet*1	3.500.600	491.400	3.500.450	479.800	3.497.600	495.700	3.499.550	488.967
	En iyi Verim*2	1.247,70	173,00	1.248,34	175,50	1.250,34	174,30	1.248,80	174,27
	ÇS(dk)	2	1	2	1	2	1	2	1
İİA	En İyi Maliyet	3.718.700	502.200	3.727.300	503.300	3.731.200	502100	3.725.733	502.533
	En iyi Verim	1.196,41	170,50	1.196,71	169,80	1.196,91	171,7001	1.196,68	170,67
	ÇS(dk)	16	1	26	1	15	1	19	1
KKA	En İyi Maliyet	3.606.700	484.300	3.616.100	486.900	3.630.000	486600	3.617.600	485.933
	En iyi Verim	1.216,90	173,80	1.218,80	174,40	1.218,50	174,9	1.218,07	174,37
	ÇS(dk)	13	1	13	1	14	1	13	1

*1-Maliyet, proje çalışanlarının proje bütçesinden alacağı ücrettir. Aynı saat aralığı için her bir çalışanın maaşı, yani alacağı ücret farklı olabilir. Tabloda proje çalışanlarına verilen net ücret toplamı gösterilmiştir.
*2-Verim, projede çalışan kişiye, motivasyon, uzmanlık, proje konusu üzerindeki tecrübesine göre insan kaynakları ve çalışanın fonksiyonel yöneticisi tarafından verilen 0 ile 1 arasında puandır. Her yönden iyi olan çalışanlar 1 puandır. Verim ile zaman ters orantılıdır(Verim=1/Zaman). Tabloda projede çalışan personelin toplam verim puanı gösterilmiştir.

Kaynak: Bu tablo Mahmut Baş tarafından hazırlanmıştır (2014).

Sezgisel yöntemlerde başlangıç çözümü performansı etkileyen önemli maddelerden birisidir. İyi bir başlangıç çözümüyle başlanılması, kısa sürede daha iyi çözümlere ulaşmayı sağlar. Algoritma yineleme sayısı ne kadar az olursa, uzay o kadar az aranır, bu yüzden başlangıç çözümü ve komşu arama yapısı önemlidir.

Başlangıç çözümünün etkisi ortadan kaldırılarak bütün algoritmalar için aynı başlangıç çözümü ile algoritmalar tekrar çalıştırılmış ve algoritmaların büyük ve küçük projeler için iyi çözüm bulmadaki başarı sıralaması değişmemiştir. Başlangıç çözümü rasgele veya aynı başlangıç çözümü atanarak yapılan çalışmalarda; büyük projelerde genetik

algoritma ısıl işlem algoritmalarına göre yüzde 3 ile 6 arasında daha verimli, karınca koloni algoritmasına göre yüzde 2 ile 4 aralığında daha fazla fayda sağlamıştır, küçük projelerde ise karınca koloni algoritması genetik algoritmaya göre yüzde 0,5 ile yüzde 1 arasında, ısıl işlem algoritmasına göre ise yüzde 2 ile 3,5 arası daha fazla fayda sağlamıştır.

8. TARTIŞMA

Test uygulamasıyla yapılan testlerin Tablo 7.1' deki sonuçlarına bakıldığı zaman, GA sarmal yazılım geliştirme metodu gibi girdisi fazla ve büyük projeler için en verimli ve en hızlı çözüm olmuştur. Sonrasında KKA ve son olarak İİA gelmektedir. Çevik yazılım geliştirme yöntemi gibi daha az girdisi olan ve daha kısa sürede tamamlanan projeler için ise en verimli ve en hızlı çözümü sunan KKA algoritması olmuştur. KKA sonrasında, verimliliği ona yakın olan GA, en sonda İİA gelmektedir. Bu sonuçlara göre büyük projelerde GA, küçük projelerde KKA kullanılması önerilmektedir.

Yapılan çalışma ile birlikte, aktivite sayısı fazla, girdisi fazla olan projelerin GA ile optimizasyonu, girdisi ve aktivite sayısı az olan projelerde ise KKA ile optimizasyonun en verimli yöntemler olduğu desteklenmiştir. Bu uygulama bir proje yöneticisi, şirket sahibi, danışmanlık firması tarafından insan kaynağının planlanmasında rahatlıkla kullanılabilir. Daha önce bu yönde yapılmış çalışmalar için farklı bir bakış açısı katılarak, güncel yazılım projelerinde uygulama ile benzer sonuçlar elde edilmiştir.

Test uygulamasından ayrıca aşağıdaki sonuçlarda çıkartılmıştır:

i. Topluluk oluşturulan optimizasyon algoritmaları (KKA, GA) genel test sonuçlarında tek nokta üzerinden arama yapan yöntemlere (İİA) göre daha başarılı sonuçlar üretmiştir.

ii. Topluluk oluşturan algoritmalar aynı yineleme sayısında daha sık ve daha fazla çözüm üretirken, tek nokta arama yapan algoritmalar daha fazla aralıklı çözümler üretmektedirler.

iii. Her bir parametre ve uygunluk fonksiyonu algoritmanın çalışmasını doğrudan etkilemektedir.

Karaoğlan ve Altıparmak (2005) yaptığı uygulama ile orta ölçekli problemlerde ile ısıl işlem algoritması ile genetik algoritmayı çözüm kalitesi ve çalışma süresi açısından karşılaştırmıştır. Bu karşılaştırmada genetik algoritmanın yüzde 0,3 ile yüzde 5 arasında iyileşme sağlamıştır. Uygulamada 10x10 boyutundaki problemden başlayarak 20x20,

30x30 ve 40x40 boyutundaki problemleri incelemiştir. 10x10 boyutundaki problem için bütün algoritmalar en iyi eş sonuçları bulurken, problemin boyutu büyüdükçe, genetik algoritma en iyi sonuçlar üretmeye başlamıştır. 40x40 problem boyutunda en iyi çözümü genetik algoritma sağlamıştır. Buna karşın çalışma zamanı açısından uzun süre genetik algoritma çalışmıştır. Bunun nedeni genetik algoritmanın toplumdaki bütün bireylerin yani her bir yinelemedeki çözümlerin uygunluğunun yeniden hesaplamasıdır.

Kuzu ve arkadaşları (2014) yaptıkları araştırmada genel olarak toplum temelli oluşturulan optimizasyon algoritmalarının tek noktadan arama yapan algoritmalara göre daha iyi sonuçlar ortaya çıkardığını belirtmişlerdir. Bunun nedeni toplumsal algoritmaların birden fazla noktada çözüm araması olarak gösterilmiştir. Toplum temelli algoritmalar çözüm aralıkları daha az sapma ortaya çıkartacak şekilde dar ve sık bir seyir izlemektedirler. Dar aralıkta çözüm üreten bu algoritmaların iyi çözümlere daha çok ulaştığını gösterilmiştir. Tek nokta arama yapan meta sezgisel yöntemler ise daha fazla aralıklı çözümler üretmektedir.

Adewole ve çalışma arkadaşları (2012), genetik algoritma ve ısıl işlem algoritmasını gezgin satıcı problemine uygulayarak karşılaştırmıştır. Uygun parametrelerin girilmesi durumunda iki algoritmada iyi sonuçlar çıkartmıştır. Genetik algoritma şehir sayısının fazla olması durumunda daha güçlü davranırken, ısıl işlem algoritması daha az şehir sayısı olan problemlerde daha hızlı çözümler sunmuştur. Genetik algoritma, ısıl işlem algoritmasına göre daha yavaş çalışmaktadır.

Alhanjouri ve Alfarra (2013) yaptığı araştırmada 25 şehir ile gezgin satıcı problemini genetik algoritma ve karınca koloni algoritması ile çözmüştür. Yapılan çalışmada karınca koloni algoritması ile en kısa mesafeyi 4,6245 birim bulurken genetik algoritma ile en kısa mesafeyi 4,6149 bulmuştur. Genetik algoritmanın daha kısa mesafe bulmasının yanında kısa mesafenin bulunması için karınca koloni algoritmasına göre daha az zaman harcadığını belirtmiştir. Ayrıca, karınca koloni algoritmasının başlangıç parametrelerinin algoritmayı fazlasıyla etkilediğini ve başlangıç parametreleri seçme işleminin zorluğundan bahsetmiştir. Genel olarak gezgin satıcı problemlerinin araştırılmasında genetik algoritmanın, karınca koloni algoritmasına göre daha iyi olduğu sonucuna ulaşmıştır.

Genetik algoritma kodlama aşamasında adım adım işletildiği için modellemesi kolay, çalışması hızlı, sonuçları ise en iyi çözüme yakın çözümler üreten algoritma olarak gözlemlenmiştir. Mutasyon oranının seçilmesi ve uygunluk fonksiyonunun tasarlanması en önemli adımlar olmuştur. Isıl işlem algoritması hem aşağı hem yukarı hareket ettiği için soğutma değerine göre çok fazla değişkenlik göstermiştir. Yerel iyilere takılmadan en iyi çözümü aramıştır. Kod yazan için tasarımı kolaydır. Karınca koloni algoritması ise kaynak planlama ve zaman planlama uygulamalarında tasarım olarak zorlayıcıdır. Düğüm ve ağ yapısı düşünüldüğünde girdi sayısı arttıkça, düğüm sayısı artacak ve algoritmada karıncaların gidebileceği yön sayısı artacaktır, buda algoritmayı zorlaştırmaktadır. Tasarımı zor olduğu için kod yazan içinde durum zordur.

Çalışma ile ulaşılan sonucun aksini söyleyen az sayıdaki makaleler incelendiğinde, bu makalelerin bir dizi yapısıyla az sayıda girdi üzerinden test edilerek genel yargılarda bulunulduğu gözlemlenmiştir.

9. SONUÇ VE ÖNERİLER

Klasik kısıtlı kaynaklarla yazılım projelerinin çizelgelemesinde projenin faaliyetleri, proje maliyetlerini en aza indirecek şekilde çizelgelenmektedir. Var olan en iyi tam çözüm algoritmaları çok sayıda aktiviteden oluşan proje programı oluşturma problemlerini oluşturamazlar. Bu yüzden, optimale yakın çizelgeler üretmeleri için sezgisel algoritmalar araştırmacıların özel ilgi alanındadır. Araştırmacılar daha iyi çözüme ulaşabilecek sezgisel algoritmalar ortaya çıkarmak için yoğun olarak çalışmaktadırlar.

Bu projede genel kullanımı büyük projelerin yönetim metodu olan sarmal yazılım geliştirme metodu ile son zamanlarda gelişen ve daha küçük parçalara bölerek projelerin yönetiminde etkin rol alan çevik yazılım geliştirme metodunun sezgisel algoritmalar ile aktivitelerinin çizelgelenmesi anlatılmıştır. Sezgisel algoritma olarak genetik algoritma, ısıl işlem algoritması ve karınca koloni algoritması kullanılarak sonuçlar maliyet ve verimlilik açısından karşılaştırılmıştır.

Genetik algoritmanın en büyük avantajı paralel çalışmasıdır. Büyük problemler için bu çok önemli bir kazançtır. Büyük bir çözüm uzayını hızlı şekilde tarar. Genetik algoritmanın verimli olabilmesi için uygunluk fonksiyonunu ve mutasyon tanımlamasına dikkat edilmelidir. Genetik algoritmada en büyük problem erken yakınsamadır. İyi bir birey yani iyi bir çözüm, diğer çözümleri de etkisi altına alarak global çözümlerin araştırılmasını engelleyebilir. Bunu aşmak için mutasyon oranının uygun değerde seçilmesi ve toplumların çeşitlendirilmesi gerekmektedir. Probleme özel algoritmanın uygulaması tasarlanması gerekir.

Karınca koloni algoritması Paralel çözümler üreterek çalışır. Olumlu geri bildirim yöntemleri ile hızlı çözümler sağlar. Değişikliklere kolay uyarlanması nedeniyle dinamik uygulamalar için kullanılabilir. Teorik olarak analizi zordur. Yineleme sayısına bağlı olarak rasgele olasılıksal arama değişir. Teorik yerine deneysel araştırmalara uyarlanır. Çözüme yakınsama zamanı öngörülemez.

Isıl işlem algoritması yerel iyi çözümlerden kaçınır. Bunu yaparken hem yokuş yukarı, hem de yokuş aşağı davranışlarda bulunur. Problemler matematiksel olarak türevlenebilir olmak zorunda değildir. Karmaşık problemler için kodlaması kolaydır. Genel olarak iyi sonuçlar verir. Rasgele çözüm aradığı için büyük çözüm uzayının olduğu uygulamalarda uygulanması efektif değildir. En iyi çözüme ulaşıp ulaşılmadığını söyleyemez. Bunun için diğer algoritmalara ihtiyaç duyar. Isıl işlemlerde soğutma çok yavaş olmalıdır, hızlı olması durumunda çözüm uzayının tamamı aranmayacaktır. Bu yavaş arama nedeniyle çözümün bulunması zaman almaktadır. Çözüm aralıkları çok fazladır ve yakınsama yavaştır.

Gelecek çalışmalarda adaptif olarak kendisini geliştiren optimizasyon algoritmaları üzerine çalışmalar yapılarak, sezgisel algoritmalarda en önemli unsur olan başlangıç parametreleri ve başlangıç çözümünün geliştirilmesi üzerine çalışmalar yapılabilir.

KAYNAKÇA

Kitaplar

Holland, J., 1975. *Adaptation in Natural and Artifical Systems.* Ann Arbor, MI: University of Michigan Press.

Karaboğa, D., 2011. *Yapay Zeka Optimizasyon Algoritmaları.* 2. dü. Kayseri: Nobel Yayın Dağıtım.

Süreli Yayınlar

Adewole, A., Otubamowo, K. & Egunjobi, T., 2012. A Comparative Study of Simulated Annealing and Genetic Algorithm for Solving the Travelling Salesman Problem. *International Journal of Applied Information Systems,* 4(4), pp. 6-12.

Alhanjouri, M., Alfarra, B., 2013.Ant Colony versus Genetic Algorithm based on Travelling Salesman Problem. *Mohammed AlhanjouriBelal Alfarra, Int. J. Comp. Tech. Appl.* 2(3), pp.570-578

Dorigo, M. & Gambardella, L., 1997. Ant colonies for the traveling salesman problem. *Biosystems,* Cilt 43, pp. 73-81.

Dorigo, M. & Gambardella, L., 1997. Ant Colony System: A Cooperative Learning Approach to theTraveling Salesman Problem. *IEEE Transaction on Evolutionary Computation,* Cilt 1, pp. 53-36.

Grefenstette, J., 1986. Optimisation of Control parameters for Genetic Algorithms. *IEEE Trans. on Systems,* Vol. SMC-16,No.1(Man and Cybernetics), pp. 122-128.

Karaoğlan, İ., Altıparmak, F., 2005.Konkav Maliyetli Ulaştırma Problemi İçin Genetik Tabanlı Sezgisel Bir Yaklaşım. *J. Fac. Eng. Arch. Gazi Universitesi*, 20(4), pp.443-454

Kelley, J., 1961. Critical path planning and scheduling: Mathematical basis. *Operations Research,9,* pp. 296-320.

Kirkpatric, S., Gelatt, C. & Vecchi, P., 1983. Optimization by Simulated Annealing. *Science,* Cilt 220, pp. 671-680.

Kolisch, R., 1996. Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. *European Journal of Operational Research,90,* pp. 320-333.

Kuzu S., Önay O., Şen U., Tunçer M., Yıldırım B.F., Keskintürk T., 2014.Gezgin satıcı problemlerinin metasezgisel ile çözümü. *İstanbul Üniversitesi İşletme Fakültesi Dergisi*, 43(1), pp.1-27

Metropolis, N. et al., 1953. Equation of state Calculations by Fast Computing Machines. *J. Chem. Phys.,* Cilt 21, pp. 1087-1092.

ÖZDAMAR, L. & ULUSOY, G., 1995. A survey on the resource-constrained project scheduling problem. *IIE Transactions,* Cilt 27, pp. 574-586.

Paksoy, S. & Uzun, A., 2008. GENETİK ALGORİTMA İLE KAYNAK KISITLI PROJE ÇİZELGELEME. *Ç.Ü. Sosyal Bilimler Enstitüsü Dergisi, Cilt 17, Sayı 2*, pp. 345-362.

Purian, F., Farokhi, F. & Nadooshan, R., 2013. Comparing the Performance of Genetic Algorithm and Ant Colony Optimization Algorithm for MobileRobot Path Planning in the Dynamic Environments with Different Complexities. *Journal of Academic and Applied Studies,* 3(2), pp. 29-44.

Santos, E. & Zhong, X., 2001. Genetic algorithms and reinforcement learning for the tactical fixed interval scheduling problem. *International Journalon Artificialintelligence Tools,* Cilt 10, pp. 23-38.

Stinson, J., Davs, E. & Khumawala, B., 1978. Multiple resource-constrained scheduling usingbranch and bound. *AIIE Transactions,* pp. 252-259.

Stützle, T. & Hoos, H., 2000. Max Min Ant System. *Journal of Future Generation Computer,* 8(16), pp. 889-914.

Verma, S. & Singhal, P., 2009. Flow-shop Sequencing Model using Genetic Algorithm. *International Journal of Computational and Applied Mathematics ISSN,* 4(2), pp. 111-114.

Diğer Yayınlar

ATEŞ, E., 2012. *karınca Kolonisi Optimizasyonu Algoritmaları İle Gezgin Satıcı Probleminin Çözümü ve 3 Boyutlu Benzetimi.* İzmir: Ege Üniversitesi Mühendislik Fakültesi.

Beydağlı, E., Kara, M., Bahşi, H. & Alparslan, E., 2009. *Güvenli Yazılım Geliştirme Modelleri ve Ortak Kriterler Standardı.* s.l., s.n.

Blazewicz, J., Lenstra, J. & Rinnooy Kan, A., 1983. Scheduling sunject o resource constraints: Classification and Complexity. *Discrete Applied Mathmatics*, pp. 11-24.

COŞKUN, A., 2007. Yapay Zeka Optimizasyon Teknikleri: Literatür Değerlendirmesi. *Doğu Anadolu Bölgesi Araştırmaları,* pp. 142-146.

ÇÖZÜMLERİ, A. Y., tarih yok *ÇEVİK YAZILIM GELİŞTİRME.* [Çevrimiçi] Available at: http://www.acm-software.com/Pdf/AboutAgile.pdf [%1 tarihinde erişilmiştir01 Aralık 2014].

De Jong, K., 1975. *Analysis of the Behavior of a Class of Genetic Adaptive Systems.* MI, Ph.D. Dissertation, Department of Computer and Communication Science, University of Michigan.

Dorigo, M. & Gambardella, L., 1997. Ant colonies for the traveling salesman problem. *Biosystems,* Cilt 43, pp. 73-81.

Dorigo, M., Maniezzo, V. & Colorni, A., 1991. *Positive Feedback as a Search Strategy,* Milano: Technical Report N.

Eliiyi, D., 2004. *Operational fixed job scheduling problem,* Ankara: Middle East Technical University.

Engin, N., Lelebbicioğlu & K. Ünver, Z., 1996. *Sınama Noktası Yerleştirme Sorunu İçin Bir Genetik Algoritma.* İstanbul, TOK'96, Otomatik Kontrol Bilimsel Toplantısı, pp. 201-210.

Erdoğan, Ş., 2008. *Kendini klonlayan karınca kolonisi yaklaşımıyla optimal yolun bulunması.* Edirne: Trakya Üniversitesi Fen Bilimleri Enstitüsü.

Kandemir, A., Leblebicioğlu, K. & Ünver, Z., 1996. *Atlama Noktası Sayısını En Aza İndirme Probleminin Genetik Algoritma ile Çözümü.* İstanbul, TOK'96, Otomatik Kontrol Bilimsel Toplantısı, pp. 211-216.

Kolisch, R., 1995. *Project Scheduling under Resource Constraints,* Heidelberg: s.n.

Köksalan, M. & Erkip, N., 2000. Yöneylem Araştırması - Halim Doğrusöz'e Armağan. pp. 89-128.

Özdemir, G., 2006. *KISITLI KAYNAKLARLA PROJE ÇİZELGELEMESİ PROBLEMLERİNDE KULLANILAN GENETİK ALGORİTMA METODLARI VE BUNLARIN KARŞILAŞTIRILMASI.* ANKARA, ANKARA ÜNİVERSİTESİ, pp. 69-70.

Paksoy, S., 2007. *GENETİK ALGORİTMA İLE PROJE ÇİZELGELEME,* Adana: ÇUKUROVA ÜNİVERSİTESİ.

Pham, D. & Karaboğa, D., 2000. *Intelligent Optimisation Techniques: Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks,* London: Springer Verlag.

Schaffer, J., Caruana, R., Eshelman, L. & Das, R., 1989. *A study of Control Parameters Affecting On-Line Performance of Genetic Algorithms for Function Optimisation.* s.l., George Mason University, pp. 51-61.

Schirmer, A., 1998. *Case-based reasoning and improved adaptive search for project scheduling,* Universitaet Kiel,Almanya: s.n.
SDDS, 1990. s.l., s.n.

Şimşek, F. & Mergen, F., tarih yok *Asenkron Motorun Çok Hedefli Tasarım Optimizasyonu.* [Çevrimiçi]
Available at: http://www.emo.org.tr/ekler/53826d77b2f4224_ek.pdf
[%1 tarihinde erişilmiştir15 Aralık 2014].

Ulusoy, G., 2000. *Proje Planlamada Kaynak Kısıtlı Çizelgeleme.* [Çevrimiçi]
Available at:
http://research.sabanciuniv.edu/16897/1/HalimDogrusoz_2000.pdf
[%1 tarihinde erişilmiştir02 Aralık 2014].

Yazılım Geliştirme Modelleri, http://sulc3.com/model.html [erişim tarihi 02.12.2014]

Yılmaz, G., 2007. *Yazılım Geliştirme Yaşam Döngüsü.* [Çevrimiçi]
Available at: www.metinakbulut.com/YAZILIM-MIMARISI/Bolum-02.ppt
[%1 tarihinde erişilmiştir1 Aralık 2014].

EKLER

EK 1: Örnek Sarmal Proje İçeriği

Ek 1.1: Sarmal proje insan kaynağı ve verimlilik değerleri

sid	name	surname	responsibility	salary	efficient	value
1	EKREM	GULTEKIN	PLANNING	2000	1	0
2	KADIR	BULBUL	DEVELOPMENT	2500	1	0
3	SEMIH	ULUTAS	DEVELOPMENT	3000	0,8	0
4	FIRAT	CINAR	RISKMANAGEMENT	3000	0,8	0
5	RAUF	AKBAS	PLANNING	3500	0,9	0
6	SONAT	KARAKAŞ	RISKMANAGEMENT	2000	1	0
7	SELÇUK	ARSLAN	DEVELOPMENT	2600	1	0
8	SUAT	GULMUS	DEVELOPMENT	2700	1	0
9	BİROL	AKPINAR	UAT	2800	1	0
10	FUNDA	ERTUGRUL	PLANNING	2700	1	0
11	BİLAL	ÖZTÜRK	RISKMANAGEMENT	2000	1	0
12	FİLİZ	ARISOY	DEVELOPMENT	2300	1	0
13	KADRİYE	GENCEL	DEVELOPMENT	2400	1	0
14	HÜSEYİN	KOCA	DEVELOPMENT	4000	0,7	0
15	KAHAN	YUREKLITURK	UAT	3400	0,9	0
16	CEMIL	YUKSEL	PLANNING	3300	0,9	0
17	ALPER	GADİŞ	DEVELOPMENT	3200	0,7	0
18	İLTEBER	DİNÇER	UAT	3700	0,8	0
19	ERSİN	ETİK	RISKMANAGEMENT	2700	0,9	0
20	MEHMET	BECERIK	DEVELOPMENT	2900	0,8	0
21	GÜROL	BOZAN	UAT	3100	0,9	0
22	SAKIR	KEMIK	UAT	3000	1	0
23	SEMIH	OGUN	PLANNING	3500	0,8	0
24	VELI	TUNCER	UAT	2200	1	0
25	BULENT	KIVANC	UAT	2500	1	0
26	TUNCER	SALI	DEVELOPMENT	2700	0,9	0
27	ERSIN	ADAL	DEVELOPMENT	4000	0,6	0
28	ANDRE	BLASER	UAT	3800	0,8	0
29	AKIF	DERYA	PLANNING	2100	1	0
30	EROL	OZGUNER	UAT	3500	0,8	0
31	ERSİN	LEBLEBİ	RISKMANAGEMENT	2700	0,8	0

Ek 1.2: Sarmal proje örnek aktivite adımları

value	slotBit	slotNumber	hourPerSlot
0	1	1	2
0	2	1	2
0	3	1	2
0	4	1	2
0	5	1	2
0	6	2	2
0	7	2	2
0	8	2	2
0	9	2	2
0	10	2	2
0	11	3	2
0	12	3	2
0	13	3	2
0	14	3	2
0	15	3	2
0	16	4	2
0	17	4	2
0	18	4	2
0	19	4	2
0	20	4	2
0	21	5	2
0	22	5	2
0	23	5	2
0	24	5	2
0	25	5	2
0	26	6	2
0	27	6	2
0	28	6	2
0	29	6	2
0	30	6	2
0	31	7	2
0	32	7	2
0	33	7	2

Ek 1.3: Sarmal proje adımları

value	step
1	PLANNING1
1	RISKMANAGEMENT1
1	DEVELOPMENT1
1	UAT1
1	PLANNING2
1	RISKMANAGEMENT2
1	DEVELOPMENT2
1	UAT2
1	PLANNING3
1	RISKMANAGEMENT3
1	DEVELOPMENT3
1	UAT3
1	PLANNING4
1	RISKMANAGEMENT4
1	DEVELOPMENT4
1	UAT4
1	PLANNING5
1	RISKMANAGEMENT5
1	DEVELOPMENT5
1	UAT5

Ek 1.4: Sarmal proje çalışanın uygun olduğu zaman seçimi

sid	slotNumber	value
1	5	0,10
29	6	0,20

EK 2: Örnek Çevik Proje İçeriği

Ek 2.1: Çevik proje insan kaynağı ve verimlilik değerleri

sid	name	surname	responsibility	salary	efficient	value
1	EKREM	GULTEKIN	PLANNING	2000	1	0
2	KADIR	BULBUL	DEVELOPMENT	2500	1	0
3	SEMIH	ULUTAS	DEVELOPMENT	3000	0,8	0
4	FIRAT	CINAR	DESIGN	3000	0,8	0
5	RAUF	AKBAS	PLANNING	3500	0,9	0
6	SONAT	KARAKAŞ	DESIGN	2000	1	0
7	SELÇUK	ARSLAN	DEVELOPMENT	2600	1	0
8	SUAT	GULMUS	DEVELOPMENT	2700	1	0
9	BİROL	AKPINAR	UAT	2800	1	0
10	FUNDA	ERTUGRUL	PLANNING	2700	1	0
11	BİLAL	ÖZTÜRK	DESIGN	2000	1	0
12	FİLİZ	ARISOY	DEVELOPMENT	2300	1	0
13	KADRİYE	GENCEL	DEVELOPMENT	2400	1	0
14	HÜSEYİN	KOCA	DEVELOPMENT	4000	0,7	0
15	KAHAN	YUREKLITURK	UAT	3400	0,9	0
16	CEMIL	YUKSEL	PLANNING	3300	0,9	0
17	ALPER	GADİŞ	DEVELOPMENT	3200	0,7	0
18	İLTEBER	DİNÇER	UAT	3700	0,8	0
19	ERSİN	ETİK	DESIGN	2700	0,9	0
20	MEHMET	BECERIK	DEVELOPMENT	2900	0,8	0
21	GÜROL	BOZAN	UAT	3100	0,9	0
22	SAKIR	KEMIK	UAT	3000	1	0
23	SEMIH	OGUN	PLANNING	3500	0,8	0
24	VELI	TUNCER	UAT	2200	1	0
25	BULENT	KIVANC	UAT	2500	1	0
26	TUNCER	SALI	DEVELOPMENT	2700	0,9	0
27	ERSIN	ADAL	DEVELOPMENT	4000	0,6	0
28	ANDRE	BLASER	UAT	3800	0,8	0
29	AKIF	DERYA	PLANNING	2100	1	0
30	EROL	OZGUNER	UAT	3500	0,8	0
31	ERSİN	LEBLEBİ	DESIGN	2700	0,8	0

Ek 2.2: Çevik proje örnek aktivite adımları

value	slotBit	slotNumber	hourPerSlot
0	1	1	8
0	2	1	8
0	3	1	8
0	4	1	8
0	5	1	8
0	6	1	8
0	7	1	8
0	8	2	8
0	9	2	8
0	10	2	8
0	11	2	8
0	12	2	8
0	13	2	8
0	14	2	8
0	15	3	8
0	16	3	8
0	17	3	8
0	18	3	8
0	19	3	8
0	20	3	8
0	21	3	8
0	22	4	8
0	23	4	8
0	24	4	8
0	25	4	8
0	26	4	8
0	27	4	8
0	28	4	8

Ek 2.3: Çevik proje adımları

value	step	Iteration	Description
3	PLANNING1	Iteration 1	Requirement Planning
2	DESIGN1	Iteration 1	Design&Risk Management
1	DEVELOPMENT1	Iteration 1	Development
1	UAT1	Iteration 1	UAT
2	PLANNING2	Iteration 2	Requirement Planning
2	DESIGN2	Iteration 2	Design&Risk Management
2	DEVELOPMENT2	Iteration 2	Development
2	UAT2	Iteration 2	UAT
2	PLANNING3	Iteration 3	Requirement Planning
2	DESIGN3	Iteration 3	Design&Risk Management
2	DEVELOPMENT3	Iteration 3	Development
2	UAT3	Iteration 3	UAT
2	PLANNING4	Iteration 4	Requirement Planning
1	DESIGN4	Iteration 4	Design&Risk Management
3	DEVELOPMENT4	Iteration 4	Development
2	UAT4	Iteration 4	UAT
1	PLANNING5	Iteration 5	Requirement Planning
1	DESIGN5	Iteration 5	Design&Risk Management
2	DEVELOPMENT5	Iteration 5	Development
2	UAT5	Iteration 5	UAT

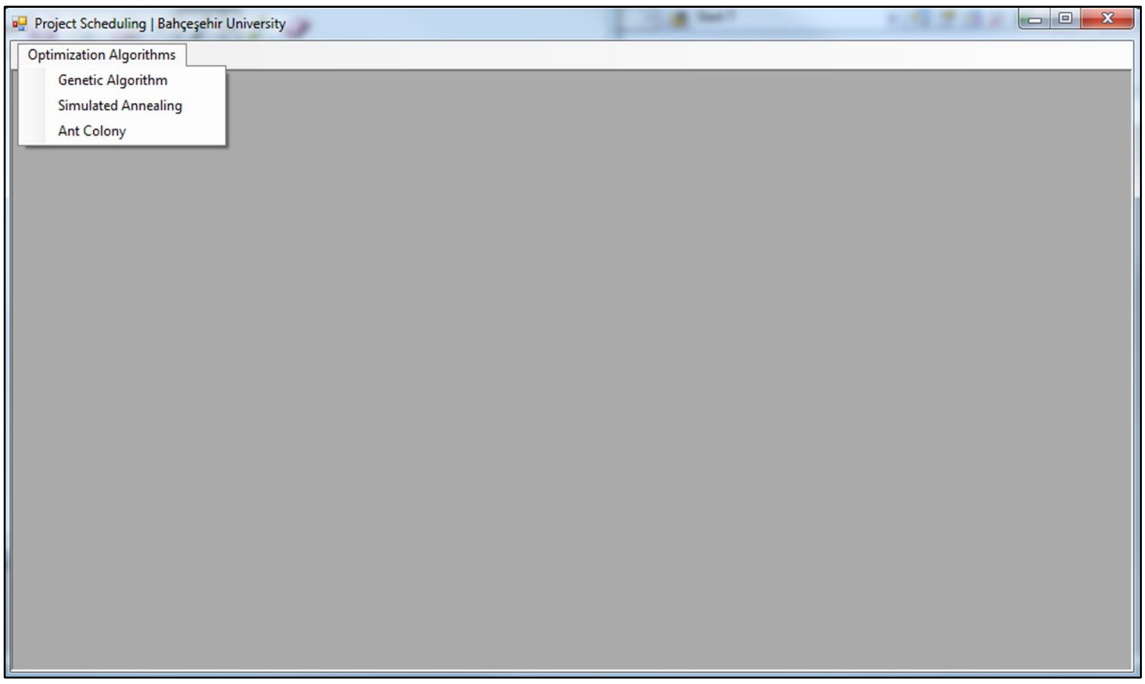
Ek 2.5: Çevik proje çalışanın uygun olduğu zaman seçimi

sid	slotNumber	value
1	5	0,10
29	6	0,20

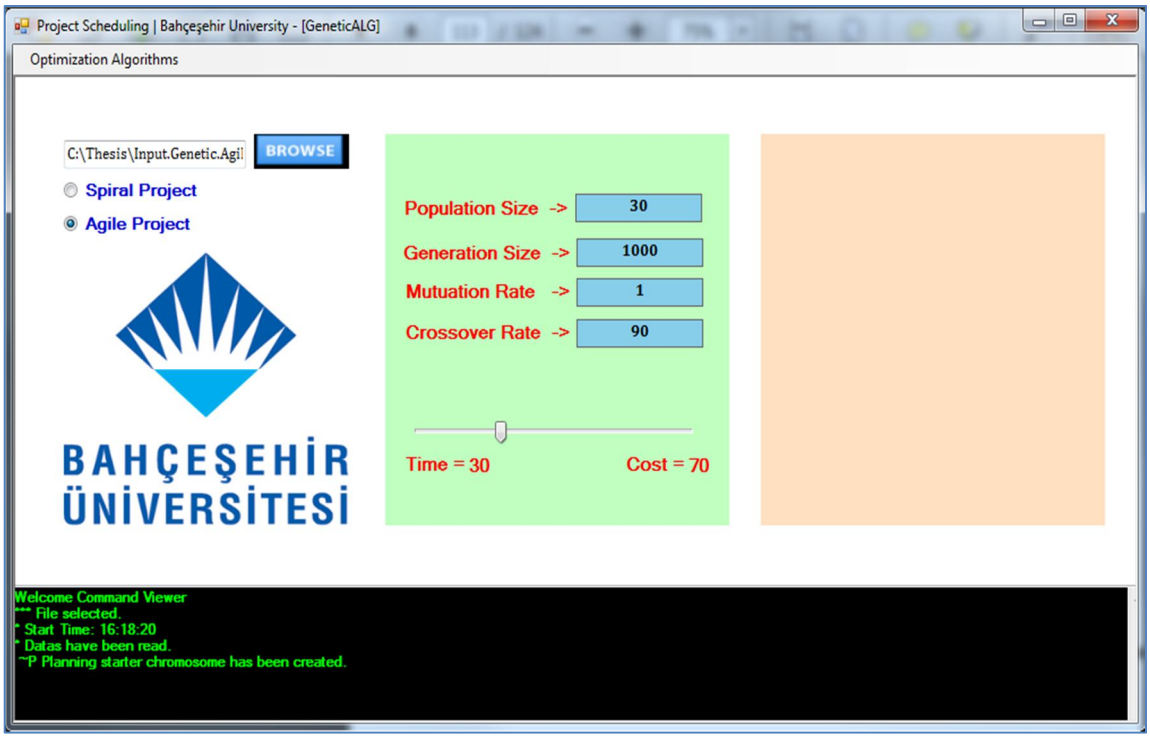
EK 3: Yazılım Geliştirme Projesi Test Uygulaması

Bu program .Net visual studio 2008 ortamında c# dilinde framework 3.5 kullanılarak yazılmıştır. Programın kullanılabilmesi için 64 bit Windows XP ve üstü işletim sistemlerinden birisi ve MS Office 2010 kurulu olmalıdır.

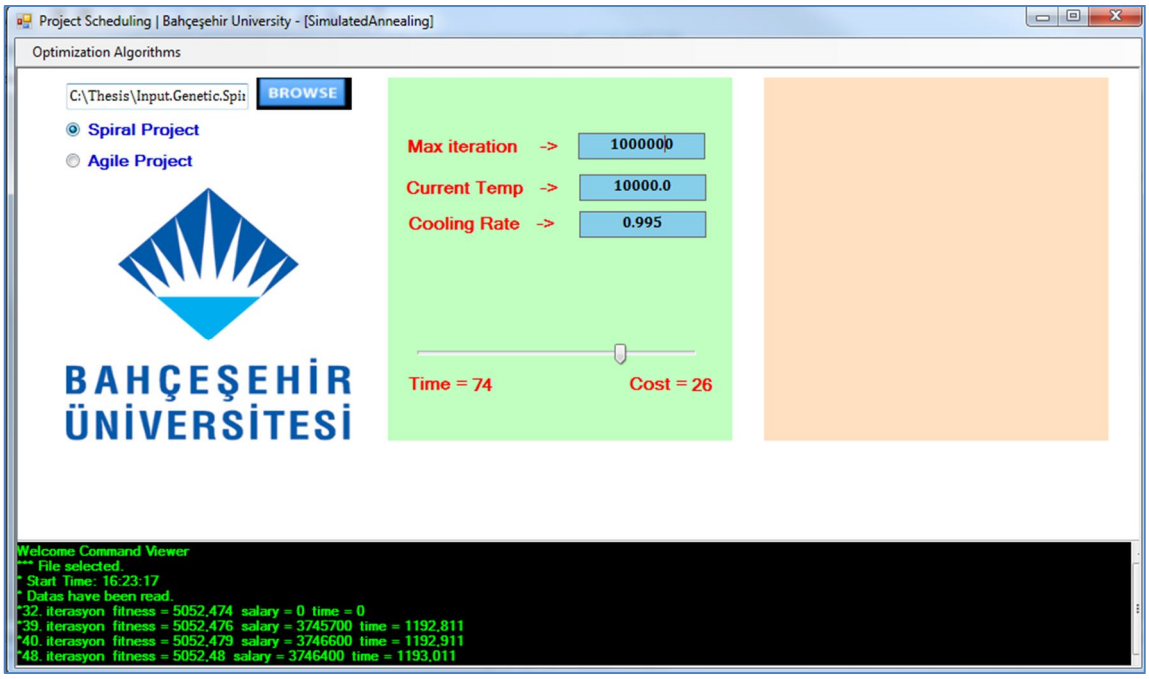
Ek 3.1: Test uygulaması ana ekranı



Ek 3.2: Genetik algoritma test ekranı



Ek 3.3: Isıl işlem algoritması test ekranı



Ek 3.4: Karınca koloni algoritması test ekranı