YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BENZER OLMAYAN PARALEL MAKİNELERİN ÇİZELGELENMESİNDE BULANIK ESASLI PROSES ZAMANLARININ GENETİK ALGORİTMA UYGULAMASI

Endüstri Mühendisi Pelin ALCAN

FBE Endüstri Mühendisliği Anabilim Dalı Endüstri Mühendisliği Programında Hazırlanan

YÜKSEK LİSANS TEZİ

Tez Danışmanı **:** Prof. Dr. Hüseyin BAŞLIGİL (Yıldız Teknik Üniversitesi)

İSTANBUL, 2008

İÇİNDEKİLER

Sayfa

SİMGE LİSTESİ

X_a	Bulanık sayı alt sınır
X_b	Bulanık sayı üst sınır
$ü(X)$	Üyelik derecesi
$ü_A$	A kümesini tanımlayan üyelik derecesi
R	Bulanık çıktı kümesi
j	İş sayısı
i	Makine sayısı
$t(i, j)$	i makinesindeki j işinin proses zamanı.
$C_{\max}$	Maksimum tamamlanma zamanı.
g	Makinelerin yapacakları işleri gösteren genler.
n	İş sayısı.
m	Makine sayısı.
k	Kromozom tanımı.
F_k	Uygunluk değeri. $S(k)$
α	Uygunluk değeri içindeki reel pozitif sayı.
β	Uygunluk değeri içindeki reel pozitif sayı.
$P(k)$	k kromozomunun seçilme ihtimali.
$S(k)$	Kromozom seçimi için kesim noktaları.
V	Makine hızı.
$F_{\max}$	Maksimum akış zamanı
S_t	Kurum zamanı
a_i	Müsaade edilen zaman.
d_i	Uygunluk değeri içindeki reel pozitif sayı.
W_i	J_i' nin toplam bekleme zamanı.
F_i	Akış zamanı.
r_i	Hazırlık zamanı.
P_{ji}	j makinesinde i işinin işlem zamanı.
L_i	C_i' nin gecikme zamanı.

C_i	Tamamlanma zamanı.
T_i	J_i nin gecikmesi; j makinesindeki i işinin gecikmesi.
E_i	J_i' nin erken bitirilmesi
$\bar{X}$	İşlerin ortalaması.
P_m	Paralel benzer makine.
Q_m	Farklı hızlardaki paralel makine.
R_m	Birbirine paralel ilişkisiz makineler.
$N_w(t)$	t anında işlenmeye hazır olmayan veya makineler arasında bekleyen işlerin sayısı.
$N_c(t)$	t anında tamamlanmış olan işlerin sayısı.
$N_u(t)$	t anında tamamlanacak olan işlerin sayısı. Bitmek üzere olan işlerin sayısı.
$N_p(t)$	t anında işlenmekte olan işlerin sayısı.
F_m	Flow shop hali.
FF_s	Esnek flow shop hali.
O_m	Open shop hali.
J_m	Job shop hali.
prmp	Alıkoyma hali.
prec	Öncelik zorlama hali
brkdown	Bozulma hali.
M_j	Seçilebilir makine kısıtlamaları.
prmu	Yerdeğiştirme (permütasyon).
block	Önleme.

KISALTMA LİSTESİ

EDD	En erken teslim tarihi algoritması.
FPTAS	Çoklu polinomial zaman yaklaşım şeması.
GA	Genetik Algoritma
LPT	En uzun proses zamanı algoritması.
LP	Lineer programlama
MDD	Değiştirilmiş teslim tarihi algoritması.
MPGA	Çok nüfuslu genetik algoritma tekniği
SA	Tavlama benzetimi.
TA	Basamak kabulü.

ŞEKİL LİSTESİ

Sayfa

ÇİZELGE LİSTESİ

Sayfa

ÖNSÖZ

Bana böylesi önemli bir konuda çalışma, düşünme, fikir edinme, bir şeyler öğrenme ve uygulama imkanı veren ve çalışmam esnasında desteğini hiç esirgemeyerek beni yönlendiren çok değerli danışman hocam Sn. Prof. Dr. Hüseyin BAŞLIGİL' e, çalışmamı tamamlamamda çok büyük bir payı olan, eleştirileriyle beni bir an olsun yalnız bırakmayan, sonsuz iyimserliğiyle her zaman yanıbaşımda olan, çok sevdiğim iş arkadaşım ve sırdaşım Arş. Gör. Şükran Şeker'e, tez boyunca verdikleri moralle yanımda olduklarını her daim hissettiğim öğrencilerime, diğer bütün yol arkadaşlarıma ve beni bugünlere getiren, kendisini tanıma fırsatı bulamasam da, fikirlerimin ve yaşamımın şekillenip büyümesinde çok büyük katkıları olan biricik dedem İsmail Yılmaz' a ve çok sevgili diğer aile büyüklerime teşekkürlerimi bir borç bilirim.

ÖZET

Günlük yaşantımızda, kesin olduğunu düşündüğümüz ancak gerçekte kesin olmayan durumlarla karşılaşırız. Bu durumların sistematik bir biçimde öngörülebilmesi ancak bazı kabullerin yapılmasından sonra mümkün olmaktadır. Birçok sosyal, ekonomik ve teknik olayda da belirsizlik ve dolayısıyla karmaşıklık bulunmaktadır. Bu belirsizliklerin analiz edilmesi Zadeh tarafından geliştirilen bulanık mantık teorisi kapsamında mümkündür.

Genetik Algoritma, olası tüm çözümlerin değil de salt bâzı seçilmiş çözümlerin denenmesi yoluyla beklenen optimum sonucu bulmaya çalışan, parametre kodlama temeline dayanan bir arama tekniğidir. GA'lar doğada geçerli olan "en iyinin yaşaması" kuralına dayanarak sürekli iyileşen çözümler üretir. Bunun için "iyi" nin ne olduğunu belirleyen bir uygunluk fonksiyonu ve yeni çözümler üretmek için yeniden kopyalama ve değiştirme gibi operatörleri kullanır.

Bu çalışmada, benzer olmayan paralel makinelerde, iş sıralama probleminde bulanık proses zamanlarına bağlı olarak, önce bulanık mantık teorisinin esasları anlatılacak ve daha sonra genetik algoritma metodu özetlenecektir. Bulanık mantığın ardından genetik algoritma hakkında bilgi verilecek ve bu konularda bir uygulama yapılacaktır. Problem sonuçları GA ile ilgili olarak Java Eclipse Europa programında anlatılmaktadır.

(Genetik Algoritma, Bulanık Mantık, İş Sıralama, Benzer Olmayan Paralel Makineler)

ABSTRACT

In our daily lives we frequently come across with circumstances that we think of certain but in fact are not. Prediction of these circumstances in a systematic manner is possible only after making some assumptions. In a variety of social, economic and technical events uncertainty and therefore complexity is always present. It is possible to analyze those uncertainties within the context of the fuzzy logic theory developed by Zadeh.

GA is a search method based on parameter encoding that purposes optimum outcome using some of chosen solutions. GA produces continually improving solutions on the basis of nature rule which provides "living which has best properties". Because of this, it uses fitness function that determines "best properties" and cross over operator for producing new solutions. GA is an evolutionary calculation technique which expands with Artificial Intelligence.

In this study we first summarize the fundamentals of the fuzzy logic theory and then summarize genetic algorithm method in the scheduling problem with fuzzy processing times on non-identical parallel machines. After that an application which is analized with fuzzy logic and genetic algorithm is defined. Problem solutions in job sequencing with GA is examined, by using a program which is prepared in Java Eclipse Europa Program.

(Genetic Algorithm, Fuzzy Logic, Job Sequencing, Non-identical Parallel Machines)

1. GİRİŞ

Hayatın büyük bir bölümü sıralama problemlerinden oluşmaktadır. Bu durum bilinçsizce olabilmektedir. Adalet duygusu ve daha bir çok faktör bu bağlamda ön plana çıkmaktadır. Sıralama işlemi üretim, imalat, yönetim, bilgisayar bilimi ve bunun gibi pek çok alanda kullanılan önemli bir prosestir.

Ayrıca günümüzün rekabete dayalı ortamında, işletmeler en az miktarda kaynak kullanarak, müşteri gereksinimlerine en hızlı yanıt verebilecek tekniklerle en kaliteli ürün ve hizmet üretmenin peşindedirler. Bu yüzden, hızlı değişen müşteri talepleri karşısında özellikle üretim planlarını en çabuk oluşturan işletmeler rekâbette bir adım öne geçecektir. Etkili planlar kurabilmek için de işletmeler optimizasyon tekniklerinden yararlanmak zorundadırlar. Klasik analitik yöntemler, şimdiye kadar plânların oluşturulmasında en çok kullanılan yöntemlerden biri idi. Ama, üretim sistemlerinde bulunan çok fazla rassal parametre ve sürekli değişen çevre, kurulması istenen bu planların yavaş oluşturulmasına ve etkisizleşmesine neden olabilmektedir (Baksak ve Erol, 2004).

Genetik Algoritma, biyolojik bir sistemin, çevresine adaptasyonunda kullandığı metodun örneklendirilmesidir. Bilgisayarda, bu tür çok parametreli optimum bulma problemlerine ve makine öğrenme problemlerine çözüm modeli olarak alınabilir. Doğal adaptasyondan esinlenen GA' nın basit olarak iskeleti:

a) Bireyin bulunduğu ortamda hayatta kalmak için, kendi kendisini değiştirerek ortama uygun hale gelmesi,
b) Bu adaptasyon boyunca, yeni üretilecek nesillere, bu özellikler ile birlikte mümkün olabilecek daha çok değişim aktarılarak, bireylerin daha çok uyumlu hale getirilmesi olarak özetlenebilir.

Burada yapılan çalışma sadece iş sıralamada genetik algoritma yaklaşımının kullanımı değildir. Aynı zamanda işlerin makinelere atanan proses zamanlarının bulanık zaman esaslı olarak ele alınarak genetik algoritma uygulamasına gidilmesi de araştırılmaktadır.

Günlük yaşantımızda, kesin olduğunu düşündüğümüz ancak gerçekte kesin olmayan durumlarla karşılaşırız. Bu durumların sistematik bir biçimde öngörülebilmesi ancak bazı kabullerin yapılmasından sonra mümkün olmaktadır. Birçok sosyal, ekonomik ve teknik

olayda da belirsizlik ve dolayısıyla karmaşıklık bulunmaktadır. Bu belirsizliklerin analiz edilmesi Zadeh tarafından geliştirilen bulanık mantık teorisi kapsamında mümkün olmaktadır.

Bu nedenle, çok değişkenli sistemlerde oldukça etkili olan ve gelişimini hâlâ sürdüren bir optimizasyon tekniği olan "Genetik Algoritma" yöntemi, araştırmacılar tarafından bu plânların oluşturulmasında kullanılmaya başlanmıştır. Bu çalışmada, bir Endüstri Mühendisliği konusu olan "İş Sıralama" üzerinde Genetik Algoritma yöntemi incelenmektedir ve bu yöntemle benzer olmayan paralel makineler üzerinde (non-identical) iş sıralama problemine çözümler sunan bir programla uygulama yapılmaktadır.

2. LİTERATÜR ARAŞTIRMASI

Lopes ve Carvalho (2005) bir dizi bağımsız işin sıralanması problemi için, bir dal ve maliyet yaklaşımı (branch and price approach) geliştirmiştir. Néron ve arkadaşları (2007) paralel makineler sıralama problemi için dal-sınır algoritmasını göstermektedirler. Rocha ve arkadaşları (2006) çalışmalarında bir dal-sınır algoritması geliştirir ve *metaheuristic GRASP* bir üst sınır olarak kullanılarak bir çözüm geliştirilir. Nessah ve arkadaşları (2005) makalelerinde benzer paralel makineler sıralama problemini incelerler. Toplam tamamlanma zamanının minimize edilmesi için boşta kalma süresi ve hazırlık zamanı tabanlı sıralama tartışılır. Buradaki problem NP-hard(zor) olarak bilinmektedir. Kilit bir kural olarak tanımlanan genel optimallik için yeterli ve gerekli bir durum geliştirilir. Sonra bu durum üzerinde şekillenen geçerli bir altküme tanımlanır. bu altkümelere ait bir sıralamanın yapılabilmesi için verimli, sezgisel algoritmalar kullanılır. Sezgisel yöntem olarak, dal-sınır algoritması kullanılır. Böylece daha düşük sınırlar ve geçerli şartlar birleştirilir. Chen ve Powell (1998) makalede m tane benzer paralel makinedeki n tane işin sıralanması problemi dağılım algoritması kullanarak yorumlamıştır. İlk olarak, problem tamsayılı bir program olarak formüle edilir. Daha sonra Dantzig-Wolfe dağılımı kullanılarak yeniden formüle edilir. Bu formülasyon temel alındığında, bir dal-sınır algoritması problem için geliştirilmektedir. Mokotoff ve Chretienne (2002) çalışmalarında bir liste sıralama algoritması ve dal-sınır algoritmasını ilişkisiz paralel makine sıralama problemi için önermektedirler. Alagöz ve Azizoğlu (2002) çalışmalarında, makine seçilebilirlik kısıtları altında, paralel makine yeniden sıralama problemini tanımlarken, bir LP modeli önerilmiştir. Ayrıca hiyerarşik probleme uygulanmak üzere bir dal-sınır algoritması geliştirilmiştir. Shim ve Kim (2006) m tane benzer paralel makinedeki n tane bağımsız işin sıralanması problemi için, bir dal-sınır algoritması önermiştir çalışmalarında. Aynı şekilde Nessah ve arkadaşları (2006) ile Dunstall ve Wirth (2004) de benzer paralel makine sıralama problemi için, bir dal sınır algoritması önermişlerdir. Ghirardi ve Potts (2004) makalede tamamlanma zamanını minimize etmek amaçlı ilişkisiz paralel makinelerde beam search algoritmasını denemektedirler. Geleneksel Beam Serach algoritması kesikli dal sınır algoritmasının bir parçasıdır. Liaw ve arkadaşları (2003) ilişkisiz paralel makinelerde verilen bir dizi bağımsız işlerin sıralama probleminde bir dal-sınır algoritması önermiştir.

Leonardi ve Raz (2006) ise makalelerinde işlerin akışı ile ilgili toplam akış zamanının optimize edilmesi problemini tartışmıştır. SRPT kuralı kullanılmıştır. Beraldi ve arkadaşlarının (2007) çalışmasında, yeni bir değişken zaman ufku (rolling horizon) ve

sabitle-gevşet sezgisel yöntemleri (fix and relax heuristics), benzer paralel makine parti büyüklükleri ile kurulum maliyeti esaslı çizelgeleme problemi için geliştirilmiştir.

Koulamas and Kyparisis (2007) iki benzer paralel makinede LPT (en uzun proses zamanı) algoritması uygulamıştır. Baker ve Bertrand' ın MDD (değiştirilmiş teslim tarihi) algoritmasını içine alan, değiştirilen bir teslim tarihi algoritması (MDD) tek makine ağırlıklandırılmamış problemi için gösterilmiştir. Mosheiov (2001) makalede paralel benzer makinelerde, minmax amaçlı, teslim tarihi atama problemi çözümünde sezgisel bir yöntem geliştirmiştir. Sezgisel iki aşamayı içermektedir: a) LPT yöntemi ile işlerin sıralanması, ve b) a adımında sağlanan sıralama için teslim tarihi atama probleminin çözümü. Koulamas ve Kyparisis (2000) makalede maksimum gecikmelerin minimize edilmesini amaçlayarak, üniform paralel makineler için bir sıralama problemi tartışmıştır. EDD (en erken teslim tarihi) kuralı bunun için genişletilmiştir. Liao ve Lin (2003) makalelerinde toplam tamamlanma zamanını minimize etmek amaçlı, iki tane üniform paralel makine problemi tartışmıştır. İki benzer paralel makine problemine dönüştürülen problem optimal bir algoritmayla çözülmeye çalışılır. Ayrıca LPT ve Multifit ezgiselleri de denenmiştir. Hwang ve arkadaşları (2003) ise paralel makine sıralama probleminde LPT algoritmasını geliştirmeye çalışmışlardır.

Omar ve Teo (2006) farklı teslim tarihleri, farklı proses zamanları ve erken teslim tarihi kısıtlarıyla, benzer paralel makinelerdeki toplam gecikmeyi minimize etmek için karma bir tamsayı programlama formülasyonunu göstermişlerdir.

Cochran ve arkadaşları (2003) makalelerinde çok amaçlı paralel makine çizelgeleme problemlerinin çözümü için iki safhalı çok nüfuslu genetik algoritma tekniği (MPGA) önermiştir. Chiu ve arkadaşları (1999) ise karma bir tamsayılı programlama modelini sıralı paralel makine operasyonları için göstermiştir. Genetik tabanlı bir algoritma optimal paralel operasyon sırasının bulunması için önerilmiştir. Armentano ve Filho (2007) makalede kurulum zamanına bağlı sıralama problemleriyle ilgili üniform paralel makineleri inceleyerek, GRASP tekniklerini önermiştir. Min ve Cheng (1999) çalışmalarında, benzer makine sıralama probleminde, tamamlanma zamanını minimize etmek için bir çeşit genetik algoritma tanımlanmaktadır. Liu ve Wu (2003) makalelerinde, evrimsel programlama metodunu benzer paralel makine üretim hattı çizelgeleme problemine geciken işlerin minimize edilmesi için uygulamışlardır. Diğer yandan, Jou (2004) çalışmasında bir üretim sıralama probleminde genetik algoritmayı çalışmıştır. Gupta ve Torres (2004) makalelerinde ortalama akış zamanını

ve geciken iş sayılarını içeren amaçları hesaba katan, paralel makinelerde sıralanan iş problemlerini tanımlamaktadır. Genetik algoritma, tabu arama, benzetim tavlaması gibi sezgiseller kıyaslanmakta ve çözüm için önerilmektedir. Chang ve arkadaşları (2005) ise çalışmalarında çok amaçlı sıralama probleminin çözümünde değişik bir genetik algoritma tekniğini (TPSPGA) denerler.

Kogan (2004) düzenli zaman bazlı çizelgeleme için polynomial zamanlı algoritma önermiştir. Liao ve Sheen (2007) de, polinomial zaman çifti (polynomial time binary) algoritmasını optimal bir çözüm bulmak için tartışmaktadırlar. Ji ve Cheng (2007) toplam tamamlanma zamanının minimizasyonunda m tane benzer makine için çoklu polinomial zaman yaklaşım şemasını (FPTAS) kullanırlar. Gupta ve Ho (2001)' nun çalışmasında, en küçük tamamlanma zamanını veren optimal sıralamanın verilmesi önemlidir. Optimal sıralama için bir algoritma önerilmiştir. Polinomial hesaplama zamanının içinde problemin yaklaşık çözümü için bir multifit algoritma geliştirilir. Hall ve arkadaşları (2000)' nın çalışmasında benzer paralel makine sıralama problemi için bir polinomial zaman algoritması geliştirilmiştir. Moukrim ve Quilliot (2004) benzer paralel makinelerde, en kısa uzunluktaki sıralamanın bulunması için bir polinomial zaman algoritması önermiştir.

Gairing ve arkadaşları (2007) ilişkisiz paralel makineler için birleştirici bir dağılım algoritması geliştirirler. Koulamas ve Kyparisis' in (2003) makalesinde, teslim zamanları durumu içinde üniform paralel makinelerdeki tamamlanma zamanı minimizasyon problemi tartışılırken dağılım sezgiselleri kullanılmıştır.

Gharbi ve Haouari (2005) makalelerinde, Jackson algoritmasının en kötü şekliyle bir ön işlem algoritması kullanıldığı zaman değerinin daha da artacağı kanıtlanmıştır. Logendran ve arkadaşları (2006) çalışmalarında, sıralama tabanlı ilişkisiz paralel makine sıralama problemini araştırırlar ve tabu yaklaşımını temel alan altı farklı yaklaşım algoritması geliştirirler. Baptiste ve Timkovsky (2001) çalışmalarında yeni bir teorem önermişlerdir. Özdeş paralel makinelerdeki, keyfi sıralanan proses zamanı işlerin önealımları üzerinde duran McNaughton' ın teoremi geliştirilmiştir.

Koulamas ve Kyparisis (2006) makalede benzer paralel makinelerle, iki safhalı montaj hattı sıralama problemindeki tamamlanma zamanının minimize edilmesi tartışılmıştır. Bu problem ilk safhada eş zamanlı süreçlerle montaj hattı probleminde genelleştirilir. İkinci safhada tek

bir montaj operasyonu vardır. İş sayıları genişlemeye başladıkça sezgisel bir yöntem kullanılmıştır. Torres ve arkadaşları (2006) makalelerinde üniform paralel makinelerdeki işlerin sıralama problemini incelemektedir. Burada makine hızı ikinci bir kaynağın tahsisiyle kontrol edilir. Geç kalan iş sayılarının minimize edilmesi önemlidir. Beş tip sezgisel iki tür olarak analiz edilip önerilir. Kravchenko ve Werner (1997) in çalışmasında, tamamlanma zamanı problemi iki benzer paralel makine için polynomial zamanda çözülemeyen bir durumdur. Söz konusu problem, keyfi makinelerle genelleştirilen, polynomial zamanda çözülemeyen (NP-hard) ve analizi sezgisel yöntemlerle yapılan bir problemdir. Makalede birli NP hard (unary NP-hard) yöntemi hazırlık zamanı kısıtları için geliştirilmektedir. Centeno ve Armacost (1997) makalede paralel makinelerdeki maksimum gecikmenin minimize edilmesi ile ilgili problemlerin çözümü için bir algoritma sunmuştur. Burada makine seçim kısıtları ve bırakma tarihleri(release dates) önemlidir. Gupta ve Torres (2000) makalede n iş m benzer makine sıralama problemi için yeni bir sezgisel önermektedir. Amaç optimal toplam akış zamanını elde etmek için maksimum tamamlanma zamanını minimize etmektir. Sezgisel varolan sezgisellerle ve uyarlamalarıyla kıyaslanınca deneyseldir.

Sivakumar ve Ganesan (2007) çalışmalarında paralel makine montaj sıralama problemini karma bir tamsayı programlama modeli olarak formüle ederler ve NP-tamamlanmış(NP-complete) olmak için ispatlanır. Sezgisel algoritma tabanlı benzetimli tavlama yöntemi, sıralama problemi çözümü için geliştirilir. Park ve Kim (1997)in makalelerinde benzer paralel makinelerdeki bir sıralama problemi için, iki ayrı sezgisel önerilmektedir: bunlar tavlama benzetimi ve tabu arama sezgiselleridir. Torres ve arkadaşları (1997) makalelerinde M benzer paralel makinedeki N işli sıralama probleminin çok kriterli problemini tartışmaktadır. Amaç, bekleyen işlerin sayılarını ve ortalama akış zamanını eş zamanlı olarak minimize etmektir. Dört sezgisel tavlama benzetimi ve neighborhood search tabanlı olarak gösterilmiştir. Meyr (2002) benzer olmayan paralel üretim hatlarında, meta-stratejilerden basamak kabulü (TA), tavlama benzetimi (SA) ve çift yeniden optimize etme metodlarının kombinasyonunu çalışmıştır. Cao ve arkadaşları (2003) çalışmalarında tabu arama mekanizmasını temel alan sezgisel bir algoritmayı, optimal veya optimale yakın çözümlerin bulunması için geliştirmişlerdir. Low (2004) ise çalışmasında ilişkisiz paralel makinelerle ilgili çok safhalı akış atölye tipi sıralama probleminde, benzetim tavlaması (SA) tekniğini önermektedir.

Alidaee ve Rosa (1997) makalede, toplam ağırlıklandırılmış ve ağırlıklandırılmamış gecikmeyi minimize etmek için paralel olarak m makinede bir dizi n işin sıralanması

problemini tartışmıştır. Chand ve arkadaşları (2000) nın makalesinde ise Jones' un (1991) modeli genişletilmektedir.

Webster ve Azizoglu (2001) makalelerinde toplam ağırlıklı akış zamanının minimize edilmesi için benzer paralel makinelerdeki hazırlık zamanlarının çizelgelenmesi problemini tartışmıştır. İki dinamik programlama algoritması sunulmuştur. Bunlar: geriye doğru akış algoritması(backward algorithm) ve ileriye doğru akış algoritması (forward algorithm) dır. Mandel ve Mosheiov (2000) makalelerinde paralel benzer makinelerdeki minimum gecikmeyi maksimize etmek amaçlı sıralama problemlerini incelemektedir. İki makineli durumlar için NP-hard yöntemi uygundur. Pseudo -polinomial dinamik programlama algoritması bu durum için ortaya çıkarılacaktır. Sun ve Wang (2003) makalede n işli sıralama problemini, genel bir teslim tarihi ve m benzer paralel makinedeki bekleme cezaları ile tartışmıştır. Çalışmada problemin NP-hard olduğu ve bir dinamik programlama algoritması ile çözüleceği önerilmiştir.

Bank ve Werner (2001) çalışmalarında ilişkisiz paralel makinelerde, toplam ağırlıklandırılmış gecikme ve erken başlama cezalarını minimize etme sıralama problemi için, bir kaç yapısal algoritmalar ile lokal yaklaşım sezgisellerini öne sürmüştür. Laguna ve arkadaşları (2000) ise çalışmalarında dağılım araştırmasına(scatter search) örnek bir yöntem kullanmıştır. Dağılım araştırması popülasyon tabanlı bir metodolojidir. Burada sözde evrimsel metotlarla (so-called evolutionary methods) özellikler paylaştırılır. Wang ve Cheng (2000) problemi çözmek için bir dinamik programlama algoritması öne sürmüştür. Park ve arkadaşları (2000)' nın makalesinde sıralama amacı, ağırlıklandırılan gecikmelerin toplamını minimize eden iş sıralarını bulabilmektir. Çalışmada, Lee (1997) tarafından geliştirilen ATCS (Hazırlıklarla birlikte görünen gecikme maliyetleri) kuralının genişletilmişi önerilmektedir.

He (1999) çalışmasında bilgisayar bağlantılı paralel makine sıralama problemini incelemiştir. Klasik LS (list scheduling-sıra algoritması) algoritması bunun için kullanılmıştır. Weng ve arkadaşları (2001) makalelerinde ağırlıklandırılmış ortalama tamamlanma zamanı amacıyla hazırlık zamanına bağlı sıralama ile ilişkisiz paralel makine sıralama problemini araştırmaktadırlar. Yedi sezgisel algoritma gösterilmektedir. Büyük hesapsal örneklere göre, algoritma 7 nin geri kalan kısımdan daha iyi performans gösterdiği ortaya çıkmıştır.

Gupta ve Ho (2001) iki paralel benzer makinede sıralanan işlerin problemi için, polinomial hesap zamanı içinde problemin tahmini çözümünü veren çoklu değiştirilmiş algoritma (modified multifit algorithm) tanımlanır.

Santos (2002) ise paralel algoritma problemini LogP modeli için tartışmıştır. Optimal toplama ağaç algoritmasının (optimal summing tree algorithm) kullanımıyla, yer değiştiremeyen ve yer değiştirebilen operatörlerin ikisi için örnek toplamda algoritmalar tasarımlanmıştır. Azizoğlu (2002)' nun makalesinde benzer paralel makinelerdeki toplam tamamlanma zamanının minimize edilmesi ile ilgili bir sıralama problemi çalışılmıştır. Bunun için bir polinomial zaman algoritması geliştirilmiştir.

Kellerer ve Strusevich (2002) çalışmalarında m tane paralel tahsis edilen makine için minimize edilen tamamlanma zamanı sıralama problemini, tekil kaynak kısıtları altında tartışmışlardır. Gao ve arkadaşları (1998) kural tabanlı bir sezgiseli N-parti işin M-paralel makinada sıralanması olarak formüle edilmesinde çalışmışlardır. Toledo ve Armentano' nun (2005) makalesinde ilişkisiz paralel makinelerde çok parçalı üretim için Lagrangian relaxation sezgiseli önerilmiştir.

Tahar ve arkadaşları (2005) ise benzer paralel makine sıralama problemi için lineer programlama (LP) teknikleri tabanlı yeni bir metod önermişlerdir. Shabtay ve Kaspi (2005) çalışmalarında benzer paralel makinelerdeki işlerin sıralama problemi için, kaynak atanması ve eş yükleme metodlarından yararlanmışlardır. Silva ve Magalhaes (2006) çalışmalarında ilişkisiz paralel makinelerde parti boyutu sıralaması için bazı sezgiseller önermişler ve bunları karar destek sistemi ile desteklemişlerdir.

Lin ve Liao (2003) ise makalelerinde minimum akış zamanı durumu için tamamlanma zamanı mimizasyonu amaçlı benzer paralel makine probleminde Lexicographic araştırma tabanlı bir algoritmayı ve alt sınır hesabı (the lower bound calculation) ile iş değiştirme kurallarını (the job replacement) kullanmışlardır.

Peng ve Liu (2003) çalışmalarında bulanık proses zamanlarıyla modellenen paralel makine sıralama problemi için bir metodoloji geliştirmiştir. Üç adet yeni bulanık sıralama modeli gösterilmiştir. Bir hibrit intelligent algoritma bütün bu modellerin çözümü için

tasarımlanmıştır. Sonuçta, bazı nümerik örnekler önerilen algoritmanın hesapsal verimliliğini kanıtlamak için gösterilmiştir.

Chen ve Wu (2004) makalelerinde basamak-kabulü metodu (threshold-accepting method), tabu listeleri ve prosedür geliştirme tabanlı, etkili bir sezgiseli toplam gecikmeyi minimize etmesi için önermişlerdir.

Çizelge 2.1 Yapılan Literatür Araştırması

No	Basım Yılı	Yazar	Paralel makine yapısı				Konu	Metodoloji	Yöneylem Araştırması Teknikleri		Problem	Optimize edilecek değişken kısıtları	Maliyet Minimizasyonu						
			Identical	Non-ident.	Related	Unrelated			Sezgisel	Diğerleri			$C_{\max}$	$F_{\max}$	S_t	r	D	T	L
1	2007	Beraldi vd.	+				Rolling-horizon and fix-and-relax heuristics for the parallel machine lot-sizing and scheduling problem with sequence-dependent set-up costs.	Sezgisel yöntemler.	Fix and relax (FR) Rolling horizon (RH) sezgiselleri.	--	Çoklu tamsayılı programlama problemi (MIP).	Toplam hazırlık maliyeti minimizas-yonu.			+				
2	2007	Koulamas ve Kyparisis.	+				An improved delayed-start LPT algorithm for a partition problem on two identical parallel machines.	Sezgisel yöntemler. Dağılım algoritması.	En uzun proses zamanı (LPT) algoritması.	--	Pm//Cmax problemi.	Makine tamamlanma zamanlarının minimizas-yonu.	+						
3	2007	Armentano ve Filho.	+				Minimizing total tardiness in parallel machine scheduling with setup times: An adaptive memory-based GRASP approach.	Meta – Sezgisel.	Hafıza tabanlı GRASP yöntemi.	--	Hazırlık zamanı maliyetlerini içeren makine sıralama problemi.	İş teslim tarihlerinin toplam gecikmesinin minimizas-yonu..						+	

4	2007	Gairing vd.				+	A faster combinatorial approximation algorithm for scheduling unrelated paralel machines.	Sezgisel yöntemler. Dağılım algoritması.	Primal dual yaklaşımı.	--	Maksimum akış problemi.	Tamamlanma zamanı minimizas-yonu.	+	+					
5	2007	Ji ve Cheng.	Paralel makine sıralaması..				Parallel-machine scheduling with simple linear deterioration to minimize total completion time.	Sezgisel yöntemler.	Çoklu polinomial zamanlı dağılım şeması (FPTAS).	--	PTCT (paralel makine toplam tamamlanma zamanı) problemi. NP-hard.	Toplam tamamlanma zamanı minimizas-yonu.	+						
6	2007	Li vd.	Paralel makine sıralaması.				Complexities and algorithms for synchronized scheduling of parallel machine assembly and air transportation in consumer electronics supply chain.	Sezgisel yöntemler	Popülasyon tabanlı benzetimli tavlama (PSAD).	Tamsayılı programlama (ILP). Çoklu tamsayılı programlama (PWESP).	Çoklu tamsayılı programlama (MIP) problemi.	Dağıtım maliyeti ve dağıtım erken bitirme geciktirme cezalarının minimizas-yonu.						+	
7	2007	Nessah vd.	+				An exact method for $P_m / sds, r_i / \sum_{i=1}^{n} C_i$ Problem.	Sezgisel yöntemler	Dal sınır algoritması. En erken tamamlanma zamanı (ECT) kuralı. Toplam akış zamanı (PRTF) kuralına öncelik atama.	--	Hazırlık zamanı ve bırakış zamanına bağlı benzer paralel makine sıralama problemi (IPMSPS).	Toplam tamamlanma zamanı minimizas-yonu.	+	+	+				

8	2007	Gurel ve Akturk.	+				Scheduling parallel CNC machines with time/cost trade-off considerations.	Sezgiseller. Tamsayılı programlama.	SPT (en kısa proses zamanı) kuralı. "Solver" tabanlı yaklaşım (SBA). MPJ (en uygun iş öncelikli) algoritması.	Karmaşık tamsayılı lineer programlama (MINLP).	Bicriteria problemi. Karmaşık tamsayılı lineer programlama (MINLP) model problemi.	Toplam üretim maliyeti ve toplam tamamlanma zamanı minimizas-yonu.	+						
9	2007	Gharbi ve Haouari	+				An approximate decomposition algorithm for scheduling on parallel machines with heads and tails.	Yaklaşık dağılım algoritması (Approximate decomposition algorithm - ADA). Sezgiseller.	Jackson algoritması.	--	Bırakış tarihleri ve dağıtım zamanları ile benzer paralel makine sıralama problemi NP-hard.	Tamamlanma zamanı minimizas-yonu.	+						
10	2007	Koulamas ve Kyparisis	+				A note on the two-stage assembly flow shop scheduling problem with uniform parallel machines.	Sezgiseller. Yaklaşım algoritması (Approximation algorithms).	Modifiye edilmiş "list scheduling" (LS') kuralı. Modifiye edilmiş "reverse list scheduling" (RLS') kuralı.	--	NP-hard. AFQ// Cmax problemi.	Tamamlanma zamanı minimizas-yonu.	+	+					
11	2007	Torres vd.	+				Scheduling uniform parallel machines subject to a secondary resource to minimize the number of tardy jobs.	Sezgisel yaklaşımlar. Tamsayılı programlama.	Moore yaklaşımı. Dal-sınır algoritması. EDD	--	Paralel makine uygunluğu kaynak sıralama (PMFRS)	Geciken iş sayısı minimizas-yonu.						+	

								(en erken teslim tarihi) kuralı.SPT (en kısa proses zamanı) kuralı.		problemi.								
12	2007	Lopes ve Carvalho				+	A branch-and-price algorithm for scheduling parallel machines with sequence dependent setup times.	Dal ve maliyet optimizasyon algoritması. Tamsayılı programlama (IP). Lineer programlama.	Dal-sınır algoritması. Sütun çıkarma algoritması (column generation algorithm); primal box.	--	İlişkisiz paralel makineler üzerinde, bırakış ve teslim tarihlerine bağlı, bir dizi bağımsız işin sıralama problemi.	Düzgün katılan maliyet fonksiyonu minimizas-yonu.			+				
13	2006	Leonardi ve Raz	+				Approximating total flow time on parallel machines.	Dağıtım algoritması. Sezgisel yöntemler.	En düşük sınır. On-line algoritmalar. En kısa kalan proses zamanı (SPRT).	--	Toplam akış zamanı problemi.	Akış zamanı optimizas-yonu.		+					
14	2006	Omar ve Teo	+				Minimizing the sum of earliness/tardiness in identical parallel machines schedule with incompatible job families: An improved MIP approach.	Karmaşık tamsayılı programlama (Mixed integer programming - MIP) Sezgiseller.	En kısa teslim tarihi (EDD).	LINGO.	En kısa teslim tarihi Problemi.	Toplam erken bitirme ve gecikmenin minimizas-yonu.			+		+	+	

15	2006	Rocha vd.				+	Exact algorithms for a scheduling problem with unrelated parallel machines and sequence and machine-dependent setup times.	Meta-sezgiseller.	GRASP. Dal-sınır algortitması (B&B).	CPLEX	İlişkisiz paralel makinelerde, teslim tarihi ve hazırlık zamanına bağlı sıralama problemi.	Teslim tarihi ve hazırlık zamanına bağlı, İlişkisiz paralel makinelerde sıralama problemi.			+		+		
16	2006	Logendran vd.				+	Scheduling unrelated parallel machines with sequence-dependent setups.	Arama algoritması. Sezgiseller.	Tabu arama (TS). Başlangıç çözüm (IS - Initial solution) algoritmaları.	--	İlişkisiz paralel makine sıralama problemi.	İşlerin ağırlıklandırılmış gecikmesinin minimizas-yonu.	+		+			+	
17	2006	Liao ve Sheen	+				Parallel machine scheduling with machine availability and eligibility constraints.	Sezgiseller.	İkili arama (Binary Search). Polinomial ikili arama algoritması.	--	Pm,Ncwin/ pmtn, rj problemi.	Tamamlanma zamanı minimizas-yonu.	+						
18	2006	Neron vd.	+				Search tree based approaches for parallel machine scheduling.	Sezgiseller. Ağaç yaklaşımı (Tree Search).	Dal-sınır algortitması (BnB). Beam search (BS). Branch-and-Greed (BnG). Sınırlı çelişki yaklaşımı (Limited discrepancy search -LDS).	--	NP-hard. R||Cmax problemi.	Tamamlanma zamanı minimizas-yonu.	+						

19	2006	Shim ve Kim	+				A branch and bound algorithm for an identical parallel machine scheduling problem with a job splitting property.	Sezgiseller.	Dal-sınır algortitması (BnB). En düşük sınır.	--	Benzer paralel makine sıralama problemi (IPMSP).	Toplam gecikme ve üretim dizilerinin minimizas-yonu.	+				+		
20	2006	Nessah vd.	+				A branch-and-bound algorithm to minimize total weighted completion time on identical parallel machines with job release dates.	Sezgisel algoritmalar.	Dal-sınır algortitması SRPT (shortest remaining processing time) kuralı.	--	Bırakış tarihlerine bağlı benzer paralel makine sıralama problemi. NP-hard.	Toplam ağırlıklandırılmış tamamlanma zamanı minimizas-yonu.	+			+			
21	2006	Aubry vd.				+	Minimizing setup costs for parallel multi-purpose machines under load-balancing constraint.	Lineer programlama. Tamsayılı programlama.	--	Karmaşık tamsayılı lineer programlama.	NP-hard. Minimum maliyet yük dengesi yapılanış problemi (MCLBCP).	Hazırlık maliyetleri minimizas-yonu.							
22	2006	Attanasio vd.					Auction algorithms for decentralized paralel machine scheduling.	Sezgiseller. Grid scheduling.	Lagrangean sezgiselleri. Lagrangean relaxation.	Tamsayılı lineer programlama.	Grid scheduling problemi.	Tamamlanma zamanı minimizas-yonu.	+						
23	2006	Silva ve Magalhaes				+	Heuristic lot size scheduling on unrelated parallel machines with applications in the textile industry.	Sezgiseller. Karar destek sistemi sıralaması (SDSS).	Pseudo-code method.	--	Ayrık yük boyutu ve sıralama problemi (Discrete lot sizing and scheduling problem -DLSP).	Takım değişimi minimizas-yonu.	+						

24	2006	Shabtay ve Kaspi	+				Parallel machine scheduling with a convex resource consumption function.	Eş yükleme metodu (Equivalent load method). Optimal kaynak dağıtımı. Sezgiseller.	Lagrangian metodu. SPT kuralı. Conway algoritması. Mc Naughton's optimizasyon algoritması. O(n log n) time optimizasyon algoritması.	--	Tamamlanma zamanı problemi. NP-hard.	Tamamlanma zamanı minimizas-yonu.	+			+			
25	2006	Toledo ve Armentano				+	A Lagrangian-based heuristic for the capacitated lot-sizing problem in parallel machines.	Optimal algoritmalar. Sezgiseller.	Lagrangian relaxation. Dal-sınır.	CPLEX.	Kapasiteli yük sıralama problemi. (CLSP). Lagrangian Problemi.	Toplam hazırlık, üretim ve envanter maliyetleri minimizas-yonu.			+				
26	2006	Tahar vd.	+				A linear programming approach for identical parallel machine scheduling with job splitting and sequence-dependent setup times.	Sezgiseller. Lineer programlama.	LP sezgiseli.	CPLEX.	Benzer paralel makine sıralama problemi (PMSP). NP-hard problemi.	Hazırlık zamanına bağlı sıralama. Tamamlanma zamanı minimizas-yonu.	+		+				
27	2006	Xu vd.	Paralel makine sıralaması.				Parallel machine scheduling with almost periodic maintenance and non-preemptive jobs to minimize makespan.	Sezgiseller. Yaklaşım algoritması (Approximation Algoritması).	BFD (Best Fit Decreasing) –LPT (en geniş proses zamanı öncelikli) kuralı.	--	Bin-packing problemi. NP-hard.	Tamamlanma zamanı minimizas-yonu.	+						

28	2006	Angel vd.	+				Truthful algorithms for scheduling selfish tasks on parallel machines.	Sezgiseller. Yaklaşım sıralaması.	"Truthful" algoritması. SPT-LPT kuralı. LDS algoritması.	--	İş sıralama problemi. P//Cmax.	Tamamlanma zamanı minimizas-yonu.	+						
29	2006	Kyparisis ve Koulamas	Uniform Paralel makine sıralaması.				Flexible flow shop scheduling with uniform parallel machines	Sezgiseller. Yaklaşım algoritması (Approximation Algoritması).	LS' kuralı ve the RLS' kuralı. İki-grup (Two -group -TG) sezgiseli.	--	NP-hard. Fm// Cmax problemi. M-kısım esnek akış atölye problemi.	Tamamlanma zamanı minimizas-yonu.	+						
30	2006	Chen ve Wu.				+	Total tardiness minimization on unrelated parallel machine scheduling with auxiliary equipment constraints	Sezgiseller.	Basamak kabulu (Threshold-accepting - TA) metodu. Tabu list. EDD kuralı. SA (benzetimli tavlama) metodu.	Hazırlık zamanına bağlı görünen gecikme (ATCS) prosedürü.	Hazırlık zamanına bağlı görünen gecikme (ATCS) problemi. NP-hard.	Toplam gecikmenin minimizas-yonu.						+	
31	2005	Dunstall ve Wirth.	+				A comparison of branch-and-bound algorithms for a family scheduling problem with identical parallel machines	Sezgiseller.	Dal-sınır (B&B) algoritması. En kısa ağırlıklandırıl-mış proses zamanı (SWPT) dizisi.	--	$P_i/s_i/\sum wC$ problemi.	Ağırlıklan-dırılmış toplam iş tamamlanma zamanı minimizas-yonu.	+		+				
32	2005	Jou	Paralel makine sıralaması.				A genetic algorithm with sub-indexed partitioning genes and its application to	Sezgiseller.	GASP SPT - EDD - SLK kuralı.		Üretim sıralama problemi. NP-hard.	Uygun fonksiyon değişkeni		+				+	

					production scheduling of parallel machines.			MRP sistemi.		minimizas-yonu.								
33	2005	Yang	Paralel makine sıralaması.				The complexity of customer order scheduling problems on parallel machines.	Sezgiseller.	Yığın sıralama (Batch scheduling). EDD kuralı.	--	Müşteri emri sıralama problemi. NP-complete.	Ağırlıklan-dırılmış ortalama yığın tamamlanma zamanı minimizas-yonu.	+						
34	2005	Gupta ve Torres.	+				Generating efficient schedules for identical parallel machines involving flow-time and tardy jobs.	Sezgiseller.	Benzetimli tavlama (SA). En kısa proses zamanı (SPT) kuralı. EDD kuralı.	Akış zamanı tabanlı algoritma.	Toplam akış zamanı ve toplam gecikmiş iş sayılarına bağlı benzer paralel makine sıralama problemi.	Toplam akış zamanı ve gecikmiş iş sayıları minimizas-yonu.		+				+	
35	2005	Dunstall ve Wirth	+				Heuristic methods for the identical parallel machine flowtime problem with set-up times.	Sezgiseller.	Dal ve sınır algoritması. SWPT-SWMPT kuralı. En uzun yığın zamanı (LBT) kuralı. LWST (en uzun ağırlıklan-dırılmış hazırlık zamanı) kuralı.	--	NP-hard. $P/s_i/\sum wC$ problemi.	Hazırlık zamanı minimizas-yonu.			+				
36	2005	Ghirardi ve Potts.				+	Makespan minimization for scheduling unrelated parallel machines: A recovering beam search approach.	Sezgiseller.	Recovering Beam Search (RBS) yaklaşımı.	--	Bütünleşik optimizasyon problemi. R//Cmax	Maksimum tamamlanma zamanı minimizas-	+						

									İkililik tabanlı algoritma (DBA). APPROX.		problemi. NP-hard.	yonu.							
37	2005	Moukrim ve Quilliot	+				Optimal preemptive scheduling on a fixed number of identical parallel machines.	Polinomial zamanlı algoritma. Lineer programlama (LP). Sezgiseller.	Aralık dizileri.	--	Öne almalı sıralama problemi. NP-complete.	Tamamlanma zamanı minimizas-yonu.	+						
38	2005	Low				+	Simulated annealing heuristic for flow shop scheduling problems with unrelated parallel machines.	Sezgiseller. Tamsayılı programlama.	Benzetimli tavlama (SA). Komşuluk yaklaşımı.	Karmaşık tamsayılı programlama.	Çok safhalı akış atölye sıralama problemi.	Toplam akış zamanı minimizas-yonu.		+					
39	2005	Chang vd.	Paralel makine sıralaması.				Two-phase sub population genetic algorithm for parallel machine-scheduling problem.	Sezgiseller.	Genetik algoritma (NSGA-II) . MOGA SPT-EDD kuralı.	--	Çok amaçlı sıralama problemi. NP-hard.	Akış zamanı ve tamamlanma zamanı minimizas-yonu.	+	+					
40	2005	Cao vd.	Paralel makine sıralaması.				Parallel machine selection and job scheduling to minimize machine cost andjob tardiness.	Sezgiseller.	Tabu search. Local Search.	--	NP-hard. Paralel makine seçim ve sıralama model (PMSSM) problemi.	Makine maliyeti ve iş gecikmesinin minimizas-yonu.						+	
41	2004	Kogan	Paralel makine sıralaması.				Optimal scheduling of parallel machines with constrained resources.	Polinomial zamanlı sıralama algoritması.	--	--	Düzgün zamanlı dinamik sıralama problemi.	Minimizing inventory, birikmiş işler ve üretim maliyetleri ile envanter minimizas-yonu.							
42	2004	Torres ve Lopez	Paralel makine sıralaması.				Using the FDH formulation of DEA to evaluate a multi-criteria problem in parallel	Sezgiseller. Lineer programlama.	Benzetimli tavlama (SA). Free Disposal	--	Paralel makine çok kriterli sıralama	Gecikmiş iş sayılarının ve tamamlanma	+					+	

					machine scheduling.		Hull (FDH) formulasyonu.		problemi.	zamanı minimizas-yonu.							
43	2004	Hop Ve Nagarur.		+			The scheduling problem of PCBs for multiple non-identical parallel machines.	Sezgiseller.	Kompozit genetik algoritma.	--	Çok amaçlı problem. PCBs Sıralama problemi.	Toplam tamamlanma zamanı minimizas-yonu.	+					
44	2004	Yang.	+				Scheduling two-component products on parallel machines.	Sezgiseller.	Baker modeli. SPT kuralı. En kısa proses zamanı (SMPT) dizisi.	--	İki fazlı montaj akış atölye problemi.	Ürünlerin toplam tamamlanma zamanı minimizas-yonu.	+		+			
45	2004	Hwang vd.	Paralel makine sıralaması.				Parallel machine scheduling under a grade of service provision.	Sezgiseller. En uzun proses zamanı önceliği (LPT).	En düşük mertebe – en uzun proses zamanı öncelikli (LG-LPT) Metodu. AW algoritması.	--	GoS ile (servisin mertebesi) Paralel makine seçilebilirlik sıralama problemi	Toplam tamamlanma zamanı minimizas-yonu.	+					
46	2004	Peng ve Liu.	Paralel makine sıralaması.				Parallel machine scheduling models with fuzzy processing times.	Sezgiseller.	Genetik algoritma. Hibrit algoritma. Fuzzy programlama.	--	Bulanık paralel makine sıralama problemi (FPMSPs).	Tamamlanma zamanı minimizas-yonu.	+				+	+
47	2004	Koulamas ve Kyparisis.	Uniform Paralel makine sıralaması.				Makespan minimization on uniform parallel machines with release times.	Yaklaşım – sezgisel metodu.	LDT (en geniş dağıtım zamanı) sezgiseli.	--	Minimum tamamlanma zamanı ve bırakış tarihine bağlı üniform paralel makine sıralama problemi.	Tamamlanma zamanı minimizas-yonu.	+					

48	2004	Lin ve Liao	+				Makespan minimization subject to flowtime optimality on identical parallel machines.	Optimal algoritma. Sezgiseller.	Lexicographic yaklaşımı. G&R sezgiseli.	--	Tamamlanma zamanı minimizas-yonu problemi. P//Cmax.	Toplam akış zamanı ve tamamlanma zamanı minimizas-yonu.	+	+					
49	2003	Sun ve Wang.	+				Parallel machine earliness and tardiness scheduling with proportional weights.	Yaklaşım algoritmaları. Sezgiseller.	List scheduling (LS) prosedürü. LPT (longest processing times) sezgiseli.	Dinamik programlama.	NP-complete. WET (weighted earliness and tardiness) problem.	Toplam ağırlıklan-dırılmış erken bitirme ve gecikme minimizas-yonu (WET).					+		
50	2003	Liu ve Wu	+				Scheduling algorithm based on evolutionary computing in identical parallel machine production line.	Sezgiseller.	HEP (hibrit evrimsel programlama algoritması). S-C 3 (deterministik algoritma).	--	Gecikmiş işleri minimize eden benzer paralel makine üretim sırası sıralama problemi.	Gecikmiş iş sayılarının minimizas-yonu.						+	
51	2003	Liaw vd.				+	Scheduling unrelated parallel machines to minimize total weighted tardiness.	Sezgiseller.	Dal-sınır algoritması. Depth-first yaklaşımı.	--	NP-hard. İlişkisiz paralel makinelerde, toplam ağırlıklan-dırılmış gecikme sıralama problemi.	Toplam ağırlıklandırıl-mış gecikme minimizas-yonu.						+	
52	2003	Haouari ve Gharbi.	+				An improved max-flow-based lower bound for minimizing maximum lateness on identical parallel machines.	Sezgiseller.	Öne almalı daha düşük sınır (PLB). Yarı öne almalı daha	--	NP-hard. $P/r_j/L_{\max}$.	Benzer paralel makinelerde maksimum gecikme					+		+

								düşük sınır (SPLB).			minimizas-yonu.							
53	2003	Liao ve Lin.	Uniform Paralel makine sıralaması.				Makespan minimization for two uniform parallel machines.	Sezgiseller.	LPT kuralı. MULTIFIT. TUMO (benzer makine optimal sıralama) algoritması. G&J (Garey ve Johnson) algoritması.	--	NP-hard.	Tamamlanma zamanı minimizas-yonu.	+					
54	2003	Cochran vd.	Paralel makine sıralaması.				A multi-population genetic algorithm to solve multi-objective scheduling problems for parallel machines.	Sezgiseller.	İki fazlı-çoklu populasyon genetik algoritma (MPGA). Çok amaçlı genetik algoritma (MOGA). Vektor değerlendirme genetik algoritma (VEGA).	--	Çok amaçlı sıralama problemi. NP-hard.	Tamamlanma zamanı, toplam ağırlıklandırılmış tamamlanma zamanı ve toplam ağırlıklandırıl-mış gecikme (TWT) minimizas-yonu..	+					+	
55	2003	Azizoğlu	+				Preemptive scheduling on identical parallel machines subject to deadlines.	Sezgiseller. Polinomial zamanlı algoritma.	SPT kuralı. LRPT (Longest remaining processing time) –BA (Backwards among available jobs) kuralı..	--	Benzer paralel makinelerde teslim tarihlerine bağlı toplam tamamlanma zamanı problemi.	Toplam tamamlanma zamanı minimias-yonu.	+						

56	2003	Alagöz ve Azizoğlu	+				Rescheduling of identical parallel machines under machine eligibility constraints.	Sezgiseller.	Dal-sınır algoritması. Polinomial zaman sezgiseli.	Lineer programlama. (LP)	Yeniden sıralama (Rescheduling) problemi. $P\ /M_i, a_j\ /\ \sum_i C_i$. ND (engellenen iş sayıları) problemi.	Ağırlıklandırılmış toplam tamamlanma zamanı ile tamamlanan toplam akış zamanı minimizasyonu.		+					
57	2003	Kellerer ve Strusevich	Paralel makine sıralaması.				Scheduling parallel dedicated machines under a single non-shared resource.	Sezgiseller. Grup teknolojisi (GT) yaklaşımı .	Polinomial zamanlı yaklaşım şeması (PTAS).	Lineer programlama. GT algoritması (Group technology).	Açık atölye tipi sıralama (Open shop scheduling) problemi. NP-hard.	Tamamlanma zamanı minimiasyonu.	+						
58	2002	Santos	Paralel makine sıralaması.				Optimal and Efficient Algorithms for Summing and Prefix Summing on Parallel Machines.	Optimal toplama (Optimal summing) algoritması.	--	Örnek toplama (Prefix Summing) algoritması. LogP modeli.	Örnek toplama (Prefix computation - summing) problemi.	Gecikmenin minimizasyonu.							+
59	2002	Hiraishi vd.	+				Scheduling of parallel identical machines to maximize the weighted number of just-in-time jobs.	Sezgiseller. Polinomial algoritma.	Lann-Mosheiov algoritması.	--	Benzer paralel makineler üzerinde öne almasız sıralama problemi.	Ağırlıklandırlmış erken biten ve geciken işlerin minimizasyonu.			+			+	
60	2002	Mokotoff ve Chretienne				+	A cutting plane algorithm for the unrelated parallel machine scheduling problem.	Yaklaşım algoritmaları. Sezgiseller. Lineer programlama	Dal-sınır (B&B) algoritması.	Karmaşık tamsayılı programlama (MIP). CPLEX.	İlişkisiz paralel makineler üzerinde sıralama Problemi.	Tamamlanma zamanı minimizasyonu.	+						

								(LP).			R//Cmax.								
61	2002	Meyr		+			Simultaneous lotsizing and scheduling on parallel machines.	Meta-sezgisel metod.	GLSPST (The General Lotsizing and Scheduling Problem with sequence dependent Setup Times) çözüm yaklaşımı. Basamak kabulü (TA). Benzetimli tavlama (SA).	--	CHES (Chesapeake Decision Sciences) problemi.	Üretim maliyetleri, envanter tutma maliyetleri ve hazırlık zamanlarına bağlı maliyet minimizas-yonu.			+				
62	2002	Kim vd.				+	Unrelated parallel machine scheduling with setup consideration and a total weighted completion time objective.	Meta-sezgisel metod.	Benzetimli tavlama (SA). Komşuluk yaklaşımı (NS) metodu.	--	İlişkisiz paralel makinelerde hazırlık zamanına bağlı sıralama problemi.	Toplam gecikmenin,. ağırlıklan-dırılmış toplam tamamlanma zamanının minimizas-yonu.	+		+			+	
63	2001	Weng vd.				+	Unrelated parallel machine scheduling with setup consideration and a total weighted completion time objective.	Sezgiseller. Simülasyon.	WSPT (ağırlıklan-dırılmış en kısa proses zamanı) kuralı.	--	İlişkisiz paralel makinelerde, bir dizi bağımsız işin hazırlık zamanına bağlı sıralaması.	Ağırlıklan-dırılmış tamamlanma zamanının minimizas-yonu.	+		+				
64	2001	Gupta ve Ho	+				Minimizing makespan subject to minimum flowtime on two identical parallel machines.	Optimizasyon algoritmaları. Sezgiseller.	Polinomial zamanlı sezgiseller. Multifit.	--	$P2 // F_h$ $(C_{\max} / \sum C_i)$	Tamamlanma zamanı, akış zamanı minimizas-yonu..	+	+					

								SPT kuralı. LPT kuralı.		problemi.								
65	2001	Baptiste ve Timkovsky .	+				On preemption redundancy in scheduling unit processing time jobs on two paralel machines.	Sezgiseller.	Mc Naughton's teoremi.	--	$P//\sum w_j C_j$.	İşlerin ağırlıklan-dırılmış toplam tamamlanma zamanı minimizas-yonu.	+						
66	2001	Webster ve Azizoğlu.	+				Dynamic programming algorithms for scheduling parallel machines with family setup times.	Sezgiseller. Dinamik programlama (DP).	Geriye ve ileri doğru DP (Backward and forward DP) algoritmaları.	--	Benzer paralel makinelerde toplam ağırlıklan-dırılmış akış zamanını minimiza eden sıralama problemi.	Toplam ağırlık-landırılmış akış zamanı minimizas-yonu.		+					
67	2001	Bank ve Werner				+	Heuristic Algorithms for Unrelated Parallel Machine Scheduling with a Common Due Date, Release Dates, and Linear Earliness and Tardiness Penalties.	Yapısal sezgiseller (Constructive heuristics). Meta-sezgiseller.	Lokal arama (Local Search). İki fazlı yapısal (Two-stage constructive) algoritmalar. Tekrarlanan (Iterative) algoritmalar.	--	İlişkisiz paralel makinelerde toplam ağırlıklan-dırılmış gecikme ve erken bitirme cezalarının minimizasyon problemi.	Toplam ağırlıklan-dırılmış gecikme ve erken bitirme cezalarının minimizas-yonu.					+	+	
68	2001	Mosheiov	+				A common due-date assignment problem on parallel identical machines.	Sezgiseller.	Daha alt sınır (Lower bound) sezgiseli. LPT kuralı.	--	Teslim tarihi atama problemi. NP-hard problemi.	Maksimum erken bitirme maliyeti, tmaksimum gecikme maliyeti, ve teslim tarihi maliyeti	+				+	+	

												minimizas-yonu.							
69	2001	Mandel ve Mosheiov	+				Minimizing maximum earliness on parallel identical machines.	Sezgiseller. Pseudo-polynomial dynamic programming.	MST tabanlı sezgisel.	--	Paralel benzer makinelerde maksimum erken bitirme minimizasyon problemi.	Maksimum erken bitirme minimias-yonu (veya minimum gecikme maksimizas-yonu).							+
70	2000	Gupta ve Torres.	+				Minimizing makespan subject to minimum total flow-time on identical parallel machines.	Sezgisel algoritmalar.	LPT – SPT kuralı. Coffman and Sethi'nin Ld ve LI sezgiselleri. UM (Upper bounding and Modified MULTIFIT) Sezgiselleri. MULTIFIT.	--	Hiyerarşik çok kriterli sıralama problemi. NP-hard.	Toplam tamamlanma zamanı minimizas-yonu.	+	+					
71	2000	Chand vd.	+				A model for parallel machine replacement with capacity expansion.	Sezgiseller. Optimal algoritmalar	Jones' modeli. Dinamik programlama algoritması.	--	Kapasite büyüme ile paralel makine yenileme problemi.	Problem döngüsü üzerinde kırılan maliyet toplamı minimizas-yonu.							
72	2000	Laguna vd.	Paralel makine sıralaması.				Minimizing weighted tardiness of jobs with stochastic interruptions in parallel machines.	Sezgiseller.	Dağılım yaklaşımı (Scatter search).	--	Ağırlıklan-dırılmış gecikme problemi.	İşlerin beklenen toplam ağırlıklan-						+	

						Tabu arama. Senaryolar.			dırılmış gecikmesinin minimizas-yonu.							
73	2000	Hall	+			Parallel machine scheduling with a common server.	Sezgiseller.	Polinomial-zaman algoritması. Pseudo - polinomial-zaman algoritması.	--	Öne almasız sıralama problemi. NP-complete.	Ağırlıklan - dırılmış geciken iş sayılarının minimizas-yonu.	+					+
74	2000	Wang ve Cheng	Paralel makine sıralaması.			Parallel machine scheduling with batch delivery costs.	Sezgiseller. Pseudo-polinomial algoritması.	SPT dizisi. List sıralaması. Dinamik programlama.	--	Parti dağıtım maliyeti ile paralel makine sıralama problemi.	Toplam akış zamanı ve dağıtım maliyeti minimizas-yonu.		+				
75	2000	Park vd.	+			Scheduling jobs on parallel machines applying neural network and heuristic rules.	Sezgiseller.	Neural network. ATCS (Apparent Tardiness Cost with Setups) rule.	--	Total weighted tardiness problem.	Minimizing the sum of weighted tardiness.	+				+	
76	2000	Koulamas ve Kyparisis	Uniform Paralel makine sıralaması..			Scheduling on uniform parallel machines to minimize maximum.	Sezgiseller.	EDD (early due date) rule.	--	Üniform paralel makine sıralama problemi.	Maksimum gecikme minimizas-yonu.						+
77	2000	He	+			The Optimal On-Line Parallel machine Scheduling.	Sezgiseller.	On-line Algoritma LS (list scheduling) algoritma.	--	On line paralel makine sıralama problemi. List scheduling problemi.	Tamamlanma zamanı minimizas-yonu. Minimum makine tamamlanma zamanı maksimizas-yonu.	+						

78	1999	Min ve Cheng	+				A genetic algorithm for minimizing the makespan in the case of scheduling identical parallel machines.	Sezgiseller.	Genetik algoritma. Benzetimli tavlama. LPT Metodu.	--	Benzer paralel makine sıralama problemi.	Tamamlanma zamanı minimizas-yonu.	+						
79	1999	Chiu vd.	Paralel makine sıralaması.				Sequencing parallel machining operations by genetic algorithms	Sezgiseller. Diğer yöntemler.	Genetik algoritma.	Karma tamsayılı programlama.	Paralel makine işlem problemi.	Tamamlanma zamanı minimizas-yonu.	+						
80	1999	Chen ve Powell	+				A column generation based decomposition algorithm for a paralel machine just-in-time scheduling problem.	Sezgiseller. Diğer yöntemler.	Dal-sınır algoritması.	Tamsayılı programlama. Dantzig Wolfe dağılımı.	Toplam ağırlıklan-dırılmış gecikme ve erken bitirme problemi.	Toplam ağırlıklan-dırılmış gecikme ve erken bitirme minimizas-yonu.						+	
81	1998	Gao vd.	Parallel injection molding machines.				A production scheduling system for parallel machines in an electrical appliance plant.	Kural tabanlı sezgiseller. Diğer yöntemler.	SPT kuralı.	--	Toplam gecikme minimizasyon problemi.	Toplam gecikme cezalarının minimizas-yonu.						+	
82	1998	Dessouky vd.	+				Scheduling Identical Jobs on Uniform Parallel Machines With Random Processing Times.	Sezgiseller.	Beklenen en erken tamamlanma zamanı (EECT) sıralaması. Dinamik programlama.	--	Beklenen toplam ağırlıklandırıl-mış tamamlanma zamanı ve beklenen genel bir teslim tarihi minimizasyon problemi.	Toplam ağırlıklan-dırılmış tamamlanma zamanı minimizas-yonu. Karşılanan genel bir teslim tarihi maksimizas-yonu.	+				+		

83	1997	Kravchenko ve Werner	+				Parallel Machine Scheduling Problems with a Single Server.	Sezgiseller. Pseudo polinomial algoritması.	List scheduling. Local search ve beam search algoritmaları.	--	NP-hard.	Tamamlanma zamanı minimizas-yonu.	+		+				
84	1997	Centeno ve Armacost	Paralel makine sıralaması.				Parallel machine scheduling with release time and machine eligibility restrictions.	Sezgiseller.	En az esnek iş (LFJ) dağıtım kuralı. En az esnek makine (LFM)kuralı.	--	Maksimum gecikme minimizasyon problemi.	Maksimum gecikme minimizas-yonu.							+
85	1997	Alidaee ve Rosa	+				Scheduling Parallel Machines To Minimize Total Weighted And Unweighted Tardiness.	Sezgiseller.	Modifiye edilmiş teslim tarihi(MDD) algoritması Baker ve Bertrand' ın. LPT kuralı.	--	Toplam ağırlıklan-dırılmış gecikme (TWT) problemi.	Toplam ağırlıklan-dırılmış ve ağırlıklan-dırılmamış gecikme minimizas-yonu.	+					+	
86	1997	Park ve Kim	+				Search Heuristics for a Parallel Machine Scheduling Problem with Ready Times and Due Dates.	Sezgiseller.	Lokal Arama. Benzetimli tavlama. Tabu Arama.	--	Emirlerin toplam maliyet minimizas-yonu problemi.	Emirlerin toplam maliyet minimizas-yonu.					+		
87	1997	Torres vd.	+				Simulated Annealing Heuristics for the Average flow-Time and the Number of Tardy Jobs Bi-Criteria Identical Parallel Machine Problem	Sezgiseller.	Benzetimli tavlama. Komşuluk yaklaşımı. SPT algoritması.	--	Geciken iş sayıları ve ortalama akış zamanı minimizasyonu problemi.	Geciken iş sayıları ve ortalama akış zamanı minimizas-yonu.		+				+	

3. BULANIK MANTIK

Her insan günlük hayatında kesin olarak bilinemeyen, bazen de önceden sanki kesinmiş gibi düşünülen ama sonuçta kesinlik arz etmeyen durumlarla karşılaşır. Bu durumların örgün (sistematik) bir şekilde önceden planlanarak sayısal öngörülerinin yapılması ancak bir takım kabul ve varsayımlardan sonra mümkün olabilmektedir. Şimdiye kadar yapılan mühendislik araştırmalarında ve modellemelerinde bu varsayım, kabul ve kavramlara kesinlik kazandırmak için değişik çalışmalarda bulunulmuştur. Halbuki, büyük ölçeklerden küçük ölçeklere doğru geçildikçe incelenen olayların kesinlikten uzaklaşarak belirsizlikler içeren yönlere doğru gitmeleri söz konusudur. Mesela, çok uzakta bulunan bir cisme bakıldığında bunun nokta şeklinde algılanması onun boyutsuz ve şekilsiz olduğu sonucuna varmamıza sebep olur. Bu cisim bize yaklaştıkça bir boyutludan önce tepsi gibi daha sonra da küre gibi sanki üç boyutlu hale dönüşür. Böylece boyutlar arasında kesin bir geçişten ziyade tedricen bir değişimin olduğuna akıl ile varılabilir (Şen, 2004).

Gerçek dünya karmaşıktır. Bu karmaşıklık genel olarak belirsizlik ve kesin düşünce ve kararlar verilemeyişten kaynaklanır. Birçok sosyal, iktisadi ve teknik konularda insan düşüncelerinin tam anlamı ile olgunlaşmamış oluşundan dolayı belirsizlikler her zaman bulunur. İnsan tarafından geliştirilmiş olan bilgisayarlar, bu türlü belirsizlikleri işlemezler ve çalışmaları için sayısal bilgiler gereklidir. Gerçek bir olayın tam olarak kavranılması insan bilgisinin yetersizliği sonucunda tam anlamı ile mümkün olamadığından insan, düşünce sisteminde ve zihninde bu gibi olayları yaklaşık olarak canlandırarak yorumlarda bulunur. Bilgisayarlardan farklı olarak insanın yaklaşık düşünme ve oldukça yetersiz, eksik ve belirsizlik içeren veri ve bilgi ile işlem yapabilme yeteneği vardır.

Genel olarak, değişik biçimlerde ortaya çıkan karmaşıklık ve belirsizlik gibi tam ve kesin olmayan bilgi kaynaklarına bulanık (fuzzy) kaynaklar adı verilir. Zadeh tarafından gerçek dünya sorunları ne kadar yakından incelemeye alınırsa, çözümün daha da bulanık hale geleceği ifade edilmiştir. Çünkü çok fazla olan bilgi kaynaklarının tümünü insan aynı anda ve etkileşimli olarak kavrayamaz ve bunlardan kesin sonuçlar çıkaramaz. Burada bilgi kaynaklarının temel ve kesin bilgilere ifade olarak, özellikle sözel olan bilgileri de ihtiva ettiği vurgulanmalıdır. İnsan sözel düşünebildiğine ve bildiklerini başkalarına sözel ifadelerle aktarabildiğine göre bu ifadelerin kesin olması beklenemez (Tanyıldızı ve Yazıcıoğlu, 2006).

Bir sistem hakkında ne kadar fazla öğrenerek bilgi sahibi olursak, onu o kadar daha iyi anlayabiliriz ve onun hakkındaki karmaşıklıklar da o derece azalır, fakat tamamen yok olmaz. İncelenen sistemlerin karmaşıklığı, az veya yeterli miktarda veri bulunmazsa bulanıklık, o kadar etkili olacaktır. Bu sistemlerin çözümlerinin araştırılmasında bulanık olan girdi ve çıktı bilgilerden, bulanık mantık kurallarının kullanılması ile anlamlı ve yararlı çözüm çıkarımlarının yapılması yoluna gidilebilir.

1965 yılında Lütfü (Lotfi) Asker Zade (Zadeh) tarafından ortaya atılan bulanık küme, mantık ve sistem kavramları bu araştırıcının uzun yıllar boyunca kontrol altında çalışması; istediği kontrolü elde edebilmesi için fazlaca doğrusal olmayan denklemlerin işin içine girmesi; yöntemin karmaşıklaşması ve çözümün zorlaşması neticesinde ortaya çıkmıştır. Bulanık kavramlarının ortaya atılması ile beraber literatürde bazı hoşnutsuzluklarda olmuştur. Bunlar arasında bazı araştırıcılar bulanıklık fikrini benimseyerek bu konuda çalışmayı teşvik etmişler, ama büyük bir çoğunlukta karşı görüşte olmuşlardır. Bunlar fuzzification yani bulanıklaştırmanın kesin olan bilimsel ilkelere uymadığını hatta bilime karşı geldiğini ileriye sürmüştür. Özellikle, ihtimaller teorisi ve istatistik gibi zaten belirsizliklerle uğraşan bilim dalları bulunduğundan, bu konularda çalışan araştırmacılar, bulanık sistemlere açık bir biçimde karşı çıkmışlardır. Bulanık yöntemlerin yapacağı her türlü hesaplamanın, ihtimal ve istatistik hesaplamalarla yapılabileceğini ileri sürmüşlerdir. Hatta bu yöntemlerin bulanık sistemlerden daha iyi sonuçlar verdiğini iddia etmişlerdir (Tanyıldızı ve Yazıcıoğlu, 2006).

İlk çıktığı zamanlarda, bulanık sistemlerin doğrudan uygulaması olmadığından, yapılan tartışmalar daha ziyade felsefik seviyede kalmış ve bunun sonucunda kuvvetli ve teorik temelleri olan ihtimaller teorisi ve istatistik yöntemleri ağır basmıştır. Ancak burada gözden kaçırılan basit bir nokta, sözel bilgilerin bulunması halinde istatistiğin fazlaca işe yaramadığıdır. Her ne kadar Bayesian teorisi gibi istatistik yöntem ile sözel bazı ifadelerin hesaplamalarda kullanılması mümkün ise de, bu yöntemlerin işleyişlerinde bazı temel kabuller (normal olarak dağılmış olmak, doğrusal olmak gibi) pratikte gerçekleşmemektedir. Bu sebeplerden, bulanık (fuzzy) sistemler dünyadaki hemen her araştırma merkezinde fazlaca rağbet görmemiştir. Özellikle de, Batı'da bu kavramlar nerede ise tamamen ihmal edilmiş, hoş karşılanmamıştır.

Bulanık kavram ve sistemlerin dünyanın değişik araştırma merkezlerinde dikkat kazanması 1975 yılında Mamdani ve Assilian tarafından yapılan gerçek bir kontrol uygulaması ile

olmuştur. Bu araştırıcılar ilk defa bir buhar makinesi kontrolünün bulanık sistem ile modellemesini başarmıştır. Bu ön çalışmadan, bulanık sistemlerle çalışmanın ne kadar kolay ama sonuçlarının da ne kadar etkili olduğu anlaşılmıştır (Baykal ve Beyan, 2004).

Daha sonraki yıllarda bulanık sistem uygulaması bir çimento fabrikasının işletilmesi ve kontrolü için yapılınca, artık bulanık kavramlar dünyanın birçok yerinde yavaş yavaş kullanılmaya başlanmıştır. Bu faaliyet, Batı'da çok yavaş olurken, Doğu'da ve özellikle Japonya, Singapur, Kore ve Malezya'da kendisini fazlaca göstermiştir. Müteakip yıllarda, bilhassa 1980'lerden sonra bulanık sistemin elektrikli süpürgeler, çamaşır makineleri, asansörler, metro ve şirket işletimi gibi konularda bulanık mantık kullanımında fazlasıyla artış olmuştur. Son yıllarda, birçok mühendislik dallarında, veri tabanlarının sözelleştirilmesinde ve birçok konularda kullanılır hale gelmiştir (Şen, 2004).

Genel olarak mühendislikte kontrol problemlerinde kullanılan bulanık mantık, tedarik zinciri değerlendirme aralığı problemlerinin de çözümünde oldukça kullanışlıdır (Shore vd., 2003).

3.1 Belirsizlik ve Kesin Olmayış

Mantık, sistem ve küme vb. için bulanıklık, belirsizliğin bir ifadesi olarak karşımıza çıkar. Geçmişte, belirsizliklerin işlenmesi ve anlamlı sonuçlara varılabilmesi için ihtimaller teorisi kullanılmıştır. Matematik ve mühendislikte bu teori belirsizlik durumlarında istatistik yöntemlerle beraber kullanılır. Bu nedenle de, bütün belirsizliklerin rasgele karakterde olduğu kavramı yaygınlaşmıştır (Dubois vd., 2003)

Rasgeleliğin en önemli özelliği, sonuçların ortaya çıkmasında tamamen şans olayının rol oynaması ve gerekli öngörülerin ve tahminlerin kesin bir doğrulukta önceden yapılamamasıdır. Ancak, bilinen belirsizliklerin hepsi rasgele karakterde değildir. Günlük hayatta karşılaşılan belirsizliklerin çoğunun rasgele olmadığı kolayca anlaşılabilir. Rasgele karakterde olmayan olayların, örneğin, sözel belirsizlikler halinde inceleme ve sonuç çıkarma işlemlerinde ihtimaller hesabı ve istatistik gibi sayısal belirsizlikleri gerektiren yöntembilimler kullanılmaz (Şen, 2004).

Etrafımızda ilgi çeken birçok sorunu, sayısal bilgilerden ziyade, çok kere görüş, eğer yargısı, takdir ve düşüncelerimizi sözel olarak ifade ederek inceler ve yorumlarız. Bu ifadelerin anlamlı olmaları ve başkalarına iletilebilmesi için mutlaka her insanın en az bir tane dile

ihtiyacı vardır. Dil ne kadar kesin olmayan bir kelime ve cümleleri ihtiva etse bile, insanın iletişim kurmasında ve bilgi akışında en etkin olan bir vasıtadır. Dildeki belirsizliklere rağmen, insanoğlu onunla birbirini kolayca anlayabilmektedir. Örneğin, "hava sıcak" denildiğinde herkes, kesin olarak hava kelimesinin anlamını anlamaktadır. Ancak, "sıcak" kelimesinin ifade ettiği anlam izafi olarak birbirinden farklı olabilir. Kutuplarda bulunan bir kişinin sıcak için 15 dereceyi anlamasına karşılık, ekvator civarındaki bir kişi için bu 35 dereceyi bulabilir. Arada birçok kişinin görüşü olarak başka dereceler de bulunur. Böylece sıcak kelimesinin altında insanların da ima ettiği sayısal anlayışın bir sonucu olarak belirsiz bir durum vardır. Bu rasgele değildir, ancak belirsizdir ve bu şekilde kelimelerin ima etikleri belirsizliklere fuzzy (bulanıklık) denir. Burada dikkat edilmesi gereken sıcak kelimesinin ne kadar fazla bir sayısal dereceler topluluğunu temsil ettiğidir, bu topluluklara bulanık küme adı verilir. Bazı insanların sıcaklığı 15 derece, bazılarının ise 35 derece gibi oldukça farklı sayısal biçimde algılamasına karşılık bu insanlar arasında ihtilaf bulunmaz. Ancak Aristo mantığında kesin olarak sıcak veya soğuktan biri vardır. Böylece, bulanık mantığın sayılardan ziyade sözel kelimeleri esas aldığı anlaşılmış olur.

Bulanık mantığın en geçerli olduğu iki durumdan ilki, incelenen olayın çok karmaşık olması ve bununla ilgili yeterli bilginin bulunmaması durumunda kişilerin görüş ve değer yargılarına yer verilmesi, ikincisi ise insan muhakemesine, kavrayışlarına ve karar vermesine ihtiyaç gösteren hallerdir. Bulanık mantıktan, karşılaşılan her türlü sorunun karmaşık da olsa çözülebileceği anlamı çıkarılmamalıdır (Kıyak ve Kahvecioğlu, 2003).

3.2 Bulanık Kümeler ve Üyelik Dereceleri

Aristo mantığına göre çalışan ve şimdiye kadar alışılagelen klasik küme kavramında, bir kümeye giren öğelerin oraya ait oluşları durumunda üyelik dereceleri 1'e, ait olmamaları durumunda ise 0'a eşit var sayılmıştır. İkisi arasında hiçbir üyelik derecesi düşünülemez. Halbuki bulanık kümeler kavramında 0 ile 1 arasında değişen, değişik üyelik derecelerinden söz etmek mümkündür. Böylece daha şimdiden bulanık kümelerdeki öğelerin üyelik derecelerinin kesintisiz olarak 0 ile 1 arasında değer aldığından söz edebiliriz. Aslında Zadeh küme öğelerinin üyelik derecelerinin 0 ile 1 arasında değişebileceğini ileriye sürerek kümeler teorisinde geniş uygulamaya sahip ve doğal hayatla uyumlu olan bulanık kümeler kavramının özellikle 1980 yılı sonrasındaki teknolojik ve bilimsel çalışmalarda etkisi büyük olmuştur. Bu şekilde tanımlanan üyelik derecelerinin her bir bulanık söz için üç temel özelliği sağlaması tanım olarak gerekmektedir. Bunlar şöyle sıralanabilir:

1. Bulanık kümenin normal olmasıdır ki, bunun için en azından o kümede bulunan öğelerden bir tanesinin en büyük üyelik derecesi olan 1'e sahip bulunması gerekliliğidir.
2. Bulanık kümenin monoton olması istenir ki, bunun anlamı üyelik derecesi 1'e eşit olan öğeye yakın sağda ve soldaki öğelerin üyelik derecelerinin de 1'e yakın olmasıdır.
3. Üyelik derecesi 1'e eşit olan öğeden sağa veya sola eşit mesafede hareket edildiği zaman bulunan öğelerin üyelik derecelerinin birbirine eşit olmasıdır ki, buna da bulanık kümenin simetri özelliği adı verilir.

Klasik kümelerle bulanık kümelerin arasındaki önemli farklardan bir tanesi, klasik kümelerin sadece bir tane dikdörtgen üyelik derecesi fonksiyonu bulunmasına karşılık, bulanık kümenin yukarıdaki üç şarttan ilk ikisini mutlaka sağlayacak şekilde değişik üyelik derecesi fonksiyonlarına sahip olmasıdır.

Yukarıda açıklananlardan sonra, bulanık küme üyelik derecesi fonksiyonlarının mutlaka simetrik olması özelliğini sağlamasının gerekmediği şimdiden akılda tutulmalıdır (Tanyıldızı ve Yazıcıoğlu, 2006).

3.3 Bulanık Sistem

İngilizcede "fuzzy" kelimesine karşı gelen Türkçedeki "bulanık" kelimesinin genel olarak puslu, dumanlı, kesinlikle ayırt edilemeyen, kesin olmayan, belirsiz, kafa karıştıran gibi bir dizi anlamı vardır. Bulanıklığın anlamı, bir araştırıcının incelediği konunun kendi tarafından tam kesinlikle bilinmemesi durumunda sahip olduğu eksik ve belirsiz bilgilerin tümüdür. Böylece araştırıcı, klasik analitik yöntemler ile dinamik ve korunum ilkelerinin elde ettiği denklemleri, verilerinde ve bilgilerinde belirsizlik yani bulanıklık bulunduğu için doğrudan kullanılamaz. Araştırıcının incelediği olay veya mekanizma sadece kesin kurallı, çıkarımlarında kabul ve varsayımlar olan denklemler yerine, onların tamamlayıcısı olarak mevcut onunla ilgili sözel ve oldukça belirsiz bilgiler de göz önünde tutularak modellenebilir. Bulanık ilkelerin yardımı ile olayların incelenmesinde veri ve bilgi bakımından bir bulanıklık söz konusu ise de, bulanık yöntemlerin işleyişi tamamen belirgindir. Araştırıcıların bulanık sistemleri kullanması için genel olarak iki sebep vardır. Bunlar şöyle ifade edilebilir (Şen, 2004):

1. Gerçek dünya olaylarının çok karmaşık olması dolayısıyla bu olayların belirgin denklemlerle tanımlanarak, kesin bir şekilde kontrol altına alınması mümkün olmaz.

Bunun doğal sonucu olarak araştırıcı, kesin olmasa bile yaklaşık fakat çözülebilirliği olan yöntemlere başvurmayı her zaman tercih eder. O halde, yapılan bütün çalışmalarda çözümler bir dereceye kadar yaklaşıktır. Aksi takdirde, çok sayıda doğrusal olmayan denklemlerin aynı zamanlı olarak çözülmesi gerekir ki, bunun günümüz bilgilerine göre belirgin olmayan kaotik çözümlere yol açacağı bilinmektedir.

2. Mühendislikte bütün teori ve denklemler gerçek dünyayı yaklaşık bir şekilde ifade eder. Birçok gerçek sistem doğrusal olmamasına, nonlineer olmasına rağmen bunların klasik yöntemlerle incelenmesinde doğrusallığı kabul etmek için her türlü gayret sarf edilir. Örneğin, mukavemet hesaplarında malzemenin gerilme altında şekil değiştirmesinin doğal olduğu, Hooke kanunu ile kesin bir ifadeyle kavuşturulmuştur. Halbuki malzemenin her zaman bu şekilde davranması beklenemez ve bu sebeple küçük de olsa bazı sapmaların olması muhtemeldir. Zaten bunun doğal sonucu olarak, mukavemet boyutlandırmalarında emniyet katsayısı gibi bir büyüklük hesaplara ithal edilerek, olabilecek belirsizlikler yine belirgin bir şekilde göz önünde tutulmuştur. Emniyet katsayısının kullanılması, bir bakıma, belirsizliklerin arka kapıdan çözümün içine katı bir şekilde sokulmasıdır. Halbuki gerçek malzemenin davranışlarında emniyet katsayısı gibi bir büyüklüğe gerek kalmadan boyutlandırma yapılması için belirsizlik ilkelerine gerek duyulur .

Çizelge 3.1 Klasik Mantık ile bulanık mantık arasındaki farklar

Klasik Mantık	**Bulanık Mantık**
A veya A Değil	A ve A Değil
Kesin	Kısmi
Hepsi veya Hiçbiri	Belirli Derecelerde
0 veya 1	0 ve 1 Arasında Süreklilik
İkili Birimler	Bulanık Birimler

Bulanık sistemlerle ilgili örneklerden yaygın olanı, bir kişinin araba sürmesini öğrenmesinde ortaya çıkan sözel bilgilerdir. Sürücü adayına hız şu kadar km.ye varınca gaza şu kadar miktar bas denilecek yerde, eğitim sırasında

"EĞER hız düşük **İSE** gaza fazlaca bas"

veya

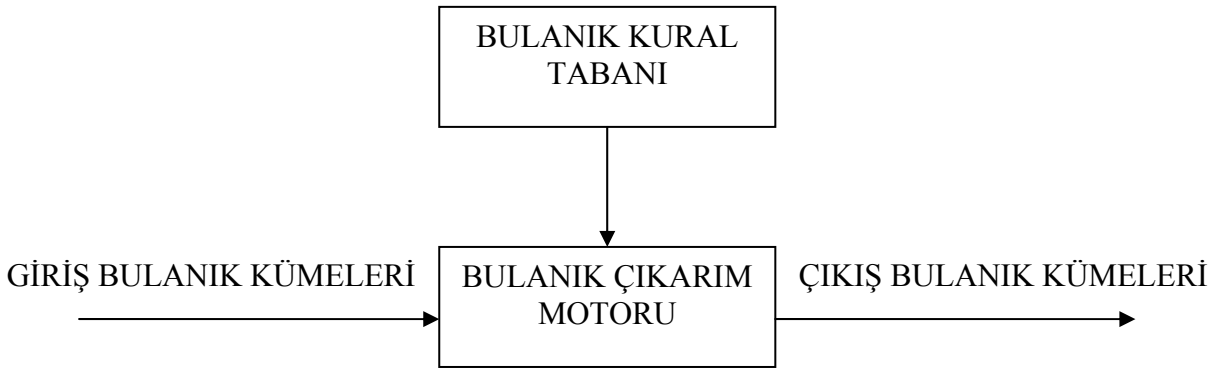
"EĞER hız yüksek **İSE** gaza az bas"

gibi kurallar söylenir. Bu kurallardaki düşük, fazlaca, yüksek ve az kelimeleri kişilerde ister istemez belirli bir aralıkta sayısal değerleri ima eder. İşte bu ima edilen değerler topluluğuna o kelimeyi temsil eden küme denir. Bu kümenin her öğesi aynı derecede önemli değildir. **EĞER-İSE** şeklinde kuralların **EĞER** ile **İSE** kelimeleri arasında kalan kısımlara **öncül kısım** ve **İSE** kelimesinden sonra olan kısma da **soncul kısım** veya **kural çıkarımı** adı verilir. Genel olarak, öncül kısımda olayla ilgili koşulları içeren deyişler vardır. Soncul kısım ise daha ziyade kontrolle ilgilidir.

Şimdiye kadar öğrenilen matematik, stokastik veya kavramsal sistemlerin hemen hepsi Şekil 3.1'de verilen üç ayrı birimden ibarettir.

Şekil 3.1 Klasik Sistem (Muşdal, 2005)

Bunlar; giriş, bu girişi çıkışa dönüştüren ve sistem davranışı denilen bir kutu ve buradan çıkış kısımlarıdır. Buradaki birimlerin hepsinde sayısal veri, çıkış veya işlemler yapılmaktadır. Bulanık sistemlerin bu klasik tasarımdan farkı, sistem davranışı kısmının ikiye ayrılarak Şekil 3.2'de gösterildiği gibi kendi aralarında bağlantılı dört birimin olmasıdır.



Şekil 3.2 Genel Bulanık Sistem (Şen, 2004)

Burada bulunan birimlerin her birinin farklı, fakat birbiri ile ilişkili olabilen aşağıdaki görevleri vardır.

1. **Genel bilgi tabanı birimi:** İncelenecek olayın maruz kaldığı girdi değişkenlerini ve bunlar hakkındaki tüm bilgileri içerir. Buna veri tabanı veya kısaca giriş adı da verilir. Genel veri tabanı denmesinin sebebi buradaki bilgilerin sayısal ve/veya sözel

olabilmesidir.

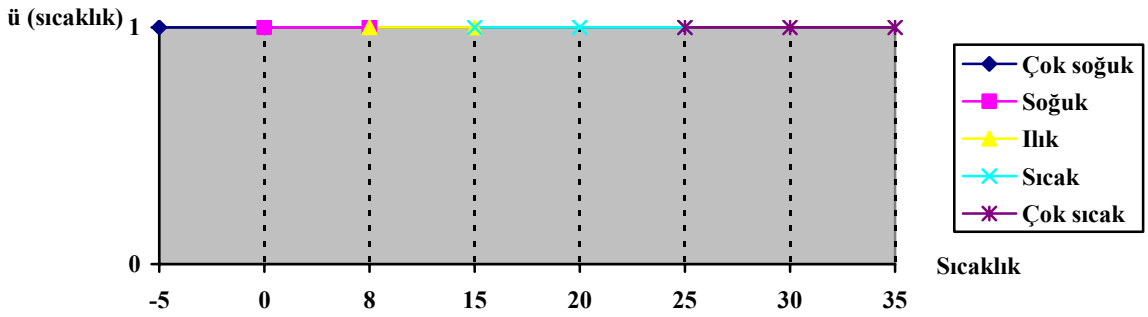
2. **Bulanık kural tabanı birimi:** Veri tabanındaki girişleri çıkış değişkenlerine bağlayan, mantıksal **EĞER-İSE** türünde yazılabilen kuralların tümünü içerir. Bu kuralların yazılmasında sadece girdi verileri ile çıktılar arasında olabilecek tüm bulanık küme bağlantıları düşünülür. Böylece, her bir kural bir veya daha fazla girdi uzayını bir çıktı uzayına mantıksal olarak bağlar. İşte bu bağlamların tümü kural tabanını oluşturur.
3. **Bulanık çıkarım motoru birimi:** bulanık kural tabanında giriş ve çıkış bulanık kümeleri arasında kurulmuş olan ilişkilerin hepsini bir araya toplayarak, sistemin bir çıkışlı davranmasını temin eden işlemler topluluğunu içeren bir mekanizmadır. Bu motor, her bir kuralın çıkarımlarını bir araya toplayarak tüm sistemin girdiler altında nasıl bir çıktı vereceğinin belirlenmesine yarar.
4. **Çıktı birimi:** Bilgi ve bulanık kural tabanlarının, bulanık çıkarım motoru vasıtası ile etkileşimi sonunda elde edilen çıktı değerlerinin topluluğunu belirtir.

Burada dikkat edilmesi gereken bir nokta genel olarak veri tabanındaki bilgilerin ve çıktıların bulanık değerler olmasıdır. Yani her birim tamamen bulanık kümelerden oluşmaktadır (Baykal ve Beyan, 2004).

3.4 Üyelik Fonksiyonları

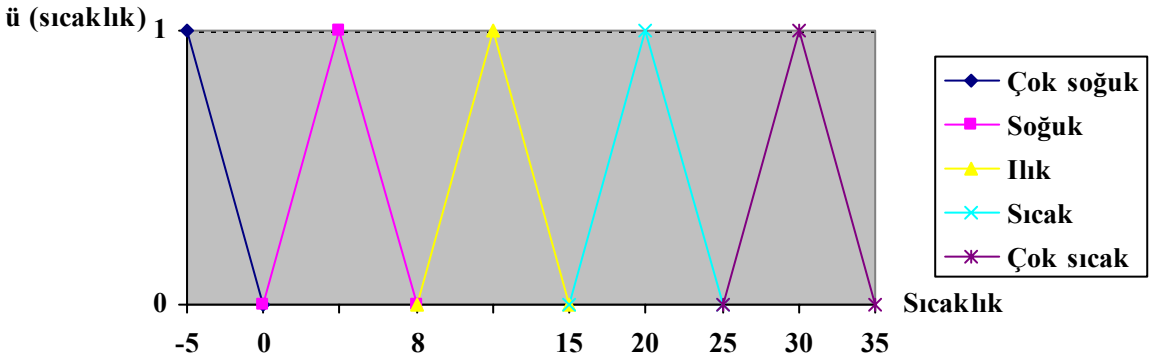
Üyelik fonksiyonu sayısal bir aralığın, bir kelime ile ifade edilen bir küme olan üyeliğini ifade eder. Göz önünde tutulan bir bulanık küme veya ifadenin temsil ettiği sayısal aralık, o ifade hakkında bilgi sahibi olan kişiler tarafından belirlenebilir (Baykal ve Beyan, 2004).

Mesela, İstanbul'da sıcaklık derecesinin değişim aralığının aşağı yukarı -5 dereceden +35 dereceye kadar olduğu söylenebilir. İşte bu aralık sıcaklık kümesinin İstanbul için öğelerin bulunabileceği aralığı belirtir. Böylece tüm sıcaklık uzayı belirlenmiştir. Ancak, günlük konuşmalarda bu sıcaklık uzayının da bir takım alt aralıklardan oluştuğu düşünülür. Mesela, "çok soğuk", "ılık", "sıcak", "aşırı sıcak" gibi. Mesela çok soğuğun -5 derece ile 0 derece, soğuğun 0 derece ile 8 derece, ılığın 8 derece ile 15 derece, sıcağın 15 derece ile 25 derece, çok sıcağın 25 dereceden başladığı söylenebilir. Burada dikkat edilirse aralık tahminlerinde bulunulmuş ve her bir alt aralıktan biri bitince diğeri başlamıştır (Şekil 3.3).



Şekil 3.3 Bitişik Dikdörtgen Gösterim (Şen, 2004)

Bu aralıkların sınırlarında yine Aristo mantığına göre katı kararlar alınmalıdır. Örneğin, 7.9 derecenin soğuk, 8.1 derecenin ise ılık olduğuna karar verilir. Bu şekilde gösterim bakımından önemli bir nokta, her alt aralığa düşen sıcaklık değerinin üyelik derecesinin, sadece o aralıkta 1'e, diğer aralıklarda ise 0'a eşit olduğudur. Bu nedenle her sıcaklık alt kelimesinin üyelik fonksiyonu yüksekliği 1'e eşit olan bir dikdörtgen şeklindedir.

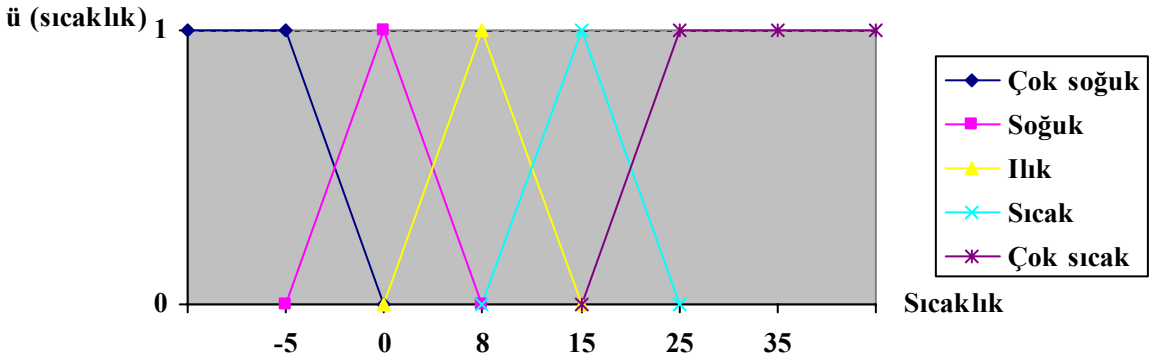


Şekil 3.4 Bitişik Üçgen Gösterim (Şen, 2004)

Şekil 3.4'de yukarıdaki tartışmanın bir doğal sonucu olarak en basit üçgen üyelik fonksiyonları bitişik olarak alınmıştır. Bu üçgenlerin de sıcaklık alt kümelerini tam yansıtmadığı açıktır. Çünkü burada da sınırlardaki sıcaklık değerlerinin üyelik dereceleri sıfır olarak düşünülmüştür. Ayrıca, bu sınır değerleri ne alttaki ne de üstteki sıcaklık alt kümelerine dahildir. Böylece, sınır değerler için tam anlamı ile bir belirsizlik vardır. Diğer taraftan, bu şekildeki alt aralıklar halen Aristo mantığına göre işlem görür. Çünkü bir alt

aralığa düşen sıcaklık değeri, sadece o alt aralığa aittir. Fakat, Şekil 3.3'den farklı olarak üyelik derecesi 1'e eşit değildir.

Biraz daha makul düşünen birisi, bu aralıkların arasındaki geçiş kısımlarının böyle birbirinin devamı olmayacağını ve bir örtüşmenin söz konusu olabileceğini söylerse, daha mantıklı, günlük hayatta geçerli ve uzlaştırıcı çözümlere gitmiş olur. Çünkü herkesin ılık sınırlarının +8 ile +15 derecede sıfır üyelik derecelerine sahip olacağını kabul etmesini savunmak mümkün değildir. Halbuki, günlük hayatta sınıra yakın olan değerlerin hangi aralığa düşeceği oldukça müphem ve şüpheli, yani bulanıktır. Böylece, sıcaklık alt aralıklarının birbiri ile örtüşmeli geçişlere sahip olmasının gerekliliği ile sonuçta Şekil 3.5'de verilen üyelik fonksiyonlarına varılır (Tanyıldızı ve Yazıcıoğlu, 2006).



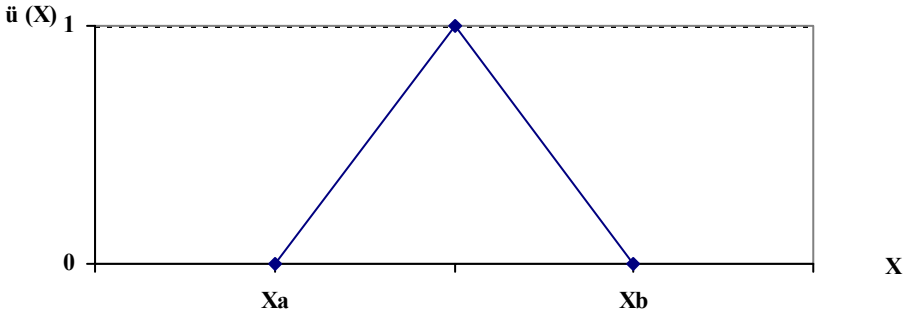
Şekil 3.5 Örtüşmeli Üçgen Gösterim (Muşdal, 2005)

Yukarıda söylenenlerden sonra ilk ve son alt aralıktaki sıcaklık durumlarının "çok çok soğuğa" veya "çok çok sıcağa" doğru giderken başka alt aralıklar olmadığından, üyelik derecelerinin 1'e eşit kalmasının makul olacağı anlaşılır. Bunun doğal bir sonucu olarak da, ilk ve son üyelik fonksiyonlarının üçgen değil de yamuk şeklinde olacağı sonucuna varılır. Böylece, her alt aralığa girişimli olarak bir üyelik fonksiyonu şekli tayin edilmiştir.

Diğer taraftan, sorun her alt aralığa, örneğin "ılık" aralığına düşen sıcaklık derecelerinin hepsinin aynı dönemde olup olmayacağıdır. Tabii olarak, ılık aralığının alt ve üst uçlarına yaklaştıkça onun komşusu olan altta soğuk, üstte ise sıcak alt kümelerine doğru geçişler beklendiği için, o geçiş bölgelerine rastlayan kısımların tam anlamıyla ılık vasfına sahip olacağı söylenemez. Böylece her bir aralığa düşen sıcaklık derecelerinin, o alt aralığın

uçlarına yakın kısımlarında önemlerini ortaya kıyasla göreceli olarak kaybedeceği sonucuna, buradan da eğer bir alt aralıkta önem derecesi diye bir değer düşünülecek olursa bunun en büyük değerlerinin o alt aralığın ortalarında, en düşük değerlerinin ise uçlarda olacağını söyleyebiliriz. Bu düşünceler bizi Şekil 3.6'da gösterilen bir geometrik gösterime sürükler.

Genel olarak, her alt aralığın ayrık üyelik fonksiyonu bu şekilde gösterildiği gibi olur. Bu fonksiyonların simetrik olması gerekmez. Böylece X_a ve X_b gibi alt ve üst sınırlara sahip X değişkeninin bu aralıktaki her değerine ayrı bir üyelik derecesi, $ü(X)$, atanmış olur. Bu aralıktaki tüm X değerleri, o X değişkeninin bir alt kümesini teşkil eder
(Baykal ve Beyan, 2004).



Şekil 3.6 Bulanık Küme (Balasubramanian ve Grossmann, 2003)

Genel olarak küme üyelerinin değerleri ile değişiklik gösteren böyle bir eğriye üyelik fonksiyonu (önem eğrisi) adı verilir. Bunun en önemli özellikleri, alt küme sınırlarındaki değerlerinin orta öğelerinkine göre daha düşük olmasıdır. Ancak klasik kümelere bir benzerlik teşkil etmesi açısından en büyük önem derecesine sahip olan ortaya yakın öğelere 1 değeri atanırsa, diğerlerinin 0 ile 1 arasında ondalıklı ve sürekli değiştiği sonucuna varılır. İşte, 0 ile 1 arasındaki değişimin, her bir öğe için değerine üyelik derecesi $(ü)$, bunun bir alt küme içindeki değişimine ise üyelik fonksiyonu adı verilir. Böylece, üyelik fonksiyonu şemsiyesi altında toplanan öğeler önem derecelerine göre birer üyelik derecesine sahiptir (Şen, 2004).

X evrensel bir küme olsun. A kümesini tanımlayan üyelik fonksiyonu

$$ü_A : x \rightarrow [0,1] \qquad (3.1)$$

şeklinde tanımlanır. Buna bağlı olarak da bulanık A kümesinin tanımı

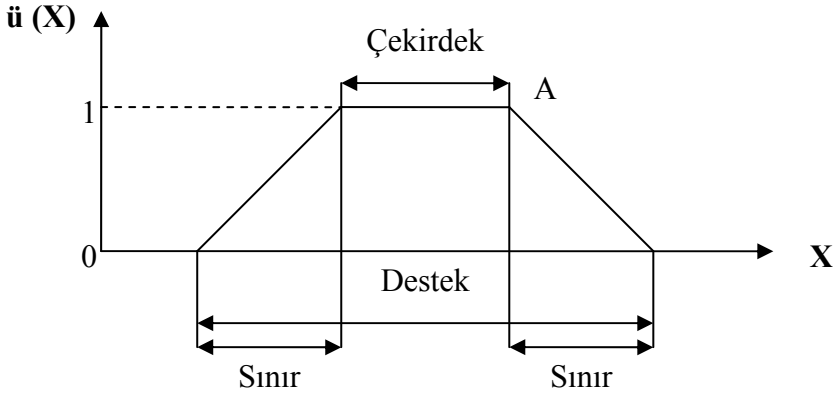
$A = \{(x, \ddot{u}_A(x) \mid x \in X)\}$ (3.2)

şeklindedir.

Matematik kurallarına uygun olarak düzgün şekilli üyelik fonksiyonları, Şekil 3.6'da gösterilen üçgenden başka, yamuk veya çan eğrisi şeklinde de olabilir. Pratik uygulamalarda bunlardan en fazla üçgen olanı kullanılır (Baykal ve Beyan, 2004)..

3.4.1 Üyelik Fonksiyonunun Kısımları

Herhangi bir A bulanık kümesi için üyelik fonksiyonunun çekirdeği (Şekil 3.7), A kümesi içinde üyelik derecesi 1'e eşit olan elemanların kümesidir.



Şekil 3.7 Bir Bulanık Kümenin Çekirdek, Destek ve Sınır Kısımları (Öndemir, 2004)

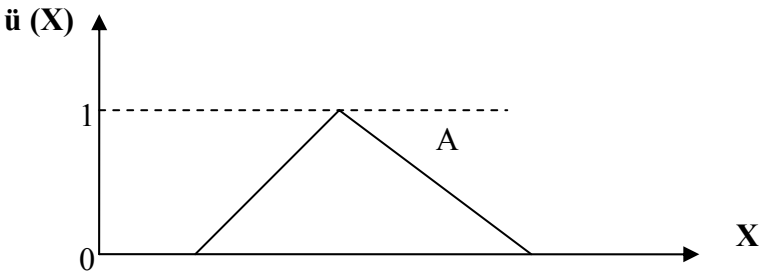
Herhangi bir A bulanık kümesi için üyelik fonksiyonunun destek kısmı (Şekil 3.7), A kümesi içindeki sıfır olmayan üyelik değerini karakterize eden evrensel küme olarak tanımlanır ve

$\text{Supp } A = \{x \in X \mid \ddot{u}_A(x) > 0\}$ (3.3)

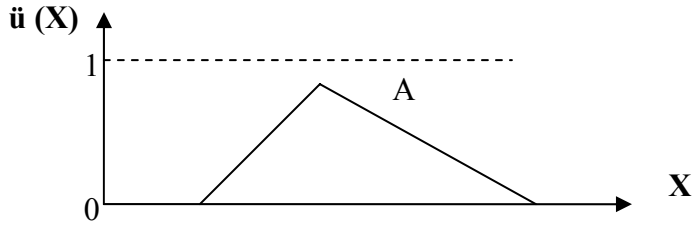
ile gösterilir (Yager ve Filev, 1994).

Herhangi bir A bulanık kümesi için üyelik fonksiyonunu sınırları (Şekil 3.7), A kümesi içindeki sıfır ve tam üye olmayan değerler ile karakterize edilen evrensel küme olarak tanımlanır. Bu küme içindeki X elemanlarının üyelik dereceleri 0 ile 1 arasındadır.

Normal bulanık küme üyelik fonksiyonu üyelik değeri 1 olan en az bir tane x elemanına sahiptir (Şekil 3.8).

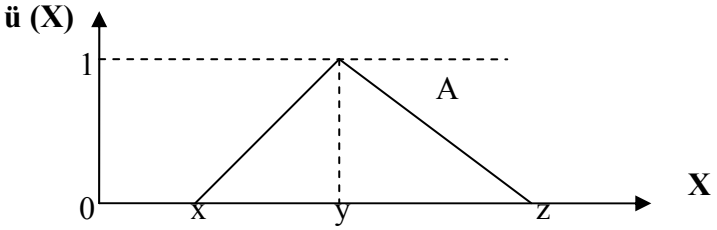


Şekil 3.8 Normal Bulanık Küme (Balasubramanian ve Grossmann, 2003)



Şekil 3.9 Subnormal Bulanık Küme (Balasubramanian ve Grossmann, 2003)

A bulanık kümesinin yüksekliği üyelik fonksiyonunun maksimum değeridir. Max $\{\ddot{u}_A(x)\}$ ile gösterilir. Eğer bu değer 1'den küçük ise bu bulanık küme subnormal bulanık kümedir (Şekil 3.9).



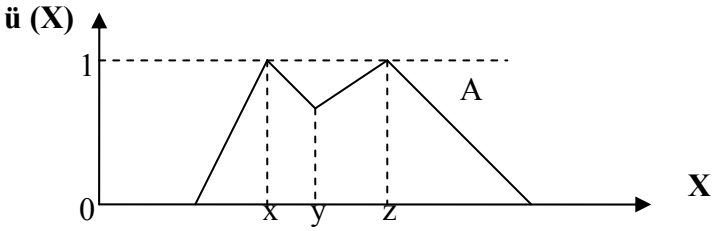
Şekil 3.10 Konveks Normal Bulanık Küme (Balasubramanian ve Grossmann, 2003)

Konveks bulanık küme, üyelik fonksiyonu monoton şekilde artan veya azalan, ya da önce monoton bir şekilde artan değerler ile monoton bir şekilde azalan kümeler olarak açıklanır.
Bir başka deyişle, eğer, herhangi x<y<z elemanları için;

$$\ddot{u}_A(y) \geq \text{Min}\,[\ddot{u}_A(x), \ddot{u}_A(z)] \qquad (3.4)$$

denklemi sağlanıyorsa bu bulanık küme, A, konveks bulanık kümedir (Şekil 3.10).

Her hangi iki konveks kümenin kesişimi de konveks bulanık kümedir. Üyelik fonksiyonunda üyelik değerleri 0.5 olan noktalar fonksiyonun geçiş (cros-over) noktaları olarak tanımlanır (Yager ve Filev, 1994).



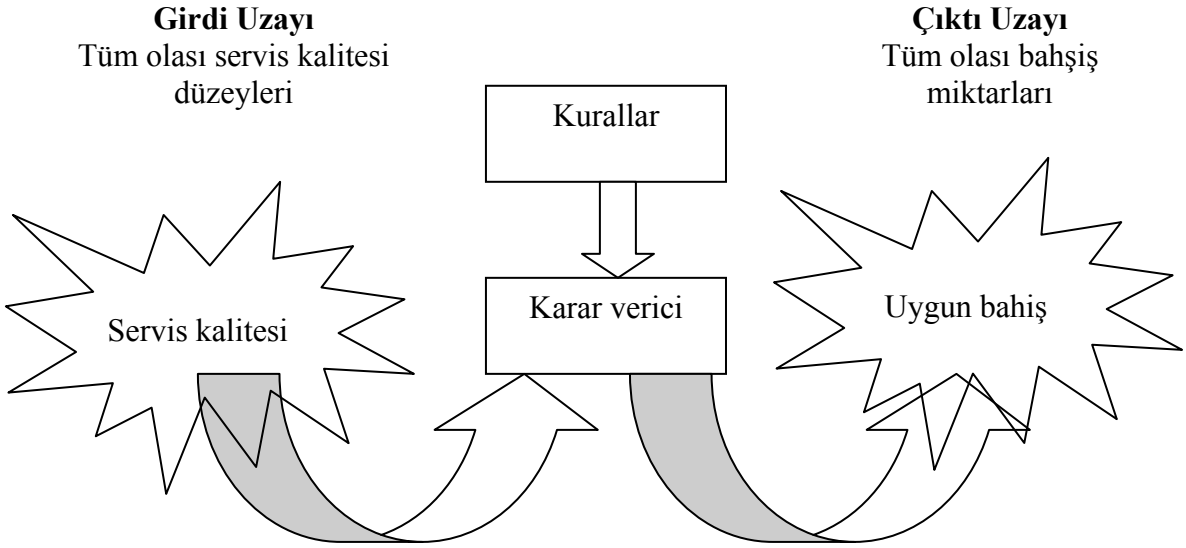
Şekil 3.11 Nonkonveks Normal Bulanık Küme (Öndemir, 2004)

Eğer A tek noktada konveks normal bir bulanık küme olarak tanımlanırsa, A bir bulanık sayıyı ifade eden bir küme olur (Öndemir, 2004).

3.5 Bulanık Kümelerde Kural Tabanı ve Çıkarım Süreci

Bulanık sistem teorisinin önemli bir katkısı bilgi tabanını doğrusal olmayan haritaya dönüştürme için sistematik bir metodoloji sağlamasıdır. Bulanık mantık, bir girdi uzayını, kural tabanı ve çıkarım motoru vasıtasıyla bir çıktı uzayına ulaştırır. Örnek olarak; bir restorandaki "servis kalitesine" göre "uygun bahşiş miktarı" kararı alınması verilebilir. İnsanlar bahşiş verecekleri zaman servisi çeşitli kriterlere göre ölçüp, bir bahşiş fonksiyonuna sokmazlar. Yapılan şey basittir. "EĞER servis kalitesi 'çok iyi' İSE 'çok' bahşiş ver" gibi birkaç tane kuralla konu açıklığa kavuşmuş olur. Bulanık mantık temelde problemlere getirdiği bu kolay çözüm nedeniyle tercih edilir (Baykal ve Beyan, 2004).

.



Şekil 3.12 Bir Girdi Altkümesinin Bir Çıktı Altkümesine Gitmesi (Öndemir, 2004)

Yukarıdaki kurala bakıldığında daha önceden karar vericinin aklında tanımlanan, servis kalitesi değişkenine ait dilsel bulanık kümeler olduğu görülür. Bu kümeler "Çok iyi", "Orta", "Zayıf" olsun, benzer şekilde bahşiş miktarı için bulanık kümeler ise "Az", "Orta" ve "Çok" olsun. Karar vericinin servis için "Çok iyi" ifadesini kullanması, herhangi bir kuralı belirli bir çıktı kümesine götürmez. Zira alınan servis kalitesinin "Çok iyi" kümesindeki üyelik derecesi uygun bahşiş miktarının "Çok" kümesindeki üyeliğini verecektir. Dolayısıyla üyelik derecesinin hesaplanmasını sağlayacak bir kesin (crisp) veya bulanık (fuzzy) girdiye ihtiyaç vardır. Böyle bir girdi sağlamak için servis kalitesinin 1–10 arasında puanlandığı varsayılabilir. Bir kabul de çıktı uzayı için yapılmalıdır. Bahşiş miktarının en küçük değeri %0 ve en büyük değeri %25 olsun.

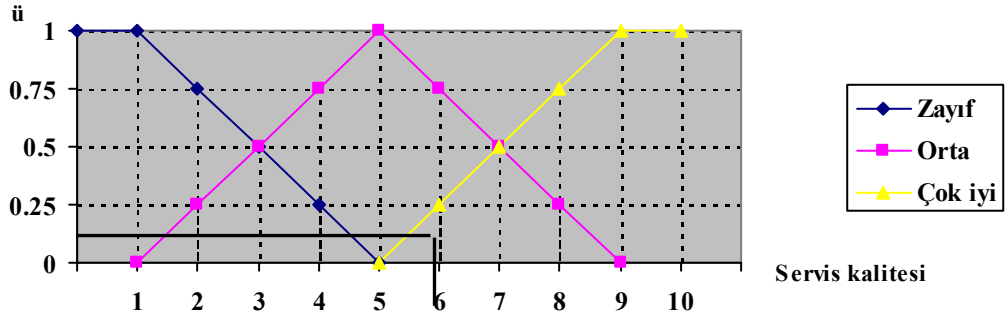
Yukarıdaki kabuller ışığında çıktı (bahşiş miktarı) şu adımlar izlenerek oluşturulur:

1. Kesin (crisp) girdinin bulanıklaştırılması (Fuzzification)
2. Kural tabanı vasıtasıyla çıkarım (Inference)
3. Çıktının durulaştırılması (Defuzzification)

3.6 Bulanıklaştırma (Fuzzification)

Bulanıklaştırma, kesin olarak ifade edilen bir girdinin, bulanık bir küme içerisindeki üyeliğinin bulunmasıdır. Bu yolla, kesin olarak ifade edilen bir sayı bir bulanık kümenin belli bir oranda elemanı olur (Yager ve Filev, 1994).

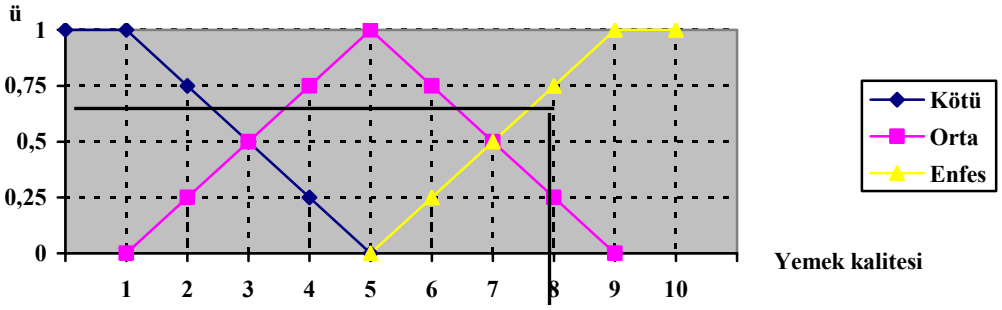
Servis kalitesine verilen 6 puanı, 0.25 üyelik derecesi ile "Çok iyi" kümesine üyedir (Şekil 3.14).



Şekil 3.13 Servis Kalitesi Değişkeninin Üyelik Fonksiyonu (Muşdal, 2005)

Bulanıklaştırma işlemi sonucunda elde edilen değer 0.25'tir ve çıkarım için bu değer kullanılacaktır.

Bahşiş örneğine, üyelik fonksiyonu Şekil 3.15'de gösterildiği gibi olan yemek kalitesi değişkeni eklensin. Kural tabanı da düzenlenerek aşağıdaki şekle getirilebilir;
EĞER servis kalitesi 'çok iyi' VEYA yemek kalitesi 'enfes' İSE 'çok' bahşiş ver.
Yemek kalitesine verilen 8 puanı, 0.75 üyelik derecesi ile "Enfes" kümesine üyedir (Şekil 3.15).



Şekil 3.14 Yemek Kalitesi Değişkeninin Üyelik Fonksiyonu (Muşdal, 2005)

Bulanık kümelerde birleşim işleminin "veya" operatörüyle ifade edilebildiği ve bu ifadenin de matematiksel olarak "max" ile gösterildiği bilinmektedir. Öyleyse, kural tabanının öncül kısmı (EĞER ile İSE arasındaki kısım) olan, "servis kalitesi 'çok iyi (Çİ)' VEYA yemek kalitesi 'enfes (EN)'" ifadesi matematiksel olarak aşağıdaki gibi çözülür.

$$\ddot{u}_{\text{Çİ}\cup EN} = \text{Max}\ \{\ddot{u}_{\text{ÇI}}(6), \ddot{u}_{EN}(8)\} = \text{Max}\ \{0.25, 0.75\} = 0.75 \quad (5.11)$$

Bu işlem sonucunda elde edilen 0.75 değeri, kural tabanının soncul (İSE'den sonraki) kısmında verilen uzayı daraltan bir çıkarımda rol oynar.

Bulanıklaştırmanın girdisi her zaman kesin (crisp) olmayabilir. Girdinin bulanık (fuzzy) bir küme olduğu durumlarda, bulanıklaştırma işlemi, iki bulanık kümenin kesiştirilmesiyle yapılır. Daha sonraki işlemler tamamıyla aynıdır (Öndemir, 2004).

3.7 Çıkarım (Inference)

Çıkarım, bulanık bir girdi kümesinin kural tabanında belirtilen bulanık bir çıktı kümesine gitmesidir. Çıkarım sırsında kural tabanının öncül kısmında (Antecedent) bulanıklaştırma ile elde edilen üyelik değeri, soncul kısımdaki (Consequent) bulanık kümeye yansıtılır ve bir çıktı bulanık kümesi elde edilir (Yager ve Filev, 1994).

3.8 Durulaştırma (Defuzzification)

Bulanık çıkarım kümesinden yola çıkarak kesin bir sayı elde etme işlemine durulaştırma denir. Durulaştırma işlemi için birçok metot mevcuttur. Ancak bu metotlardan yedi tanesi son yıllarda araştırmacılar tarafından sıklıkla önerilmektedir. Bu çalışmada, uygulamalarda sıkça kullanılan bu metotlardan en çok kullanılan üç tanesi özetlenmiştir (Baykal ve Beyan, 2004).

3.9 Bulanık Proses Zamanları

Belirtildiği gibi bulanıklık tercih veya belirsizlik olarak gösterilmektedir. Gerçek zamanlı sistemlerde proses zamanları olarak düşünülürse, bu zamanlar belirsiz değişkenler olarak kabul edilmelidir. Bu nedenle, bu tip belirsizliğin olduğu bir ortamda bulanıklığı ifade eden bulanık proses zamanların düşünülmesi oldukça doğal olmaktadır (Wang vd., 2002).

Şimdiye kadar pek çok modeldeki işlem zamanları bulanık sayılar olarak kullanılmıştır. Örneğin McCahon ve Lee bulanık yamuksal sayıları flow-shop problemlerinde kullanmışlardır. Ishibuchi genel bulanık sayıları sıralama modellerinde kullanılan işlem zamanları için kullanmışlardır. Sonuçta, tamamlanma zamanı işlem zamanının aynısı değildir. Görüldüğü üzere sıralama problemlerinde pek çok kez bulanık sayılara işlem zamanlarında yer verilmiştir (Mok vd., 2007).

Bulanıklık genel olarak seçilebilirliği veya kesinsizliği belirtmektedir. İşlem zamanları bulanık sayılar olduğu zaman, tamamlanma zamanları da bulanık sayılar olur
(Peng ve Liu, 2004).

3.9.1 Bulanık Aritmetik İşlemler

Eğer A $R \in (-\infty,+\infty)$' da, söz konusu kümenin bir elemanı ise $\mu_A(x)$ üyelik fonksiyonu $R \to [0,1]$ aralığında oluşur. Diğer bir deyişle A kümesi $A = [a_1, a_3]$ aralığında ise genel olarak $\mu_A(x)$ üyelik fonksiyonu (3.5) formülüyle gösterilebilir.

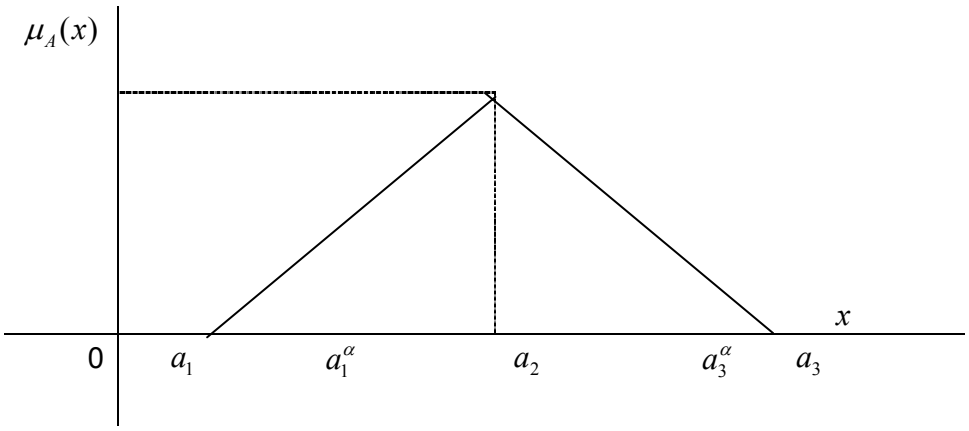
$$\mu_A(x) = \begin{cases} 0, & x < a_1 \\ 1, & a_1 \le x \le a_3 \\ 0 & x > a_3 \end{cases} \qquad (3.5)$$

Üyelik fonksiyonları genellikle, üçgensel üyelik fonksiyonları ve yamuk üyelik fonksiyonları olmak üzere iki başlık altında incelenmektedir.

$\mu_A(x)$ üçgensel üyelik fonksiyonu, (3.6) formülünde tanımlanmıştır (Mok vd., 2007).

$$\mu_A(x) = \begin{cases} 0, & x < a_1 \\ \dfrac{x - a_1}{a_2 - a_1}, & a_1 \le x \le a_2 \\ \dfrac{a_3 - x}{a_3 - a_2}, & a_2 \le x \le a_3 \\ 0, & x > a_3 \end{cases} \qquad (3.6)$$

(3.6) formülüne göre küme, $A = (a_1, a_2, a_3)$ olmalıdır. Burada a_2 normal değerli üyelik olarak tanımlanabilir. Bulanık Mantık bu noktada bir α katsayısına bağlı olarak a_2' ye yakın değerlerin, bu değere yüklenen anlam ile temsil edileceğini varsaymaktadır. Diğer bir deyişle a_2' deki belirsizlik, varsayılacak ya da dağılıma göre bulunabilecek bir α katsayısı ile tolere edilebilir. Söz konusu komşuluk Şekil 3.16' da gösterilmiştir



Şekil 3.15 Sayıların Komşuluğu

α değeri bulanık mantık terminolojisinde kesim katsayısı olarak adlandırılır. a_1^α ve a_3^α sayıları ise a_2 normal değerinin komşuluğunu oluşturan aralığın alt ve üst sınır değerleridir. Diğer bir deyişle a_1^α ve a_3^α aralığındaki tüm sayılar a_2 normal değeri ile aynı anlama sahiptir. a_1^α ve a_3^α değerleri (3.7) ve (3.8) formülleri yardımıyla bulunabilir (Terano, 1997).

$$\frac{a_1^{\alpha} - a_1}{a_2 - a_1} = \alpha \qquad (3.7)$$

$$\frac{a_3 - a_3^{\alpha}}{a_3 - a_2} = \alpha \qquad (3.8)$$

(3.7) ve (3.8) formüllerinden $\forall \alpha \in [0,1]$ için $A_{\alpha} = [a_1^{\alpha}, a_3^{\alpha}]$ aralığı oluşturulabilir. a_1^{α} ve a_3^{α} değerleri (3.9) ve (3.10) formüllerinde gösterilmiştir.

$$a_1^{\alpha} = \alpha(a_2 - a_1) + a_1 \qquad (3.9)$$

$$a_3^{\alpha} = a_3 - (a_3 - a_2)\alpha \qquad (3.10)$$

Örneğin üçgensel bulanık mantık sayılarına ilişkin küme $A = (-5,-1,1)$ ise bu durumda (3.6) formülünden üyelik fonksiyonu,

$$\mu_A(x) = \begin{cases} 0, & x < -5 \\ \dfrac{x+5}{4}, & -5 \le x \le -1 \\ \dfrac{1-x}{2}, & -1 \le x \le 1 \\ 0, & x > 1 \end{cases} \qquad (3.11)$$

olarak bulunur. Eğer karar verici α kesim katsayısını 0,5 olarak saptamışsa -1 normal değerinin komşuları (3.9) ve (3.10) formüllerinden $a_1^{0,5} = -3$ ve $a_3^{0,5} = 0$ olarak bulunacaktır. Diğer bir deyişle -1 normal değeri ile aynı anlam düzeyinde bulunan sayılar kümesi $[-3,0]$ aralığıdır.

4. İŞ SIRALAMA

Hayatın büyük bir bölümü sıralama problemlerinden oluşmaktadır. Bu durum bilinçsizce olabilmektedir. Adalet duygusu ve daha bir çok faktör bu bağlamda ön plana çıkmaktadır. Sıralama problemlerinde temel felsefeler zamanlama ve maliyettir. Çünkü sıralamada zaman maliyetle beraber hareket etmektedir. Minimum zamanda işin yapılması önemlidir. İşlere de bu şekilde yaklaşılmaktadır. Yani maksimum kar minimum maliyettir.

Sıralama işlemi üretim, imalat, yönetim, bilgisayar bilimi ve bunun gibi pek çok alanda kullanılan önemli bir prosestir. Genel olarak ihtiyaç duyulan kısıtların altında, kaynakların yüksek zamanlı işlere atanmasıyla ilgilenir. Uygun olan sıralama şekli, bütün zamanı küçültürken, ürün kalitesini arttırır. Kaynaklar ve işler pek çok formlarda olabilmektedir. Sıralama amacı, sonuncu işin tamamlanma zamanı minimizasyonu olabileceği gibi, her işin teslim tarihinden önce tamamlanması da olabilmektedir (Hong vd., 1998).

Sıralama problemlerinin yapısı gereği pek çok kabuller yapılmaktadır. Bu kabuller şunlardır:

1. Her iş bir varlıktır; her iş belirli operasyonlardan oluşmuştur. Aynı işin iki işlemi aynı anda işlenemez. Böylece bitmiş bir ürünün, montaj önceliğine göre imal edilmiş parçalarının ortaya çıkardığı problemleri ihmal ediyoruz.
2. Kesinti yoktur: Bir operasyon o makinede başlamadan önce, daha önce başlamış olan operasyon tamamlanmalıdır.
3. Her makinede bir adet olmak üzere her bir işin m sayıda belirli operasyonları vardır. bir işin aynı makinede iki işlem gerektireceği olasılığına izin verilmemektedir. Aynı şekilde her bir işin tüm makinelerde işlenmesine ısrar edilmektedir. Bir iş, bir veya daha fazla makineyi atlayamaz.
4. Erteleme yoktur. Her iş tamamlanıncaya kadar işlenmelidir.
5. İşlem zamanları sıralamadan bağımsızdır. Burada özellikle iki şey kabul edilir. Birincisi her işi tesis etme zamanı sıralamadan bağımsızdır. Yani bir işin yapılması için makinenin ayarlanması son işten bağımsızdır. İkinci olarak işlerin makineler arasında taşınması için geçen zaman gözardı edilebilir.
6. İşlem sırasında ara stoklara müsaade edilir; yani işler bir sonraki makinenin boşalmasını bekleyebilir. Bu yersiz bir varsayım değildir. Bazı problemlerde işlerin yapılması operasyondan operasyona devamlı olmalıdır.

7. Her makineden sadece bir tane vardır. bir işin yapılmasında seçilebilecek makineler olmasına izin verilmemektedir. Bu varsayım diğerlerinin arasında belli makinelerin yaratacağı darboğazlardan kurtulmak için bu makinelerin çoğaltılmasını bertaraf eder.
8. Makineler işsiz (boş) olabilir.
9. Hiçbir makine aynı anda birden fazla iş yapamaz.
10. Makineler hiçbir zaman bozulmamaktadır. Ve sıralama periyodu boyunca hazır bulunmaktadır.
11. Teknoloji kısıtlar önceden bilinmekte ve sabittirler.
12. Hiçbir rasgelelik yoktur. Özellikle; a) işlerin sayısı biliniyor ve sabit, b) makinelerin sayısı biliniyor ve sabit, c) işlem zamanları biliniyor ve sabit, d) hazırlık zamanları biliniyor ve sabit, e) bir problemi belirtmek için gerekli bütün nicel değerler biliniyor ve sabit.
Bazen belirli örneklerde bu kabullerin bir veya ikisi gözardı edilmektedir (Başlıgil, 1988).

Sıralama genelde iki sınıfa ayrılmaktadır: öne almalı ve öne almasız sıralama. Öne almalı sıralama aktivitelerin, bazı diğer önemli aktivitelerin daha hızlı işlem görmesi için kesikli olmasına izin vermektedir. Diğer taraftan, öne almasız sıralama sistemleri ise, aktivitelerin hiçbir zaman kesikli olmasına izin vermemektedir (Hong vd., 1998).

İş sıralamada erteleme yoktur. Sıralama optimize edildikten sonra erteleme yapılmamaktadır. Zaten stoklara müsaade edilmezse geciken işin gecikmesi bütün sıraya etki etmektedir. Diğer taraftan makinalar işi, işler de makinayı bekleyebilmektedir. Arzu edilen ise makina ve iş hazır ise hemen işin başlayabilmesidir. Öyle bir planlama yapılmalıdır ki, minimum ara stoklara izin verilebilmelidir.

Sıralamanın amaçları firmadan firmaya ve günden güne değişecektir. Belki de amaç bütün bölümlerde pahalı işgücü ve makinaların nadiren boş kalacağı eşit bir aktivite seviyesi sağlamaktır. Amaç mümkün olduğu kadar kısa zamanda işleri bitirmek veya işi belirli kontrat tarihinde başarmış olmaktır.

Pek çok kaynak üzerinde pek çok işin sıralanması genel haliyle NP-tam (NP- complete) problemi olarak bilinmektedir. Sıralama problemlerinin yüksek karmaşıklığı nedeniyle pek çok sezgisel yaklaşım farklı sıralama problemi çözümünde geliştirilmiştir
(Litoiu ve Tadei, 2001).

Genel bir sıralama modeli bir dizi kaynak (R) üzerinde sıralanan bir iş (T) sistemini içerir. Bu noktada bir dizi maliyet fonksiyonu performansı ve bir dizi kısıtlar önem kazanmaktadır. Kaynaklar pek çok durumda makineler veya işlemcilerdir (Anglani vd., 2005) :

$R = \{R_1, R_2, ..., R_m\}$

Kaynaklar benzer veya farklı işlem hızlarına sahip olabilirler. Pek çok durumda, ek kaynaklar da devreye girebilmektedir. Sistem işleri şöyle tanımlanabilir:

$T = \{T_1, T_2, ..., T_n\}$ ifadesi bir küme işi tanımlamaktadır. Burada bir iş, bir program aktivitesi veya bir programın yürütülebilir bir kısmını temsil etmektedir.

Şimdiye kadar bulanık kümeler sıralamada iki başlık altında toplanmışlardır
(Nezhad ve Assadi, 2008):

* Modelleme: sıralama determinizmi ima etmesine rağmen, genel sıralama modellerinde kesinsizlik görülmektedir. Bu durum bulanık kısıtlar, bulanık teslim tarihleri ve bulanık proses zamanları ile ilişkilendirilmektedir.
* Deterministik modellerin çözümü.

4.1 Kısıtlar

Bu noktada iş sıralamada iki önemli kısıt devreye girmektedir. Bunlar şöyle sıralanmaktadır:

a) Öncelik Kısıtı: Bir işin yapılabilmesi için o işten önceki işlerin mutlaka yapılması gerekliliğidir.

b) Teknolojik Kısıt: O işin teknolojik nedenlerden dolayı yapılabilmesi özelliğidir.
İş görenler makinelerin onlara sağladıkları teknolojik koşullar ile ancak o işi yapabilmektedir.

4.2 Çalışma Ölçütleri

d_i : Teslim zamanı. Bir işin başlangıcından bitip müşteriye teslim edilmesine kadar geçen zaman.

a_i : Müsaade edilen zaman.

W_{ik}: J_i işinin, k' ıncı işlemden önce geçen bekleme zamanıdır. k' ıncı iş, işlem sırasına göre gelen iştir.

W_i : J_i' nin toplam bekleme zamanı. $W_i = \sum_{k=1}^{m} W_{ik}$

C_i : Tamamlanma zamanı. $C_i = r_i + \sum_{k=1}^{m}(W_{ik} + P_{ij(k)})$

F_i : Akış zamanı. $F_i = C_i - r_i$

r_i : Hazırlık zamanı.

P_{ji} : j makinesinde i işinin işlem zamanı.

$L_i : C_i$' nin gecikme zamanı. Basitce söz verilen tarih ile bitirme (tamamlanma) zamanı arasındaki fark. $L_i = C_i - d_i$ dir.

$T_i : J_i$ nin gecikmesi; j makinesindeki i işinin gecikmesi T_i' dir. $T_i = \max\{L_i, 0\}$

E_i: J_i' nin erken bitirilmesi $E_i = \max\{-L_i, 0\}$ dir.

$\bar{X}$: İşlerin ortalaması . $\bar{X} = 1/n \sum_{i=1}^{n} X_i$.

$N_w(t)$: t anında işlenmeye hazır olmayan veya makineler arasında bekleyen işlerin sayısı.

$N_c(t)$: t anında tamamlanmış olan işlerin sayısı.

$N_u(t)$: t anında tamamlanacak olan işlerin sayısı. Bitmek üzere olan işlerin sayısı.

$N_p(t)$: t anında işlenmekte olan işlerin sayısı.

$N_w(t) + N_p(t) + N_c(t) = n$ n : Yapılacak işlerin sayısı.

$N_w(t) + N_p(t) = N_u(t)$

$N_u(0) = n$ ve $N_u(C_{\max}) = 0$

P_m : Paralel benzer makine.

Q_m : Farklı hızlardaki paralel makine.

R_m : Birbirine paralel ilişkisiz makineler.

F_m : Flow shop hali.

FF_s : Esnek flow shop hali.

O_m : Open shop hali.

J_m : Job shop hali.

prmp : Alıkoyma hali.

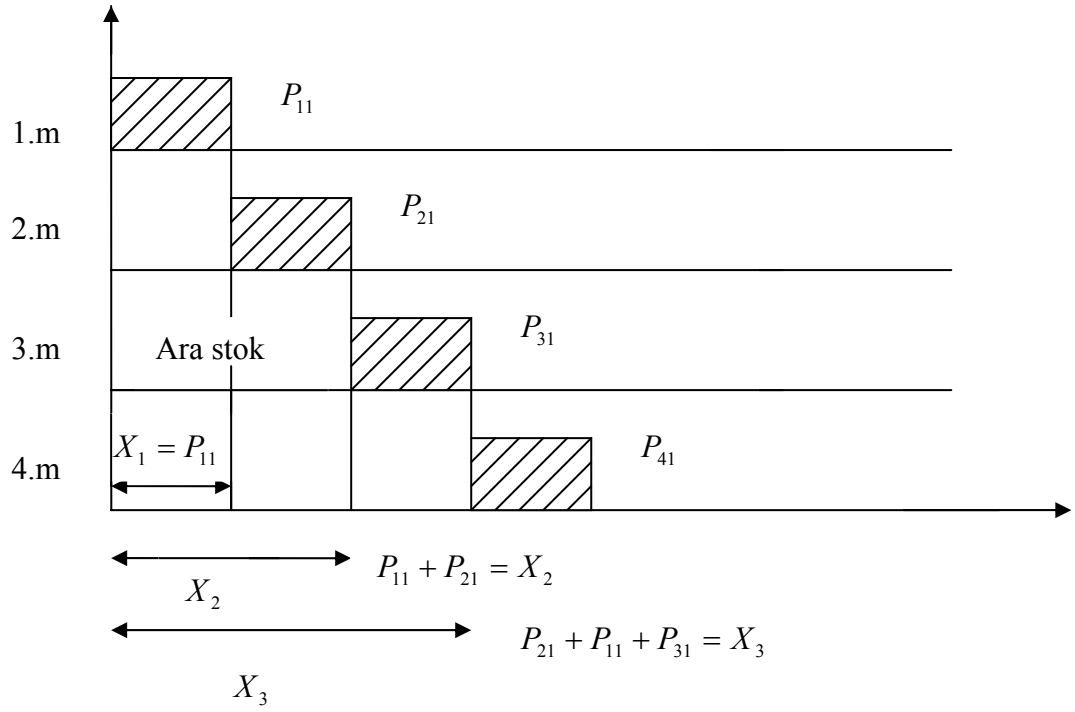
prec : Öncelik zorlama hali.

brkdown : Bozulma hali.

M_j : Seçilebilir makine kısıtlamaları.

prmu : Yerdeğiştirme (permütasyon).

block : Önleme



Şekil 4.1 Gantt diyagramı.

1.Hal : $r_i = 0$ ise $C_i = F_i$ olur. $C_i = \sum_{j=1}^{n} (P_{ji} + W_{ji})$ olmaktadır.

2. Hal (İdeal Hal) : $\sum_{j=1}^{n} W_{ji} = 0$, $r_i = 0$, $C_i = \sum_{j=1}^{n} P_{ji} = F_i$ dir.

3. Hal : $r \neq 0$, $\sum_{j=1}^{n} W_{ji} = 0$, $C_i = r_i + F_i$, $C_i = r_i + \sum_{j=1}^{n} p_{ji}$

Montaj hattında beklemeler olmamaktadır. Beklemeler sıfır kabul edilir. Bu durum en ideal haldir. Montaj hattında taşımalar da yoktur. İşlem sürelerine dahildir. Bu durum 2. HAL ‘ i tanımlamaktadır.

4.3 Job-Shop Tipi Sıralama Problemlerinin Parametreleri

n : işlerin sayısı

m : Makinaların sayısı.

A : Makina atölyesinde uygulanacak modeli veya akış modelini tanımlar. A şunlar olabilir:

F : Flow-shop durumu içindir. Yani bütün işler için makine sırası aynıdır.

P : Flow-shop durumu permütasyonu içindir.

G : Teknolojik sınırlamalar halinde hiçbir kısıtlamanın olmadığı genel job-shop durumu.

B : Programın değerlendirilmesi ile performans ölçütünü tanımlar.

5. PARALEL MAKİNELERİN ÇİZELGELENMESİ

Paralel sıralanmış bir dizi makine hem teorik açıdan hem de pratik açıdan önemlidir. Teorik bakış açısında, tek makine durumunun genelleştirilmiş durumudur ve esnek kayan bant halinin özel bir şeklidir. Pratik bakış açısında önemlidir. Çünkü gerçekte karşımıza en sık çıkan durum paralel kaynaklar durumudur. Öte yandan paralel makine teknikleri çok aşamalı sistemlerin araştırılması prosedürlerinde de kullanılır.

Paralel durumdaki makinelerle ilgilenirken, yapım zamanı kayda değer bir amaç haline gelir. Pratikte paralel durumdaki makinelerde yük dengelemesi problemi ile sık karşılaşılır ve yapım zamanını minimize ederek iyi bir denge sağlanmış olunur (Park ve Kim, 1997).

Paralel makinelerde sıralama iki aşamalı bir işlem olarak kabul edilmelidir. Birinci aşamada, hangi işin hangi makinede işleneceği saptanmalı, ikinci aşamada ise her makinedeki iş yükü için sıralama tespit edilmelidir. İşlem zamanı amacına göre sadece birinci aşama önemlidir.

Paralel makinelerde öncelikler tek makine durumuna göre daha büyük önem taşır. Tek makine koşulunda öncelikler genelde işlerin teslim zamanlarının farklı olması durumunda önem taşır. Ancak paralel makinelerde öncelikler işler aynı zamanda teslim edilecekse dahi önemlidir (Moukrim ve Quilliot, 2005).

5.1 Paralel Makine Çeşitleri

Literatürde yapılan araştırmalardan paralel makine çeşitlerini dört grup altında toplayabiliriz. Bunlar şöyle sıralanmaktadır:

- Aynı tip olup, hızları da aynı olan paralel makineler.
- Aynı tip olup, hızları farklı olan paralel makineler.
- Farklı tip olup, aynı hızda olan paralel makineler.
- Farklı tipte olup, farklı hızda olan paralel makineler.

5.2 Paralel Tezgahlardan Oluşan Atölyede Sıralama Problemi

Atölyede birden çok tezgahın paralel olarak çalışması, sıralama problemi için değişik modeller kurmayı gerektirmektedir. Bu tür bir atölyede bulunan tüm tezgahlar, mevcut n adet işi yapabilecek kapasite ve yetenektedirler. Bu kabulden başka tezgah ve işler için şu kısıtlar konulmaktadır:

1. m adet tezgah sürekli olarak çalışabilirler ve bir tezgah aynı anda birden çok işi işleyemez.
2. Başlangıç anında mevcut, birbirinden bağımsız n adet iş, yalnız bir tezgah tarafından işlenerek atölyeyi terk ederler.
3. İş tarifleri önceden bilinmektedir, işlem süreleri belirli ve sabittir.

Paralel tezgahlar hali diğer problemlerden farklı olarak 2 boyutlu bir kararı geliştirmek için incelenmektedir. Kararın birinci boyutu işlerin tezgahlara dağıtılması, ikinci boyutu ise, bu tezgahlardaki işlenme sırası ile ilgilidir. Karar prosesi bu iki boyutu kapsayacak şekilde ele alınarak aşağıdaki etkinlik ölçütlerine göre problem incelenmektedir
(Koulamas ve Kyparisis, 2000).

5.2.1 Yapım Süresinin Minimizasyonu

Yapım süresinin minimizasyon problemi öne geçmeli ve geçmesiz halde bazı farklılıklar göstermektedir.

1) Öne geçmeli halin incelenmesi:

Bir iş, bir tezgahta bitmeden başka bir tezgaha aktarılabiliyorsa, problem oldukça basit bir şekilde çözülebilir. Bu halde yapım süresinin minimum değeri;

$$M^* = \frac{1}{m}\sum_{j=1}^{n} P_j \qquad (5.1)$$

Şeklinde ifade edilebilir. M^* değeri hesaplandıktan sonra işler şu şekilde sıralanır:

Adım 1: Herhangi bir iş, 0 anında başlamak üzere 1. tezgahta programlanır.

Adım 2: Programlanmamış işlerden biri bu işten sonra gelecek şekilde aynı tezgahta programlanır. Bu tezgahtaki işlem süreleri M^* oluncaya kadar aynı işlem tekrarlanır.

Adım 3: tezgahta en son programlanan iş M^* süresi içinde bitirilemiyorsa, bu işe ait geri kalan işlem süresi bir sonraki tezgahta programlanır ve adım 2 ye dönülür. Tezgahta en son programlanan iş M^* süresi sonunda bitiriliyorsa, bir sonraki tezgaha programlanmamış işlerden biri tahsis edilerek adım 2 ye dönülür.

Adım 2 ve adım 3 tüm işler programlanıncaya kadar sürdürülür (Başlıgil, 1988).

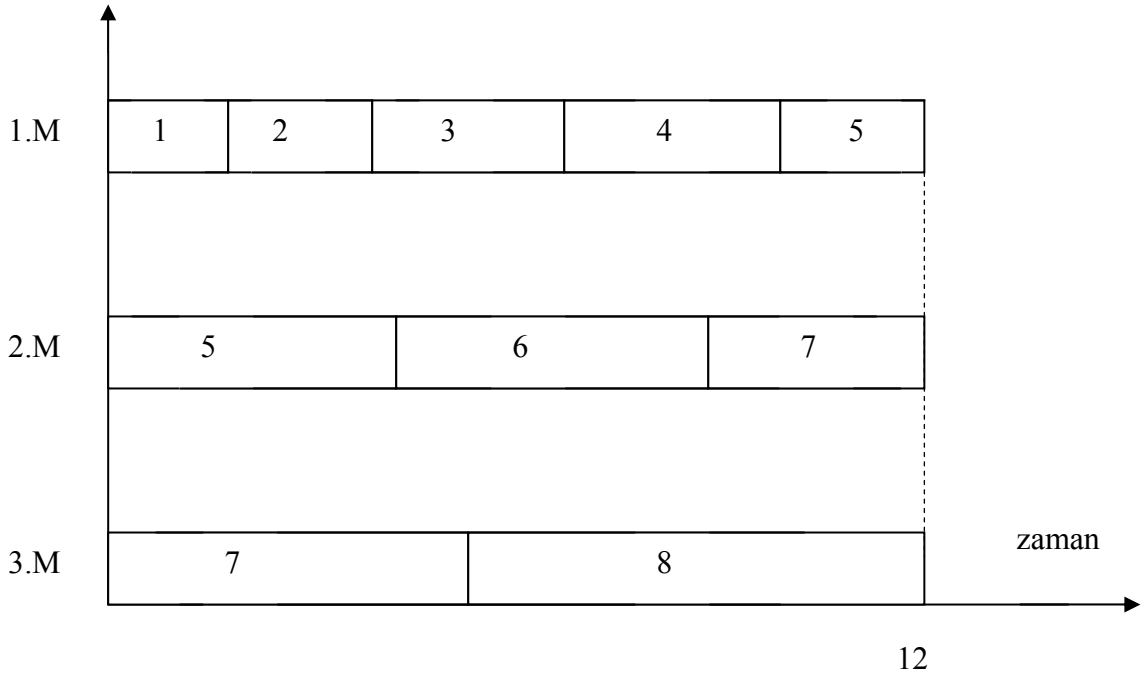
Örnek Problem:

J işi	1	2	3	4	5	6	7	8
P_j	1	2	3	4	5	6	7	8

Şekil 5.1 Öne geçmeli hal ile ilgili bir örnek

$$M^* = \frac{1}{m}\sum_{j=1}^{n} P_j = \frac{1}{3}\sum_{j=1}^{n} P_j = 12 \text{ dir.}$$

O halde işler şu şekilde gösterilir:



Şekil 5.2 8 iş 3 makine halinin gantt diyagramında gösterimi

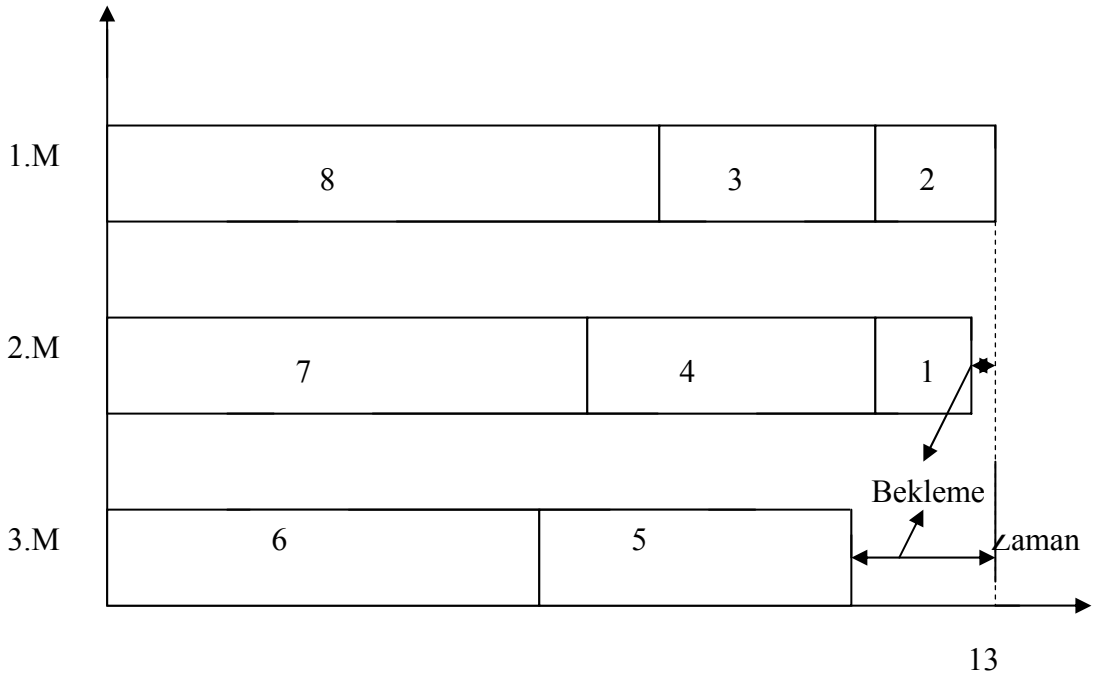
2) Öne geçmesiz halin incelenmesi:

Öne geçmesiz halde, yapım süresinin minimizasyonu probleminde bazı güçlükler ortaya çıkmaktadır. Tam çözüm için tamsayılı programlama yöntemine dayanan bazı modeller kurulmuştur, ancak bunların kullanışlı çözümlerini bulmak mümkün olamamıştır. Öne geçmesiz hal için yeterli hassasiyetle çözüm verebilen bazı sezgisel algoritmalar geliştirilmiştir. Bunlardan biri şöyledir:

Adım 1: İşler LPT (Longiest processing time) sırasına göre dizilir (En uzun işlem süreli iş, en başa gelecek şekilde).

Adım 2: İşler bu sıraya göre tezgahlara tahsis edilir. Tahsis şöyle yapılır. Programlanmamış işlerden ilki en az yüklü tezgaha tahsis edilir. Bu işlem tüm işler programlanıncaya kadar sürdürülür. Örnek problem için bu yöntemin verdiği çözüm şöyledir.

LPT sırası: 8,7,6,5,4,3,2,1



Şekil 5.3 LPT' ye göre oluşturulan Gantt diyagramı

Paralel makinelerin çizelgelenme problemi literatürde genel olarak birbirinden bağımsız işlerin, birden fazla paralel makineye, işlerin toplam akış süresinin (flow time) ya da en son işin tamamlanma süresinin (makespan) en küçüklenmesi olarak ele alınmaktadır (sırasıyla $P//F_{\max}$ ya da $P//C_{\max}$ olarak not edilir) (Shabtay ve Kaspi,2006).

Çizelgeleme problemlerinin karmaşıklık derecelerini konu eden bir çok çalışma yaplmış ve çoğunun NP-Tam (NP-Complete) sınıfına ait olduğu sonucu ortaya konmuştur. Paralel makinelerin çizelgelenme problemlerinin çoğu da NP sınıfına aittir. Her ne kadar en uygun çizelgelemeyi bulmak için Matematiksel Programlama uygulama imkanı olsa da, bu problemlerin çözüm süreleri önceden belli olmayan polinomyal bir yapı arz ettiğinden, boyutları arttığında etkin çözüm alınamaz. Bu yüzden araştırmacılar, daha etkin ve en uygun

sonuca yakın sonuç üretebilen sezgisel algoritmalara yönelmişlerdir. Bunlardan bazıları Tabu Search, Yapar Sinir Ağları ve Genetik Algoritma (GA) olarak sayılabilir

Bu algoritmalara arasında GA sonuca hızlı yakınsaması, kolay tatbik edilebilmesi ve en uygun sonuca yaklaşık sonuçlar verebilmesi nedeniyle, özellikle sanayide uygulanan üretim çizelgeleme problemlerine uygulamada, bir adım öne çıkmaktadır (Balin vd., 2007).

5.3 Paralel Makinelerin Çizelgelenme Probleminin İfadesi

5.3.1 Özdeş Paralel Makinelerde Sıralama

Özdeş (identical) paralel makinelerde işlerin tamamlanma süresini en küçükleyen (makespan) çizelgeleme problemi şu şekilde tanımlanabilir: n tane bağımsız iş, m tane özdeş paralel makineye atanacaktır, her işin işlem süresi sabittir ve her iş, her makinede işlenebilir. En uygun çizelge, tüm işlerin tamamlanması için gereken süreyi en küçükleyen çizelgedir. j işinin işlem süresi $t(j)$ olarak not edilmektedir. j işinin, i makinesinde işlem görüp görmediği $x(i,j)$ mantıksal (boolean) değişkeni tarafından belirlenir. $x(i,j)$ değişkenleri tarafından oluşturulan X matrisi şu özelliklere sahipir (Sun ve Wang, 2003):

- her elemanının değeri 1 ya da 0'a eşittir,
- her sütununda sadece bir tane 1 vardır,
- toplam n tane 1 değerine sahip elemanı vardır.

Tüm işlerin tamamlanması için gereken süre de şu şekilde ifade edilebilir:

$$C_{\max} = \max_{i=1}^{m} \left\{ \sum_{j=1}^{n} x(i,j) \times t(j) \right\} \qquad (5.2)$$

Bu takdirde problemin amaç fonksiyonu ise şu sekide yazılır:

$$\min C_{\max} = \max_{i=1}^{m} \left\{ \sum_{j=1}^{n} x(i,j) \times t(j) \right\} \qquad (5.3)$$

Bu problemin GA ile kodlanması ve çözülmesine bir sonraki paragrafta yer verilmektedir (Balin vd, 2008).

5.3.2 Özdeş Olmayan Paralel Makinelerde Sıralama

Özdeş olmayan (non-identical) paralel makinelerde ise, m adet özdeş olmayan paralel makinede n adet iş prosese sokulmaktadır. j işi i makinesiyle de işleme sokulabilmektedir.

Her makinenin farklı hızı vardır. ayrıca belli bir zamanda sadece bir iş prosese girebilmektedir. Her iş kesilme olmaksızın işlem görmektedir. i makinesindeki j işinin proses zamanı $t(i,j)$ ile ifade edilmektedir. Maksimum tamamlanma zamanını minimize eden bir sıralama bulunmaya çalışılmaktadır.

$x(i,j)$ boole değişkenleridir. Burada j işi i makinesiyle (eğer $x(i,j)=1$) işleme sokulmakta, (eğer $x(i,j)=0$) ise sokulmamaktadır. X matrisi $x(i,j)$ değişkenlerinden oluşmaktadır ve şöyle özelliklere sahiptir (Balin vd, 2008):

- Bütün elemanlar "0" veya "1" e eşittir, $x(i,j) \in \{0,1\}$
- Her sütun değeri "1" olan bir tek elemana sahiptir, $\sum_{i=1}^{m} x(i,j) = 1, j = 1,..,n$
- "1" olarak değerlendirilen eleman sayıları n' dir, $\sum_{j=1}^{n} \sum_{i=1}^{m} x(i,j) = n$

Farklı makinelerde işlem gören j işi sıradaki eşitlikle açıklanabilir:

$t(i_1, j) \times V_1 = t(i_2, j) \times V_2$

Maksimum tamamlanma zamanı ise şuna eşittir:

$$C_{\max} = \max_{i=1}^{m} \left\{ \sum_{j=1}^{n} x(i,j) \times t(i,j) \right\} \qquad (5.4)$$

Böylece, amaç fonksiyonu şöyle formüle edilmektedir:

$$\min C_{\max} = \max_{i=1}^{m} \left\{ \sum_{j=1}^{n} x(i,j) \times t(i,j) \right\} \qquad (5.5)$$

6. GENETİK ALGORİTMA (GA)

Genetik algoritma (GA), doğadaki evrim mekanizmasını örnek alan, stokastik bir arama metodudur ve bir veri grubundan özel bir veriyi bulmak için kullanılır. Genetik algoritma rassal arama tekniklerini kullanarak çözüm bulmaya çalışan, parametre kodlama esasına dayalı bir arama tekniğidir (Gözütok ve Özdemir, 2004).

Genetik algoritma tekniği, Michigan Üniversitesinde yer alan John Holland ve arkadaşlarının liderliğindeki çalışmalar sonucu 1970 li yıllarda ortaya çıkmış ve 1975 de Holland "Doğal ve Yapay Sistemlerin Uygulanması" adlı kitabını yayınlamıştır (İşçi ve Korukoğlu, 2003). Holland tarafından 1975 yılında geliştirilen Genetik Algoritma (GA), bir çok kombinatoriyel optimizasyon problemi üzerinde başarılı bir şekilde uygulanmıştır Genetik Algoritmalar, Evrimsel Genetik ve Darwin'in Doğal seleksiyonuna benzerlik kurularak geliştirilmiştir (Engin ve Fığlalı, 2002).

Bugün bilgisayar yöntemleri biyolojik değerlendirmeden esinlenerek evrimsel hesaplama olarak adlandırılan bir şemsiye altında gruplandırılmıştır. Evrimsel hesaplamanın ana elemanları aşağıda tanımlanmaktadır (İşçi ve Korukoğlu, 2003):

(1) Değerlendirme stratejileri
(2) Evrimsel programlama
(3) Genetik algoritmalar.

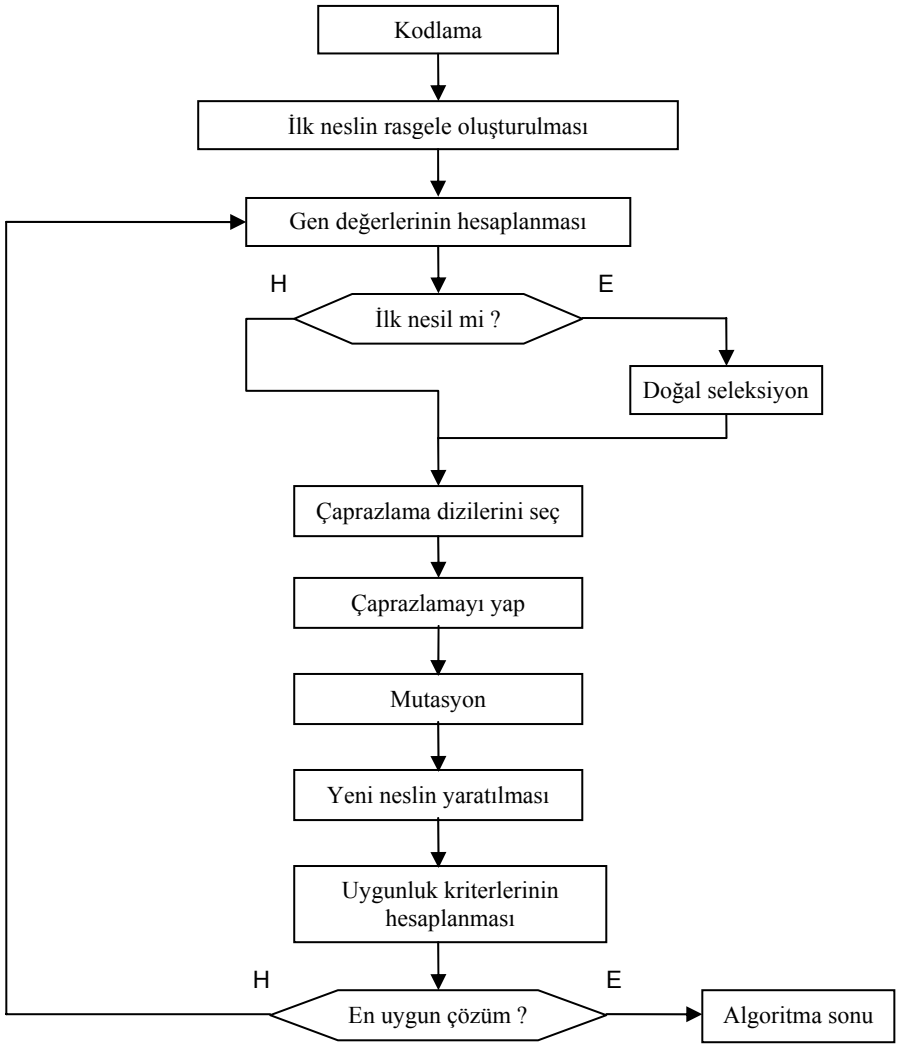
Günümüzde, doğrusal programlamayla modellenemeyen karmaşık problemlerin çözümü için Yapay Sinir Ağları, Tavlama Benzetimi (Simulated Annealing-SA) ve Genetik Algoritma (GA) gibi stokastik metodların kullanımı hızla artmıştır. Günümüzde çok hızlı gelişen bilgisayarların saniyedeki işlem yapma kapasiteleri GA'nın ihtiyaçlarını karşılayabilecek seviyeye çoktan ulaşmıştır. Ancak şu da unutulmamalıdır ki, GA kesin çözümü elde edip edemediğini bilememekte, ancak en iyi çözüme devamlı yaklaşmaktadır
(Uçaner ve Özdemir, 2002).

6.1 Genetik Algoritma Adımları

Genetik algoritmalar doğada geçerli olan en iyinin yaşaması kuralına dayanarak sürekli iyileşen çözümler üretir. Bunun için "iyi"nin ne olduğunu belirleyen bir *uygunluk* (fitness) fonksiyonu ve yeni çözümler üretmek için *yeniden kopyalama* (recombination), *değiştirme* (mutation) gibi operatörleri kullanır. Genetik algoritmaların bir diğer önemli özelliği de bir

grup çözümle uğraşmasıdır. Bu sayede çok sayıda çözümün içinden iyileri seçilip kötüleri elenebilir. Genetik Algoritmalar, yeni bir nesil oluşturabilmek için 3 aşamadan geçer (Balin vd., 2007) :

1. Eski nesildeki her bir bireyin uygunluk değerini hesaplama.
2. Bireyleri, uygunluk değerini göz önüne alarak (uygunluk fonksiyonu) kullanılarak seçme.
3. Şeçilen bireyleri, çaprazlama (crossover), mutasyon (mutation) gibi genetik operatörler kullanarak uyuşturma.



Şekil 6.1 Genetik Algoritma'nın genel yapısı (Balin vd., 2007).

GA yöntemi, yıllar boyu süregelen genetik mühendisliği ve biyoloji alanında yapılan çalışmalar sonucu ortaya atılmış bir tekniktir ve her sisteme uygulanamaya bilir. Standart bir GA yöntemi aşağıdaki gibi verilebilir (Liu ve Wu, 2003):

- ✓ Başlangıç populasyonunu rastlantısal olarak üret.
- ✓ Populasyon içindeki tüm kromozomların amaç fonksiyonu değerlerini hesapla.
- ✓ Tekrar üreme, çaprazlama ve mutasyon operatörlerini uygula.
- ✓ Oluşturulan her yeni kromozomun amaç fonksiyonu değerlerini bul.
- ✓ Amaç fonksiyonu değerleri kötü olan kromozomları populasyondan çıkar.
- ✓ 3-5 arasındaki adımları tekrar et.

Genetik algoritmanın çalışmasını aşağıdaki adımlarla özetleyebiliriz (Emel ve Taşkın, 2002) :

1. Adım : Olası çözümlerin kodlandığı bir çözüm grubu oluşturulur. Çözüm grubuna biyolojideki benzerliği nedeniyle populasyon, çözümlerin kodları da kromozom olarak adlandırılır. Bu adıma populasyonda bulunan birey sayısını belirleyerek başlanır. Bu sayı için bir standart yoktur. Genel olarak önerilen 100-300 aralığında bir büyüklüktür. Büyüklük seçiminde yapılan işlemlerin karmaşıklığı ve aramanın derinliği önemlidir. Populasyon bu işlemden sonra rasgele oluşturulur.

2. Adım : Her kromozomun ne kadar iyi olduğu bulunur. Kromozomların ne kadar iyi olduğunu bulan fonksiyona uygunluk fonksiyonu denir. Bu fonksiyon işletilerek kromozomların uygunluklarının bulunması ise hesaplama(evalution) adı verilir. Bu fonksiyon genetik algoritmanın beynini oluşturmaktadır. GA da probleme özel çalışan tek kısım bu fonksiyondur. Çoğu zaman GA nın başarısı bu fonksiyonun verimli ve hassas olmasına bağlı olmaktadır.

3. Adım : Bu kromozomları eşleyerek yeniden kopyalama ve değiştirme operatörleri uygulanır. Bu sayede yeni bir populasyon oluşturulur. Kromozomların eşlenmesi kromozomların uygunluk değerlerine göre yapılır. Bu seçimi yapmak için rulet tekerleği seçimi, turnuva seçimi gibi seçme yöntemleri vardır.

4. Adım : Yeni kromozomlara yer açmak için eski kromozomlar ortadan kaldırılır. Eski kromozomlar çıkartılarak sabit büyüklükte bir populasyon sağlanır.

5. Adım : Tüm kromozomların uygunlukları tekrar hesaplanır. Tüm kromozomlar yeniden hesaplanarak yeni populasyonun başarısı bulunur.

6. Adım : GA defalarca çalıştırılarak çok sayıda populasyon oluşturulup hesaplanır. Eğer zaman dolmamışsa 3. adıma gidilir.

7. Adım : O ana kadar bulunan en iyi kromozom sonuçtur. Çünkü populasyonların hesaplanmasında en iyi bireyler saklanmıştır.

6.2 Genetik Algoritma' nın Diğer Yöntemlere Üstün Tarafları

Son yıllarda araştırmacılar Genetik Algoritma (GA), tavlama benzetimi (simulated annealing), karınca kolonisi, yapay sinir ağları, tabu arama gibi çeşitli yöntemler geliştirmişlerdir. Bu yöntemlerden genetik algoritmalar yapısının basit ve uygulanmasının kolay olması nedenleriyle ön planda yer almaktadırlar (Özdağlar vd., 2006).

Genetik algoritmalar, doğal seçim ilkelerine dayanan bir arama ve optimizasyon yöntemidir. Temel ilkeleri John Holland tarafından ortaya atılmıştır.Genetik algoritmaların, fonksiyon optimizasyonu, çizelgeleme, mekanik öğrenme, tasarım, hücresel üretim gibi alanlarda başarılı uygulamaları bulunmaktadır (Liu ve Wu, 2003).

Geleneksel optimizasyon yöntemlerine göre farklılıkları olan genetik algoritmalar, parametre kümesini değil kodlanmış biçimlerini kullanırlar. Olasılık kurallarına göre çalışan genetik algoritmalar, yalnızca amaç fonksiyonuna gereksinim duyar. Çözüm uzayının tamamını değil belirli bir kısmını tararlar. Böylece, etkin arama yaparak çok daha kısa bir sürede çözüme ulaşırlar. Diğer bir önemli üstünlükleri ise çözümlerden oluşan populasyonu eş zamanlı incelemeleri ve böylelikle yerel en iyi çözümlere takılmamalarıdır.

Genetik algoritmaların geleneksel optimizasyon yöntemlerine olduğu gibi sözü edilen yapay zeka yöntemlerine göre de çeşitli alanlarda üstünlükleri bulunmaktadır. Bu üstünlükler genetik algoritmaların arama yapısı ile ilgilidir. Genetik algoritmaların arama yapısı ise, alt diziler teoremi ve yapı blokları hipoteziyle açıklanmaktadır (Min ve Cheng, 1999).

Genetik algoritmalar, klasik optimizasyon algoritmalarından dört temel noktada ayrılır:
(i) GA parametrelerin kendileri ile değil, parametre takımının kodlanmış bir haliyle uğraşırlar.
(ii) GA aramaya tek bir noktada değil, bir nokta ailesinden başlar. Dolayısıyla yerel bir optimuma takılmadan çalışabilirler.
(iii) GA amaç fonksiyonunun (objective function) türevlerini ve bir takım ek bilgileri değil, doğrudan amaç fonksiyonunun kendisini kullanırlar.
(iv) GA da deterministik değil rastlantısal geçiş kuralları kullanılır (İşçi ve Korukoğlu, 2003).

6.3 Genetik Algoritma Parametre ve Operatörleri

Genetik Algoritma tekniğinin çaprazlama olasılığı ve mutasyon olasılığı olmak üzere iki basit parametresi vardır. Çaprazlama olasılığı çaprazlamanın hangi sıklıkta yapılacağını belirtir.

Eğer hiç çaprazlama yapılmaz ise (çaprazlama olasılığı %0) yeni bireyler eski bireylerin aynısı olur ama bu yeni kuşağın eskisiyle aynı olacağı anlamına gelmez. Eğer bu oran %100 olursa yeni bireyler tamamıyla çaprazlama ile elde edilir. Çaprazlama eski bireylerden iyi taraflar alınarak elde edilen yeni bireylerin daha iyi olması umuduyla yapılır. Mutasyon olasılığı ise mutasyonun hangi sıklıkta yapılacağını belirtir. Mutasyon olmaz ise yeni birey çaprazlama veya kopyalama sonrasında olduğu gibi kalır. Eğer mutasyon olur ise yeni bireyin bir kısmı değiştirilmiş olur. Eğer bu oran %100 olursa kuşak içindeki bireyler tamamen değişir, %0 olursa hiç değişmeden kalır.

GA' ların parametreleri; çaprazlama oranı, mutasyon oranı, populasyon büyüklüğü, seçim, kodlama (encoding), çaprazlama ve mutasyon tipi gibi genel parametrelerdir. Çaprazlama oran yüksek olmalıdır. Buna karşılık mutasyon oranı çok düşük olmalıdır. Şaşırtıcı olan çok büyük populasyon büyüklüğü genellikle GA de erini arttırmaz (Çözüm bulma hızı anlamında). İyi populasyon büyüklüğü yaklaşık olarak 20-30 olmalıdır, bununla birlikte bazen 50-100 daha iyi sonuç verebilir. Seçim için genellikle rulet tekerleği kullanılır, bunun yanısıra rank seçimi, kararlı durum(steady state) ve elitizm gibi seçim yöntemleri de kullanılmaktadır (İşçi ve Korukoğlu, 2003).

GA üç temel işlemden (operatörden) meydana gelmektedir: (1) Seçim, (2) Çaprazlama ve (3) Mutasyon. GA'da her çözüm bir dizi (birey) olarak kodlanmakta ve dizilerin bir yığını ile çözüme ulaşılmaktadır. GA'nın basit formunda başlangıç yığını rassal olarak oluşturulur. Yığındaki her birey ikili düzende veya tamsayı olarak kodlanmaktadır. Bu bireyler değerlendirme aşamasında deşifre edilerek belirli amaç fonksiyonu yada fonksiyonlarında gösterdikleri performanslarına (uygunluklarına) göre değerlendirilmektedir (Uçaner ve Özdemir, 2002).

Kullanılan genetik operatörler, varolan populasyon üzerine uygulanan işlemlerdir. Bu işlemlerin amacı daha iyi özelliğe sahip yeni nesiller üretmek ve arama algoritmasının alanını genişletmektir. Farklı uygulamalarda farklı operatörler kullanılmakla birlikte genetik algoritmada 3 standart operatör kullanılır. Bu operatörler (Cochran vd., 2003):

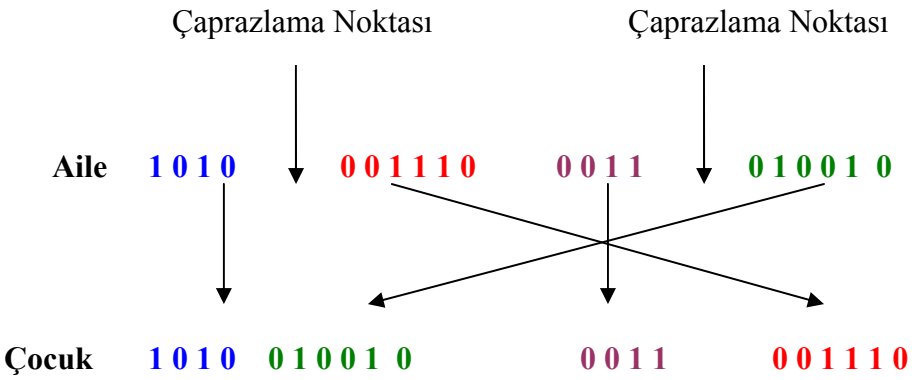
-Yeniden Üretim (Reproduction)
-Çaprazlama (Crossover)
-Mutasyon (Mutation)

6.3.1 Yeniden Üretim (Reproduction)

Nesil Üretimi (Genarational Reproduction): Varolan nesilin yerine yeniden nesil oluşturmaktır. Kararlı Durum Üretimi (Steady-State Reproduction): Sadece nesildeki birkaç birey yeni bireylerle yer değiştirir. Yeniden üretim sonucunda elde edilen ara nesil, çaprazlama ve mutasyon genetik operatörleri kullanılarak yeni nesil elde etmede kullanılır.

6.3.2 Çaprazlama (Crossover)

Çaprazlama operatörü GA lardaki en önemli operatördür. ki çözümün yapıları kullanılarak yeni bir çözüm oluşturulması esasına dayanır. Çaprazlama işlemi genel olarak ikili dizilerin parçalarının değiş tokuşu şeklinde gerçekleştirilir. Farklı uygulamalarda farklı kodlama yöntemleri kullanıldığı için farklı çaprazlama yöntemleri kullanılır, tek noktalı çaprazlama, iki noktalı çaprazlama, ve üniform çaprazlama gibi.



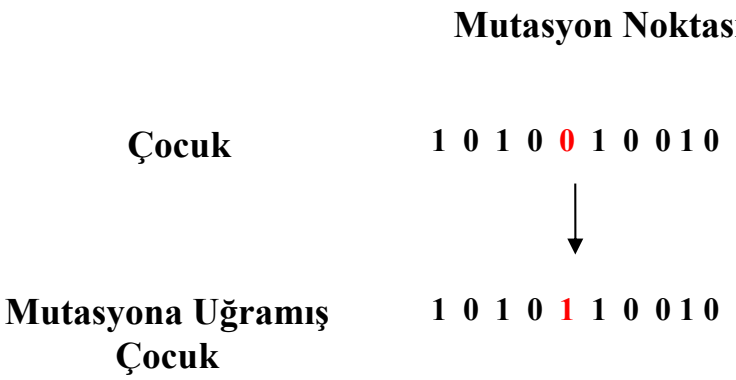
Şekil 6.2 Tek Noktalı Çaprazlama (İşçi ve Korukoğlu, 2003).

Amaç, Aile (parent) kromozom genlerinin yerini değiştirerek Çocuk (child) kromozomlar üretmek ve böylece varolan uygunluk değeri daha yüksek olan kromozomlar elde etmektedir.

6.3.3 Mutasyon (Mutation)

Mutasyon işlemi, problemin popülasyondaki çözümlerinin yerel optimuma düşmesini engellemek için kullanılır. Mutasyon yeni dölü rasgele değiştirir. Mutasyon da çaprazlama gibi şifrelemeye bağlıdır (Nabiyev, 2005).

Mutasyon GA lardaki operasyonda karar verici olarak ikinci derecede rol oynar. Amaç, varolan bir kromozomun genlerinin bir ya da bir kaçının yerlerini değiştirerek yeni kromozom oluşturmaktır. Yeniden ve sürekli yeni nesil üretimi sonucunda belirli bir süre sonra nesildeki kromozomlar birbirini tekrarlama konumuna gelebilir ve bunun sonucunda farklı kromozom üretimi durabilir veya çok azalabilir. İşte bu sebeple nesildeki kromozomların çeşitliliğini arttırmak için kromozomlardan bazıları mutasyona uğratılır.



Şekil 6.3 Tek Bir Mutasyon (İşçi ve Korukoğlu, 2003).

6.3.4 Gen Takası

Hangi şifrelemenin kullanılacağına karar verdikten sonra, çaprazlama veya gen takası adımına geçilir. Çaprazlama, genleri ebeveyn kromozomlardan seçer ve yeni bir döl oluşturur. Bu işlem rasgele çaprazlama noktası (veya noktaları) seçer ve bu noktadan (veya noktalardan) itibaren kromatidlerin (alt dizilerin) yer değişmesiyle yeni kopyaların oluşturulmasını sağlar. Kodlama türlerine bağlı olarak çaprazlama ve mutasyon işleminin farklı şekilleri bulunmaktadır.

1. İkili kodlanmış kromozomlarda 4çeşit çaprazlama gerçekleştirilmektedir.

A) Tek Noktalı Gen Takası: Bir çaprazlama noktası seçilir. Bu noktaya kadar olan bit ler (genler) birinci ebeveynden, geriye kalanlar ise diğerinden alınarak yeni bir kromozom oluşturulur (Nabiyev, 2005).

Kromozom 1: **11011** / *00100110110*

Kromozom 2: *11011* / **11000011110**

Döl 1: **11011** / **11000011110**

Döl 2: *11011*/ *00100110110*

B) Çift Noktalı Gen Takası: Burada iki kırılma noktası seçilir. İlk noktaya kadar olan bit ler birinci ebeveynden , iki nokta arasındaki bit ler ikinci ebeveynden , kalanalar ise tekrar birinci ebeveynden yani yeni kromozoma kopyalanır.

11001**011** + 11**011**111 = 11011111

C) Tek Biçimli (Üniform) Takas: bu çaprazlama biçiminde bit ler rasgele şekilde her iki ebeveynden yeni kromozoma kopyalanmaktadır.

1**10**01**011** + **1**10**111**11 = 11011111

D) Aritmetik Takas: Yeni bir kromozom oluşturmak için değişik aritmetik işlemler uygulanmaktadır.
11001011 + 11011111 = 11001001 (AND) VE işlemi (Nabiyev, 2005).

6.3.5 Seçim (Selection)

Kromozomlar çaprazlamaya ebeveyn olmak için popülasyondan seçilirler. Burada problem, kromozomların seçimi ile ilgilidir. Darwin' in evrim teorisine göre, en iyi olanlar kurtulmalı ve yeni dölü oluşturmalıdır. En iyi kromozomları seçmenin bir çok yolu vardır. Rulet tekeri seçimi (roulette Wheel selection), Boltzman seçimi (Boltzman selection), turnuva seçimi (tournament selection), sıralama seçimi (rank selection), sabit durum seçimi bunlardan bazılarıdır (Chiu vd., 2000).

Rulet Tekeri Seçimi: Burada seçilme işlemi bireylerin uygunluk değerlerine göre yapılır. Tüm bireylerin uygunluk değerleri bir tabloya yazılır ve toplanır. Sonra uygunluk değerleri, toplama bölünerek bireylerin [0,1] aralığında seçilme olasılıkları belirlenir. Sayıların hepsi bir tabloda tutulur. Seçilme olasılıklarını tuttuğumuz tablodaki sayılar birbirine eklenerek rasgele bir sayıya kadar ilerlenir. Bu sayıya ulaşıldığında ya da geçildiğinde sok eklenen sayının ait olduğu çözüm seçilmiş olur.

Sıralama Seçimi: Rulet tekeri seçimi uygunluklar çok farklıysa problemlere yol açacaktır. Örneğin, en iyi kromozomun uygunluğu, rulet tekerinin %90' ında ise, diğer kromozomlar çok az seçilme şansına sahip olacaktır. Sıralama seçimi önce popülasyonu sıralamakta ve ardından her kromozomun bu sıralamada uygunluğunu aramaktadır. En kötüsü 1 uygunlukta,

ikinci kötüsü 2 uygunlukta vb, en iyisi ise N uygunlukta olacaktır. Böylece bütün kromozomlara seçilme şansı doğacaktır.

Sabit Durum Seçimi: Burada ebeveynlerin seçimi için kromozomların büyük parçaları bir sonraki jenerasyona taşınmalıdır. Her nesilde yeni bir döl oluşturmak için birkaç kromozom seçilir. Daha sonra az uygunlukla kötü olan bazı kromozomlar atılır ve yeni döl onların yerine getirilir (Nabiyev, 2005).

6.3.6 Seçkinlik (Elitism)

Çaprazlama ve mutasyonla yeni popülayon üretildiği zaman, en iyi kromozom kaybedilecektir. Seçkinlik, önce en iyi kromozomu (yada birkaç en iyi kromozomu) yeni popülasyona kopyalar. Geri kalanı diğer yöntemlerle devam ettirilir. Seçkinlik bulunan en iyi sonucun kaybını önlediğinden GA' nın performansını oldukça hızlı bir şekilde arttırır
(Jou, 2005).

6.4 Genetik Algoritma Kullanarak Paralel Makinaların Çizelgelenmesi

Genetik algoritmanın performansını, üreme, çaprazlama ve mutasyon operatörleri ile çaprazlama ve mutasyon oranları önemli ölçüde etkilemektedir (Uçaner ve Özdemir, 2002).

6.4.1 Kodlama

Son on yılda yapılan çalışmalarda , m farklı makine ve n işten oluşan ve NP-Zor ($m > 2$ için) olarak kabul edilen akış tipi çizelgeleme problemleri için Genetik Algoritma (GA) başarılı sonuçlar vermiştir. Birbirinden farklı m -makine ve n -iş' den oluşan ve her bir işin aynı sıra ile m farklı operasyondan geçtiği, akış tipi çizelgeleme problemleri ile ilgili ilk çalışma Johnson tarafından yapılmıştır. Çok makine ($m > 2$) problemleri için son yıllarda çeşitli sezgisel yöntemler geliştirilmiştir. Bu yöntemler içinde en iyi çözüm veren arama metotlarından biri de Genetik Algoritmadır (Engin ve Fığlalı, 2002).

İşlerin makinelere atanmalarını gösteren X matrisinin her bir satırı, o satır numarasının simgelediği makineye atanan işleri gösterir. Bu satırlar, makinelerin yapacakları işleri gösteren genler ($g_1,...,g_i,...g_m$) olarak adlandırılabilir. Bir i geninin elemanlarından ($x(i,j), j=1,..,n$) değeri sıfırdan farklı olanlar, o makinenin yapacağı işleri gösterirler. Bir makinenin tamamlanma zamanı, o makinede yapılacak işlerin toplam sürelerine eşittir; bu değer, o makinede yapılacak işleri gösteren i geninin (g_i) değeri olarak tanımlanabilir

(Balin vd., 2007) :

$$f(g_i) = \sum_{j=1}^{n} x(i,j) \times t(i,j), \quad i = 1,...,m \qquad (6.1)$$

6.4.2 İlk neslin yaratılması

X matrisi, *n* tane işin, *m* tane makineye atanmasını gösterdiğine göre, her sütunda bir tane 1 olacak şekilde *x(i,j)* mantıksal değişkenlerinin değerleri rassal olarak atanırsa, bir ilk çözüm elde edilmiş olur. Aynı şekilde, birden fazla çözüm matrisi yaratmak mümkündür. Bu matrislerden her biri kromozom (*k)* olarak adlandırılır ve bir sonraki nesillerin ebeveynlerini oluştururlar. Kromozom sayısı *N* olarak belirtilir.

6.4.3 Uygunluk değerinin hesaplanması

Bir nesil yaratıldıktan sonra ilk adım, o nesildeki her kromozomun uygunluk değerini hesaplamaktır (F_k). Uygunluk değerleri, kromozomların belirttiği atamaların amaç fonksiyonu değerleri olarak kabul edilebilir. Eğer problem bir en küçükleme problemi ise uygunluk değerleri amaç fonksiyonunun tersi olarak alınmalıdır ya da α ve β iki reel pozitif reel sayı olmak üzere şöyle bir fonksiyon önerilebilir (Balin vd., 2007):

$$F_k = \alpha \times e^{-\beta \times C_{\max}(k)} \qquad (6.2)$$

6.4.4 Üreme

Uygunluk değerleri yüksek olan kromozomların yeni nesillerin yaratılmasında kullanılması toplumun evrimleşmesine olumlu katkıda bulunur. Daha önce belirtildiği üzere, ilk neslin kalitesi, GA ile elde edilecek sonucun değerine önemli etkide bulunmaktaydı. Bu son iki önermenin ışığında, kalabalık bir ilk nesil yaratılıp uygunluk dereceleri yüksek olan kromozomlar ile çözüm aramaya devam etmenin daha yararlı olacağı açıktır. Uygunluk derecesi yüksek olan nesillerin seçiminde "rulet tekerleği seçimi", "turnuva seçimi" ya da "sıralama seçimi" gibi teknikler kullanılabilir. Bu tekniklerden en çok kullanılan rulet tekerleği seçimi şu şekilde gerçekleştirilir(Balin vd., 2007) :

- k kromozomunun seçilme ihtimali P(k) olsun, $P(k) = \dfrac{F(k)}{\sum_{i=1}^{N} F(i)} \quad k = 1,...,N \qquad (6.3)$

- kromozom seçimi için *S(k)* kesim noktalarını yaratalım, $S(0) = 0$

$$S(k) = P(1) + ,..., + P(k) \quad k=1,...,N \qquad (6.4)$$

- N tane [0,1] aralığında rassal sayı yaratalım, ζ_s ve $s=1,...,N$
- Her bir ζ_s sayısı için elde edilen $S(k-1) < \zeta_s < S(k)$ eşitliği, seçilen k kromozomlarını verir.

6.4.5 Çaprazlama

Bir önceki nesilden daha iyi nitelikler içeren yeni kromozomlar yaratmak amacıyla çaprazlama işlemi yapılmaktadır. Her kromozom bir çizelgeleme önermektedir; her bir makinenin yapacağı işler, o makineyi temsil eden gende (g_i, $i=1,...m$) saklıdır. Bu işler, i geninin sıfırdan farklı değere sahip elemanları tarafından gösterilir ($x(i,j) \neq 0$). Çaprazlama, kromozomların seçilen iki geni arasında yapılır(Balin vd., 2007) :

- işleri en geç biten makineyi temsil eden genin, $\max_{i=1}^{m}\{f(g_i)\}$, en ufak işlem süresine sahip işi seçilir;
- bu iş, işlerini en erken biten makinenin, $\min_{i=1}^{m}\{f(g_i)\}$, işlerine eklenir.

Böylece işler, makinelere daha homojen olarak dağıtılmış olur; makinelerin toplam tamamlanma süresi (C_{max}) azalır.

6.4.6 Mutasyon

Çaprazlama, sadece mevcut kromozomların genlerinin değiştirilerek nesillerin iyileştirilmesini sağlar. Mutasyon ise çözüm uzayına yenilerinin eklenmesi ile genetik çeşitliliği sağlar. Bu değişiklik, neslin üyelerinden bazılarının (kromozomların, k matrislerinin) gen değerlerinin ($x(i,j)$) rassal olarak değiştirilmesi ile sağlanabilir. Bu değişiklik, k matrisinin her j sütununda sadece bir tane bulunması gereken sıfırdan farklı gen değerinin satır değiştirilmesi (işin rassal olarak başka bir makineye atanması) sureti ile gerçekleştirilebilir. Mutasyon, büyük boyutlu problemlerin çözümünde karşılaşılan kalabalık üyeli nesillerin yer aldığı algoritmalarda kullanılır. Bir sonraki paragrafta yer alan sayısal uygulamada bu operatöre ihtiyaç duyulmamaktadır (Balin vd., 2007).

Genetik Algoritmalar sadece seçim ve çaprazlama operatörlerinden meydana gelselerdi çok hızlı bir şekilde yerel bir optimuma ulaşabilirlerdi. Toplulukta çeşitlilik yaratabilmek, çaprazlama sonucunda kaybolabilen iyi özellikleri geri kazanabilmek ve genel en iyiye ulaşabilmek için bireylerdeki kodlar belli bir olasılık ile değişime (mutasyon) uğratılmaktadır (Uçaner ve Özdemir, 2002).

6.4.7 Uygunluk ölçütleri

Mutasyon işlemi gerçekleştirildikten sonra yeni nesil oluşturulur. Bu nesle ait tüm genlerin temsil ettiği makinelerin tamamlanma süreleri (*f(g$_i$)*) ve kromozomların amaç fonksiyonlarının (C_{max}) değerleri hesaplanır. Bir kromozom için, işleri en erken biten makine (*u makinesi*) ile işleri en geç biten makinenin (*v makinesi)* tamamlanma süreleri arasındaki fark, işleri en erken biten makinenin en kısa işlem zamanından daha küçük ise o kromozom için ulaşılabilecek en uygun çözüme ulaşılmış demektir. Bu uygunluk ölçütü şu şekilde yazılır :

$$f(g_v) - f(g_u) < \min_{j=1}^{m} \{ x(u, j) \} \qquad (6.5)$$

Eğer tüm kromozomlar için yukarda belirtilen uygunluk ölçütü sağlanıyorsa, hiç bir kromozomun $C_{max}(k)$ değeri iyileştirilemiyor demektir ve algoritma son bulur. Problemin sonucunu, bu değerlerin en küçük değeri verir :

$$C_{\max} = \min_{k=1}^{N} \{ C_{\max}(k) \} \qquad (6.6)$$

7. ÖNERİLEN MODELİN TANIMI

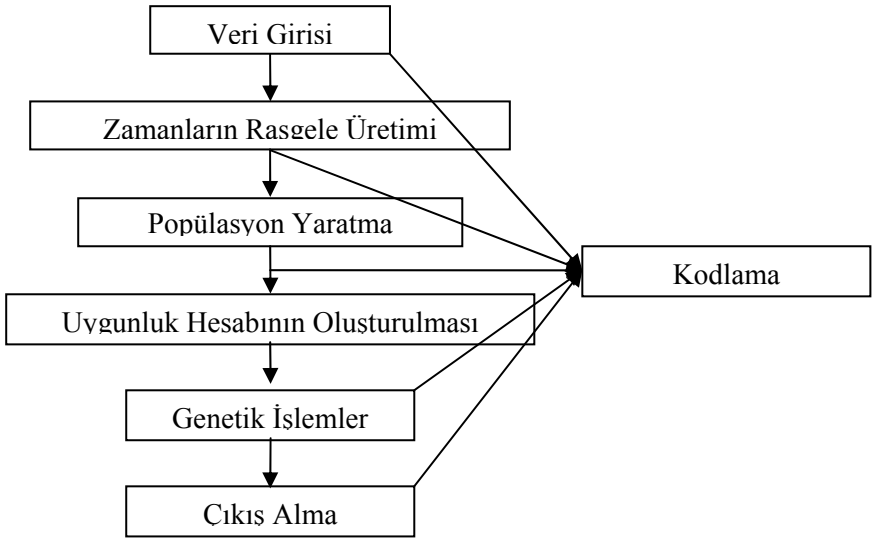
Geleneksel sıralama problemlerinde iş proses zamanları, hazırlık zamanları, teslim tarihleri gibi parametreler deterministik olarak kabul edilir. Ancak, gerçek-dünya durumlarında bu parametreler sık sık belirsiz durumlarla karşılaşmaktadır. Dolayısıyla, sıralama problemleri temel olarak iki kategoriye ayrılmaktadır: deterministik sıralama ve kesin olmayan sıralama (stokastik, bulanık, vb.).

Bu şekilde sıralama probleminde çeşitli faktörler gerçek dünyaya formüle edileceği zaman sık sık kesin olmayan, belirsiz durumlarla karşılaşılmaktadır. Bu durum insan nedenli sebeplerle özellikle oluşmaktadır. Bu tip durumlarda bulanık proses zamanlarının, bulanık teslim tarihlerinin ve bunun gibi parametrelerin düşünülmesinin çok daha uygun olacağı gözükmektedir. Dahası, pek çok çalışma bulanık sıralama problemleri üzerinde yoğunlaşmaktadır. Bulanık sıralama konusundaki ilk çalışma 1979 yılında yapılmıştır. Ishii ve arkadaşları bulanık teslim tarihleri ile sıralama problemlerini ilk olarak tartışmışlardır (Peng ve Liu, 2004).

Bulanık sıralama problemleri için literatürde gösterilen üç temel yaklaşım bulunmaktadır: klasik dağıtım kurallarını direkt olarak bulanıklaştırma, bulanık sıralamayı ve bulanık üstün ilişki metotlarını kullanarak sıralama ve genetik algoritma (GA), benzetimli tavlama (SA), tabu arama (TS) yöntemlerini içeren sezgisel dağılım metotları ile optimal çizelgelere karar verilmesi için matematiksel programlama modellerinin çözülmesi (Peng ve Liu, 2004).

Bu çalışmada öncelikle proses zamanları rassal olarak seçilmiş, ardından benzer olmayan paralel makineler çizelgeleme problemine uyarlanmıştır. İlk olarak 9 iş 4 makine problemi ele alınarak daha sonra bulanık alınan proses zamanları genetik algoritmaya dökülmüştür. Ardından aynı şekilde 10 makine 30 iş problemi incelenecektir. Program olarak "JAVA Eclipse Europa" programı kullanılarak, genetik algoritmanın kodlama kısmı yerine getirilmiş ve ardından genetik algoritma uygulanmıştır.

İki adet problem örneği paralel makineler üzerinde incelenmektedir. Bunlardan ilki 9 iş 4 makine problemi için yapılmaktadır. Kurulan model ise şöyledir:



Şekil 7.1 Modelin Tanımı

Modelleme içerisinde yapılan işlemler sırası ile şöyledir;

- ✓ Veri Girişi
- ✓ Zamanların Rasgele Üretimi
- ✓ Popülasyon Yaratma
- ✓ Uygunluk Hesabının Oluşturulması
- ✓ Genetik İşlemler
- ✓ Çıkış Alma

Bu çalışmada öncelikle 4 makine 9 iş problemi incelenmiştir. Bu problemde popülasyon büyüklüğü 50, mutasyon oranı ise 1/1000 alınmıştır. Ardından 10 makine 30 iş problemi ele alınmıştır. Burada ise popülasyon büyüklüğü 100, mutasyon oranı da yeniden 1/1000 alınmıştır. Genelde bu tip sıralama problemlerinde mutasyon oranları 1/1000 alınmaktadır.

7.1. Modelin Metodolojisinin Açıklanması

Veri Girişi: İlk yapılan işlem "veri girişi" işlemidir. Program ilk açıldığında belirlenen "çalışma alanı" klasörünün içinden uygun proje seçilmelidir. Bu projeler "*pelin*" klasöründe alt alta dizili bulunmaktadır. Burada "*pelin*" dosyasının altındaki dosyanın adı "src" dosyasıdır. Programın akışı bu dosyadan ilerleyecektir. Alt sınıflar "*default package*" denilen alt dosyadan açılmaktadırlar. Buna göre alt sınıflar (classes) şunlardır;

- Genetik Java
- Hesapla Uygunluk Java
- Main Java
- Program Akışı Java
- Veri Giriş 1 Java
- Yarat Popülasyon Java
- Yarat Zaman Java

Program açılarak yazılan kodlardan ilk matriks ortalama zamanları, ikincisi ortalamadan ne kadar düşük olacağını (minimum değerler) ve üçüncüsü ise ortalamadan ne kadar yüksek olacağı (maksimum değerler) ile ilgili değerleri vermektedir. *Veri Giri1.Java* denilen butona basılınca çıkan ekranda kodlar yazılmaktadır.

Zamanların Rasgele Üretilmesi: Burada çeşitli algoritmalar kullanılmaktadır. Bu algoritmalara internet ortamından ulaşılmaktadır. Bilgisayar ortamında *http://www.iro.umontreal.ca/~simardr/ssj/indexe.html* (SSJ: A Java library for a stochastic simulation API specification- SSJ stokastik simülasyon için Montreal Üniversitesi' nde geliştirilmiş bir Java kütüphanesidir.) adresinden her türlü algoritma (simülasyon algoritmaları) rahatlıkla bulunabilmektedir. Burada "üçgensel dağılım" sağlayan kütüphaneden yararlanılacaktır. Java API denilen referans dökümantasyonundan genetik algoritmaya uygun kodlar kullanılmaktadır.
(http://www.iro.umontreal.ca/~simardr/ssj/doc/html/index.html).

Zamanlar dizisi içine üçgensel dağılıma uygun (yukarıdaki adresten yararlanılarak) zamanlar yerleştirilmektedir. Alt, üst ve orta değerleri hesaplanan üçgensel dağılımla oluşturulmuş kümeler teker teker algoritmada yazılarak bir sonuç değeri oluşturulmaya çalışılmaktadır. Burada "*random stream* (rasgele akım)" türünde bir obje yaratılmıştır. Bu objeye bağlı olarak ise "*MRG31k3p*" isimli algoritma kullanılmış ve üçgensel dağılımla uygun sayılar üretilmiştir. Bu algoritma internet üzerinden bulunan kütüphanede yer almaktadır.

Popülasyon Yaratma: Makine sırası "*ArrayList* (Java' da kullanılan bir liste türüdür)" kullanılarak, genetik algoritmada liste türüne döndürülmektedir. Bu liste sayesinde hiç birşey ikinci kere yazılmamaktadır. Ayrıca liste türüne döndürülmesi rasgele sıra türetme imkanı

sağlamaktadır. Bu işlem "*shuffle*" koduyla tekrarlama işlemini sağlayarak popülasyonların daha net bir şekilde ve birbirinden ayrı bir şekilde yaratılmasını sağlamaktadır. İlk bireyler böylece rasgele sıralanmış olarak kromozomlara dağıtılmaktadır. Daha sonra bu kromozomlar dizi türüne çevrilerek işlem yapılmasına olanak sağlanmaktadır.

***Uygunluk Hesabının Oluşturulması*:** Öncelikle kromozom hattından tüm sıralar 1. makineye atanır. Daha sonra atanan sıradaki son işlem 2. makineye atanır. Eğer 1. makinanın toplam işlem süresinden çıkartılan işlemin süresi 2. makinedeki toplam iş süresi ve yeni atanan işlemin iş süresinin toplamından büyükse, atama gerçekleşmektedir. Böylece son iterasyona kadar bu mantıkla makineler arasında atanamayacak iş kalıncaya kadar devam edilmek zorunda kalınacaktır.

Genetik İşlemler: İlk popülasyon oluştuğunda kromozomlardaki uygunluk değerine bakılarak, bu popülasyondaki en iyi %10 kromozom korunmaktadır. Bu oran diğer nesillere direkt aktarılmaktadır. Geriye kalan %90 lık kesim çaprazlama yapılarak, diğer nesile aktarılır. Bu işlem "elitizm" olarak adlandırılmaktadır. Bu şekilde en iyi kromozomlar korunarak uygun bireylerin oluşması için çaprazlama ortamı sağlanmaktadır.

Uygulamanın her ikisinde de kromozomlar seçilirken "*turnuva seçim kuralı*" uygulanmıştır. Turnuva Seçim Yönteminde yerine koyarak ya da koymadan rastgele *t* adet birey seçilir ve bu büyüklüğe turnuva genişliği adı verilir. Bu gruptaki en iyi birey, yeni popülasyona kopyalanır. Bu işlem kullanıcı tarafından önceden kararlaştırılan çevrim sayısı kadar tekrarlanır. Kromozomlar çaprazlanırken "*iki noktalı çaprazlama metodu*" kullanılmıştır. İki nokta çaprazlamada iki nokta arasında kalan alt dizilerin değiştirilmesiyle iki yeni birey elde edilir.

Çıkış Alma: Bu işlemle her bir iterasyondaki en iyi sıra ve uygunluk değerinin, uygun komutla konsolda görünmesi sağlanmaktadır. Böylelikle genetik algoritma işlemi tamamlanmaktadır. Sonuç olarak uygun çıktılar konsolda gözükmektedir. Böylelikle genetik algoritma sonlandırılmaktadır.

7.2 Modelin Sayısal Uygulaması

Bu kısımda, m benzer olmayan (non-identical) makine üzerinde n tane iş sıralama problemi, GA kullanılarak çözülmeye çalışılmıştır. Sayısal bir örnek olarak 9 iş 4 benzer-olmayan

makine üzerinde sıralanmıştır. Amaç, tamamlanma zamanını (makespan - $C_{\max}$) minimize etmektir. Makinelerin hızları $(V_1, V_2, V_3, V_4) = (1,2,3,4)$ olarak verilmiştir. Problem verileri Çizelge 7.1' deki gibi özetlenmektedir.

Çizelge 7.1 İşlerin proses zamanları

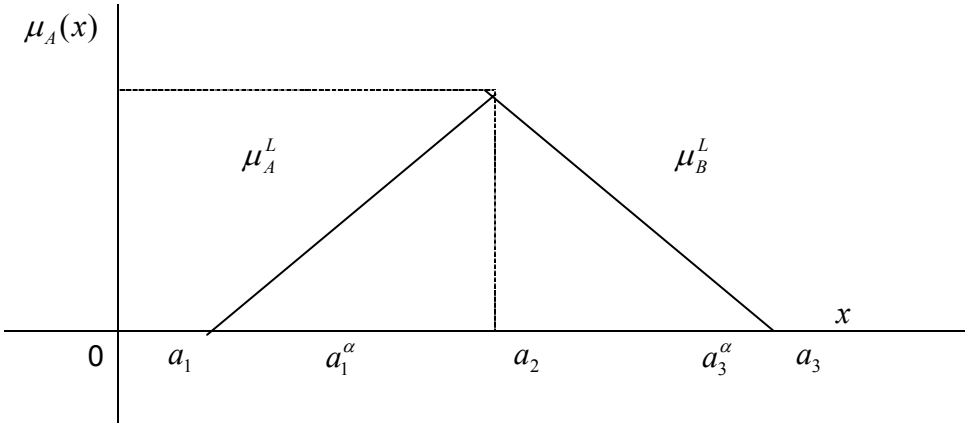
Proses Zamanları (min.)	***1. iş***	***2. iş***	***3. iş***	***4. iş***	***5. iş***	***6. iş***	***7. iş***	***8. iş***	***9. iş***
Makine 1 (M.1)	17,0	13,0	23,0	31,0	15,0	19,0	21,0	25,0	13,0
Makine 2 (M.2)	7,0	8,0	12,0	15,0	9,0	10,0	11,0	14,0	6,0
Makine 3 (M.3)	4,5	5,5	6,0	7,5	3,0	4,0	6,5	3,5	4,5
Makine 4 (M.4)	4,6	3,7	5,8	7,0	2,2	5,0	3,4	5,3	2,7

Bulanık sayıların pek çok tanımlaması literatrürde bulunabilir. Burada iş proses zamanları üçgensel bulanık proses zamanlarına dönüştürülmüştür. Her iş bulanık bir proses zamanına sahiptir ve bulanık proses zamanları ise üçgensel bulanık sayılardır. Üyelik fonksiyonu $\mu_A(x)$ şöyle tanımlanmaktadır.

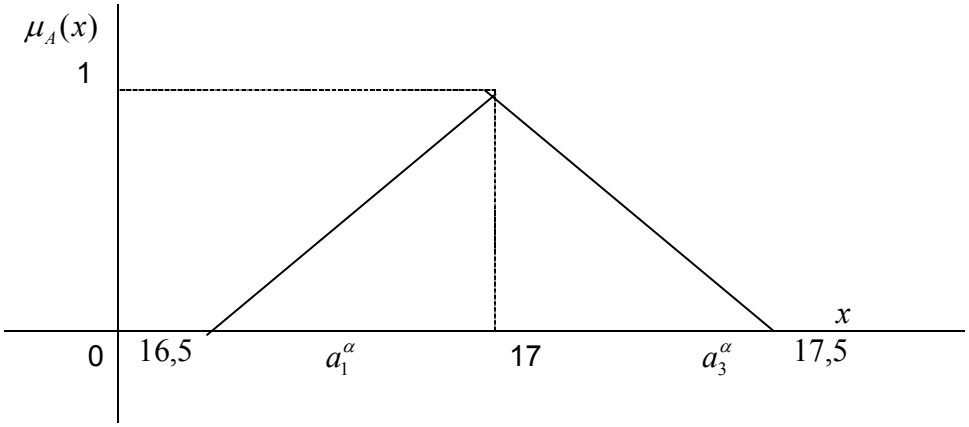
$\mu_A(x)$ üçgensel üyelik fonksiyonu, şöyle tanımlanmaktadır:

$$\mu_A(x) = \begin{cases} 0, & x < a_1 \\ \dfrac{x - a_1}{a_2 - a_1}, & a_1 \le x \le a_2 \\ \dfrac{a_3 - x}{a_3 - a_2}, & a_2 \le x \le a_3 \\ 0, & x > a_3 \end{cases}$$

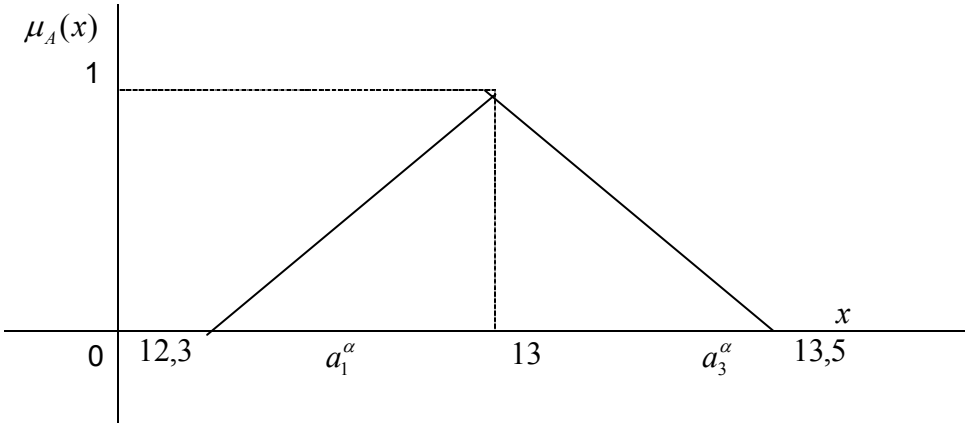
formülüne göre küme, $A = (a_1, a_2, a_3)$ olmalıdır. Burada a_2 normal değerli üyelik olarak tanımlanabilir. Bulanık Mantık bu noktada bir α katsayısına bağlı olarak a_2' ye yakın değerlerin, bu değere yüklenen anlam ile temsil edileceğini varsaymaktadır. Diğer bir deyişle a_2' deki belirsizlik, varsayılacak ya da dağılıma göre bulunabilecek bir α katsayısı ile tolere edilebilir.



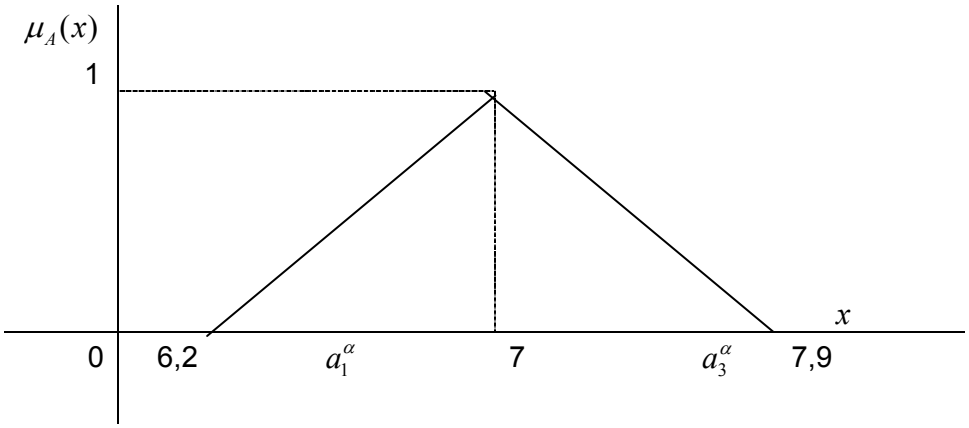
Şekil 7.2 Üçgensel bulanık sayıların gösterimi, sayıların komşuluğu



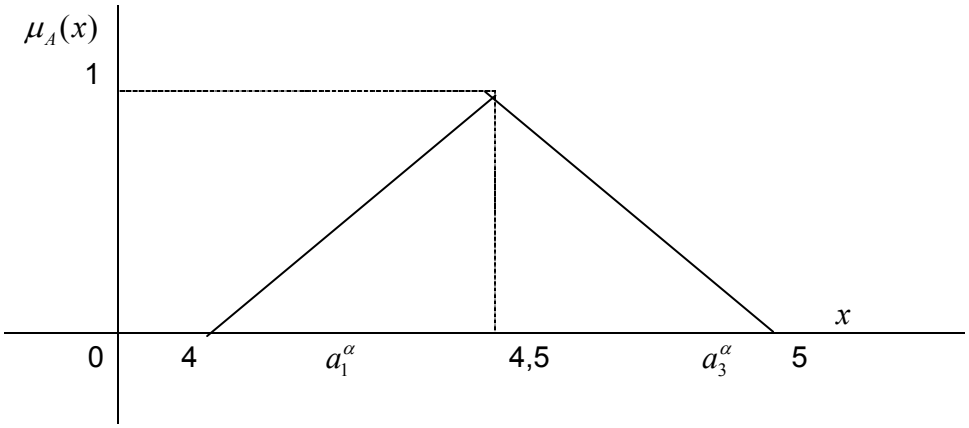
Şekil 7.3 Birinci makineye atanan birinci işin üçgensel bulanık gösterimi



Şekil 7.4 Birinci makineye atanan ikinci işin üçgensel bulanık gösterimi



Şekil 7.5 İkinci makineye atanan birinci işin üçgensel bulanık gösterimi



Şekil 7.6 Üçüncü makineye atanan birinci işin üçgensel bulanık gösterimi

Bütün bu sayısal komşuluk gösterimleri diğer tüm işler ve makineler için yapılır. Sayısal bulanık aralıklar rasgele atanmaktadır.

Çizelge 7.2 İşlerin bulanık proses zamanları.

Proses Zamanları (min.)	***1. iş***	***2. iş***	***3. iş***	***4. iş***	***5. iş***	***6. iş***	***7. iş***	***8. iş***	***9. iş***
Makine 1 (M.1)	(16.5,17,17.5)	(12.3,13,13.5)	(22.5,23,23.4)	(30.7,31,31.6)	(14.6,15,15.7)	(18.5,19,19.9)	(20.5,21,21.4)	(24.3,25,25.6)	(12.6,13,13.5)
Makine 2 (M.2)	(6.2, 7, 7.9)	(7.8, 8, 8.4)	(11.8,12,12.4)	(14.3,15,15.6)	(8.5, 9, 9.4)	(9.7, 10, 10.2)	(10.4,11,11.6)	(13.4,14,14.6)	(5.6, 6, 6.5)
Makine 3 (M.3)	(4, 4.5, 5)	(5, 5.5, 6)	(5.8, 6, 6.9)	(7, 7.5, 8)	(2.4, 3, 3.6)	(3.7, 4, 4.6)	(6, 6.5, 7)	(2, 3.5, 4)	(4, 4.5, 5)
Makine 4 (M.4)	(4.2, 4.6, 5)	(3.5, 3.7, 4)	(5, 5.8, 6)	(6.5, 7, 7.5)	(2, 2.2, 2.7)	(4.5, 5, 5.5)	(3, 3.4, 4)	(5, 5.3, 5.8)	(2, 2.7, 3)

- İşlerin proses zamanları Çizelge 7.2’ de bulanık olarak alınmıştır.
- Çizelge 7.1’ deki değerler tamamen rassal alınmıştır.
- Bulanık üçgensel sayılar oluşturularak Genetik Algoritma uygulamasına gidilmiştir.

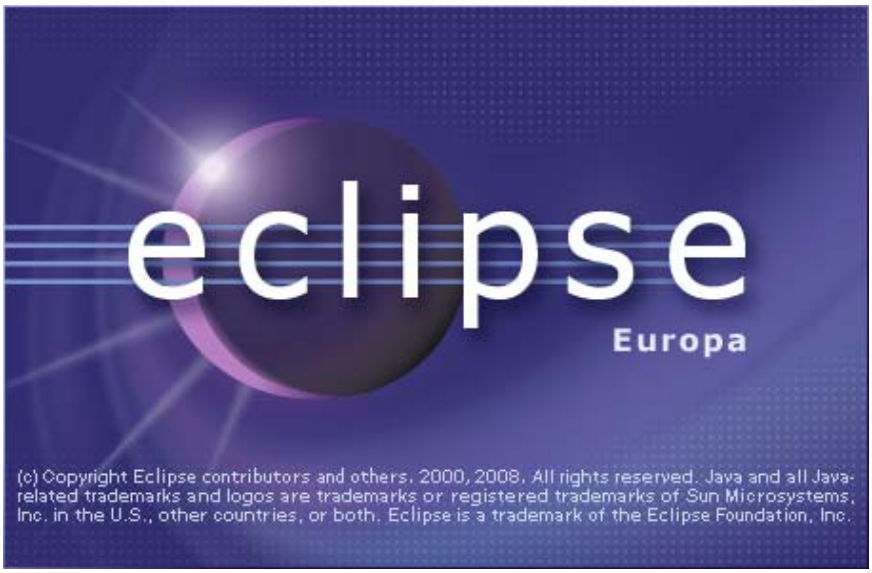
Bulunan bulanık aralıksal değerlerin ardından “JAVA Eclipse” programına geçilir. Bu programla ilgili en önemli aşama ise genetik algoritmanın da en önemli noktalarından birisi olan kodlamanın yapılmasıdır. Şimdiki bölümde Eclipse Europa programı anlatılacak ve gerekli uygulamalar yapılacaktır.

7.2.1 Eclipse Programı

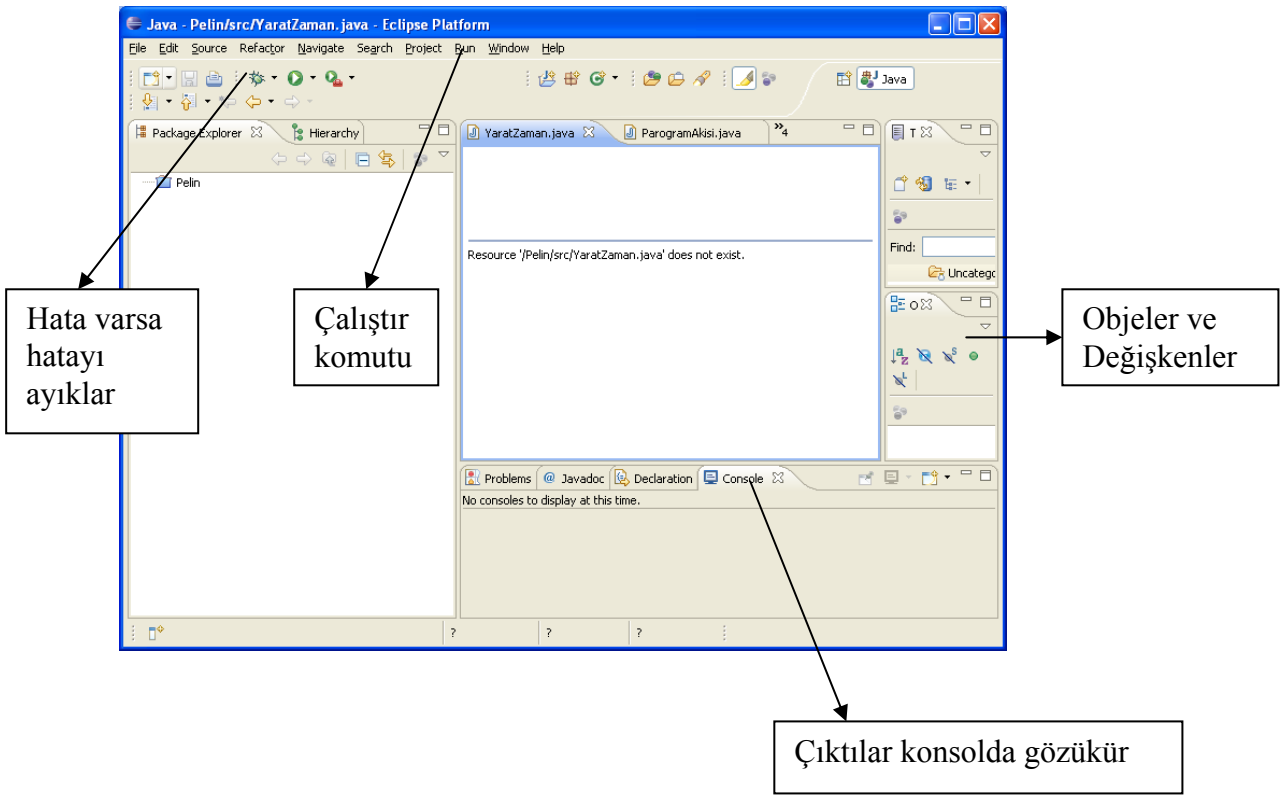
Birçok eklentinin bir araya gelmesinden oluşan yapısı sayesinde Eclipse, hem çok hızlı geliştirilmekte olan, hem de son derece dinamik bir program olma özelliğine sahiptir. İlk tasarlandığı zamanlarda basit bir IDE (integrated development environment) olan Eclipse, günümüzdeki 3.0 versiyonuyla birlikte artık her türlü kullanıma uygun bir "Rich Client Platform" haline geldi ve başka IDE'lerin yaratılması için Java dünyasının en çok tercih edilen platformlarından oldu. Burada "*Rich Client Platform*" özelliği yazı yazılan dökümanları temsil eden bir özelliktir. Eclipse'i kullanmaya başlamak için ana sayfası olan "http://www. eclipse.org" www.eclipse.org 'a girip "*downloads*" bölümünden bilgisayara indirmek gerekmektedir.

Eclipse'in en yararlı özelliklerinden biri kod yazımını fark edilir derecede hızlandıran "*Quick Fix*" özelliğidir. Bu özellikle yanlış yapılan seçenekler arka arkaya sıralanır ve hatalar uygun bir şekilde düzeltilir. Sadece derleme (*compile*) ettikten sonra çıkan hataları değil, yazım esnasında gördüğü eksiklikleri de düzeltebilmesiyle kullanıcıya büyük kolaylık sağlamaktadır.

Satırın solunda görünür hale gelen ampul işareti, o satır ile ilgili bir düzeltme yapılabileceğini işaret etmektedir. Farenin sol tuşuyla ampul ikonuna tıklayarak ya da "Ctrl+1" kısa yolunu kullanarak yapılabilecek düzeltmelerin listelendiği "*Quick Fix*" penceresine erişilebilir. Satırın altında beliren pencerede seçilen düzeltmenin önizlemesi de görülebilmektedir. Bu sayede gerekli düzenlemeyi bulmak için deneme yanılma yönetimine başvurmaya gerek kalmamaktadır.



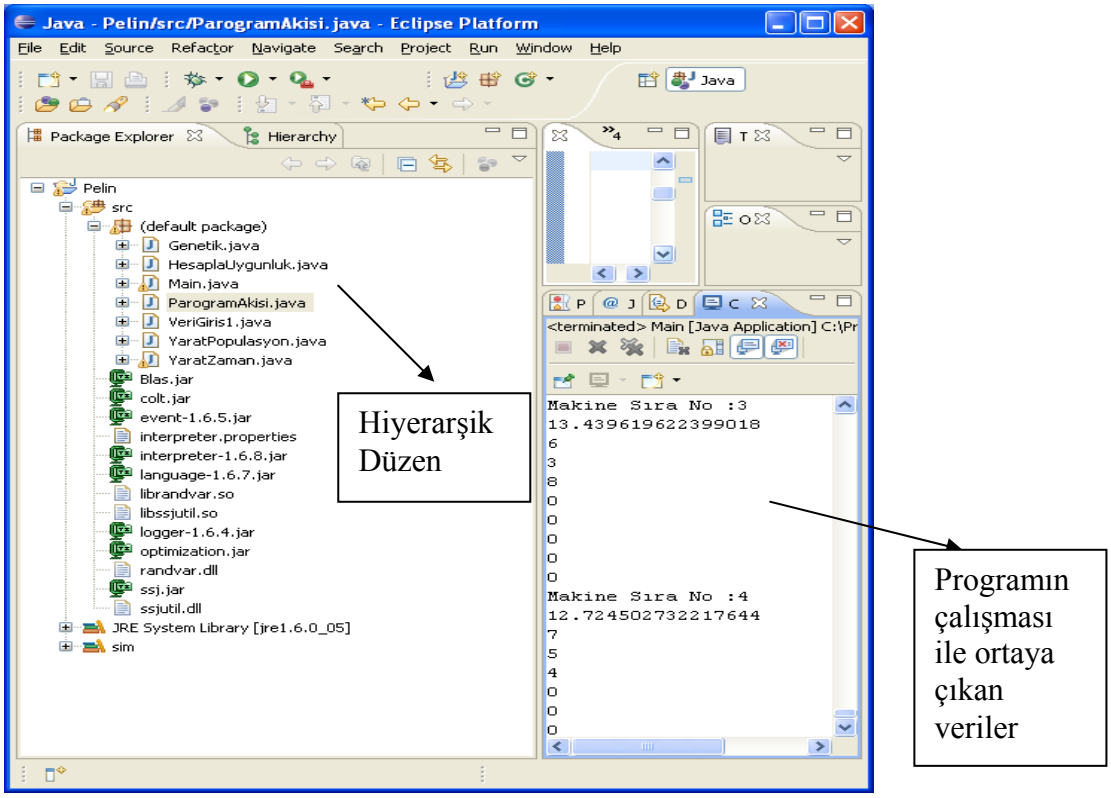
Şekil 7.7 Kullanılan Java Programı – Eclipse Europa



Şekil 7.8 Boş Java Dosyası

Açık kod olduğu için, dünyadaki her programcının Eclipse ile uyumlu bir şekilde istediği eklentiyi geliştirme hakkına sahip olması ve programın zaten iç içe geçmiş eklentilerden oluşan yapısı sayesinde yeni eklentilerin kolayca yüklenebiliyor ve kullanılabiliyor olması bu

dinamik programın en büyük avantajıdır. Açık kod özelliği, kodların nasıl derlendiğini göstermektedir. Windows' ta bu özellik bulunmamaktadır.



Şekil 7.9 Java dosyası içinde bulunan hiyerarşik genetik algoritma işlemleri

7.2.2 Dört Makina Dokuz İş için Genetik Algoritma Uygulaması

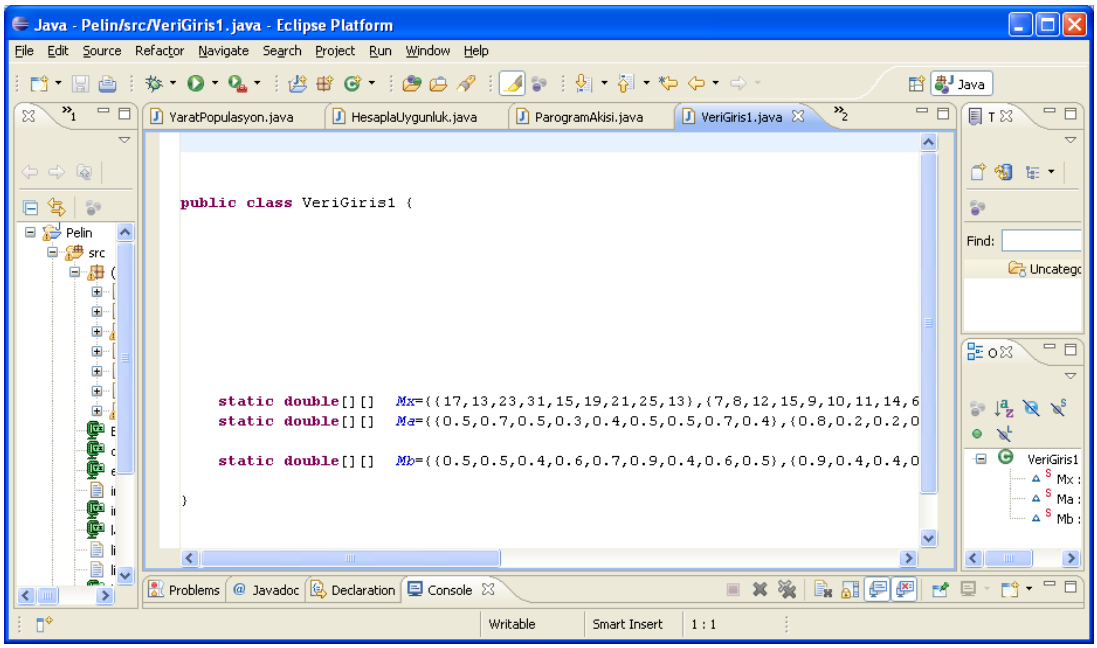
7.2.2.1 Veri Girişi

İlk olarak veri girişi yapılmalıdır. Program ilk olarak açıldığında belirlenen "work space (çalışma alanı)" klasörünün içinden uygun proje seçilir. Bu projeler "*pelin*" projesinde alt alta dizilir. Burada "*pelin*" dosyasının altındaki dosyanın adı "*src*" dir. Bu bir kaynak dosyasıdır. Aslında programın akışı bu dosyadan ilerleyecektir. Bu sınıflar "*default package*" denilen alt dosyadan açılmaktadırlar. Buna göre alt sınıflar (classes) şunlardır;

- Genetik Java
- Hesapla Uygunluk Java
- Main Java
- Program Akışı Java
- Veri Giriş 1 Java
- Yarat Popülasyon Java

- Yarat Zaman Java

Java kod dilinde öncelikle yüklem ardından nesne veya özne gelmektedir. Bu javaya has bir özellik olmaktadır. Bu kodlarında baş harfleri büyük, diğerleri küçük yazılmaktadır. Program açılarak yazılan kodlardan ilk matriks ortalama zamanları, ikincisi ortalamadan ne kadar düşük olacağını (minimum değerler) ve üçüncüsü ise ortalamadan ne kadar yüksek olabileceği (maksimum değerler) ile ilgili değerleri vermektedir. Bu değerler ile ilgili yazılan kodlamalar aşağıda görülmektedir. "*Veri Giri1.Java*" denilen butona basılınca çıkan ekranda kodlar yazılmaktadır.



Şekil 7.10 Yapılan veri girişi ile ilgili bilgisayar ekranı

```
public class VeriGiris1 {

        static double[][]
Mx={{17,13,23,31,15,19,21,25,13},{7,8,12,15,9,10,11,14,6},{4.5,5.5,6,7.5,3,4,6.5,3.5,4.5},{
4.6, 3.7,5.8,7.0,2.2,5.0,3.4,5.3,2.7}};

        static double[][]
Ma={{0.5,0.7,0.5,0.3,0.4,0.5,0.5,0.7,0.4},{0.8,0.2,0.2,0.5,0.5,0.3,0.6,0.6,0.4},{0.5,0.5,0.2,0.5,
0.6,0.3,0.5,1.5,0.5},{0.4,0.2,0.2,0.5  ,0.2,0.5,0.4,0.3,0.7}};
```

```
        static double[][]
Mb={{0.5,0.5,0.4,0.6,0.7,0.9,0.4,0.6,0.5},{0.9,0.4,0.4,0.6,0.4,0.2,0.6,0.6,0.5},{0.5,0.5,0.9,0.5,
0.6,0.6,0.5,0.5,0.5},{0.4,0.3,0.2,0.5  ,0.5,0.5,0.6,0.5,0.3}};
}
```

Burada *Mx* değerleri örneğimizin ana değerlerini göstermektedir. *Ma* ve *Mb* değerleri ise ana değerlerden meydana gelen sapma değerleridir. Kodlama bitirilirken "}" işareti java kod dizininde kullanılmaktadır.

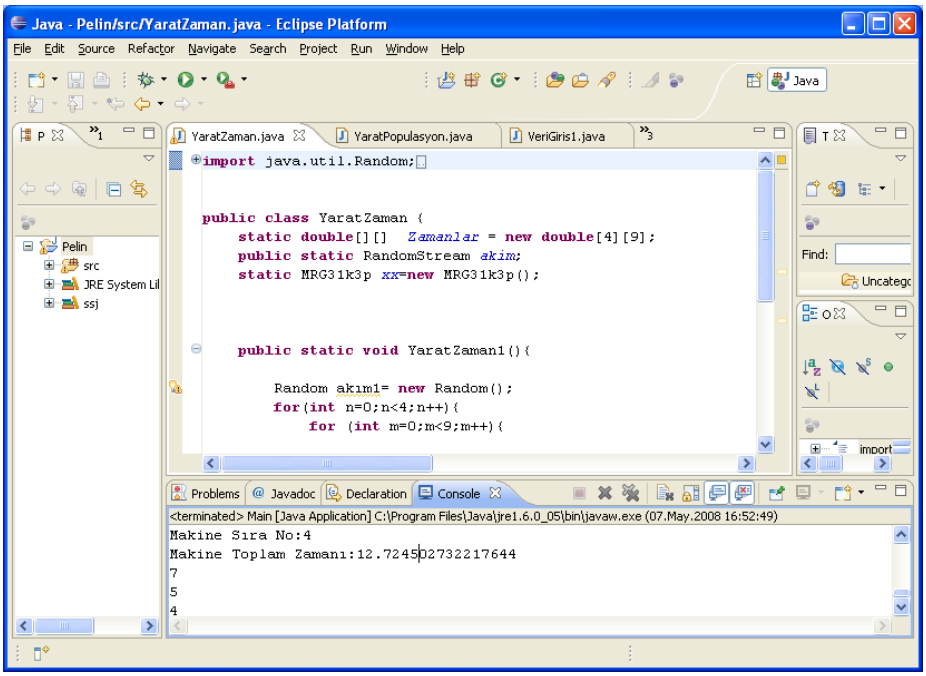
7.2.2.2 Zamanların Rasgele Üretilmesi

Belki de en karışık noktalardan birisidir. Çünkü burada çeşitli algoritmalar kullanılacaktır. Bu algoritmalara ise internet ortamından ulaşılacaktır.

Bilgisayar ortamında http://www.iro.umontreal.ca/~simardr/ssj/indexe.html (*SSJ*: A Java library for a stochastic simulation API specification- SSJ stokastik simülasyon için Montreal Üniversitesi' nde geliştirilmiş bir Java kütüphanesidir.) adresinden her türlü algoritma (simülasyon algoritmaları) rahatlıkla bulunabilmektedir. Burada "üçgensel dağılım" sağlayan kütüphaneden yararlanılmaktadır. Java API denilen referans dökümantasyonundan genetik algoritmaya uygun kodlar kullanılmıştır.
(http://www.iro.umontreal.ca/~simardr/ssj/doc/html/index.html).

Önceki sayfalarda verilen zamanlara bağlı olarak zamanlar dizisi içine üçgensel dağılıma uygun (yukarıdaki adresten yararlanılarak) zamanlar yerleştirilmiştir. Alt, üst ve orta değerleri hesaplanan üçgensel dağılımla oluşturulmuş kümeler teker teker algoritmada yazılarak bir sonuç değeri oluşturulmaya çalışılır.

Burada "*random stream (rasgele akım)"* türünde bir obje yaratılmıştır. Bu objeye bağlı olarak ise "*MRG31k3p*" isimli algoritma kullanılmış ve üçgensel dağılımla uygun sayılar üretilmiştir. Bu algoritma internet üzerinden bulunan kütüphanede bulunmaktadır.



Şekil 7.11 Zamanların yaratılması ile ilgili bilgisayar ekranı

Bu noktada yazılan kodlamalar ise aşağıda sıralanmaktadır;

```
import java.util.Random;
import umontreal.iro.lecuyer.probdist.TriangularDist;
import umontreal.iro.lecuyer.randvar.TriangularGen;
import umontreal.iro.lecuyer.rng.MRG31k3p;
import umontreal.iro.lecuyer.rng.RandomStream;

public class YaratZaman {
static double[][]  Zamanlar = new double[4][9];
        public static RandomStream akim;
        static MRG31k3p xx=new MRG31k3p();
        public static void YaratZaman1(){

                Random akım1= new Random();
                for(int n=0;n<4;n++){
                        for (int m=0;m<9;m++){
                TriangularDist ok= new TriangularDist((VeriGiris1.Mx[n][m]-
```

```
VeriGiris1.Ma[n][m]),(VeriGiris1.Mx[n][m]+VeriGiris1.Mb[n][m]),VeriGiris1.Mx[n][m]);

        TriangularGen ok15= new TriangularGen(xx, ok);
        Zamanlar[n][m]=ok15.nextDouble();
        System.out.println(Zamanlar[n][m]);
            }
```

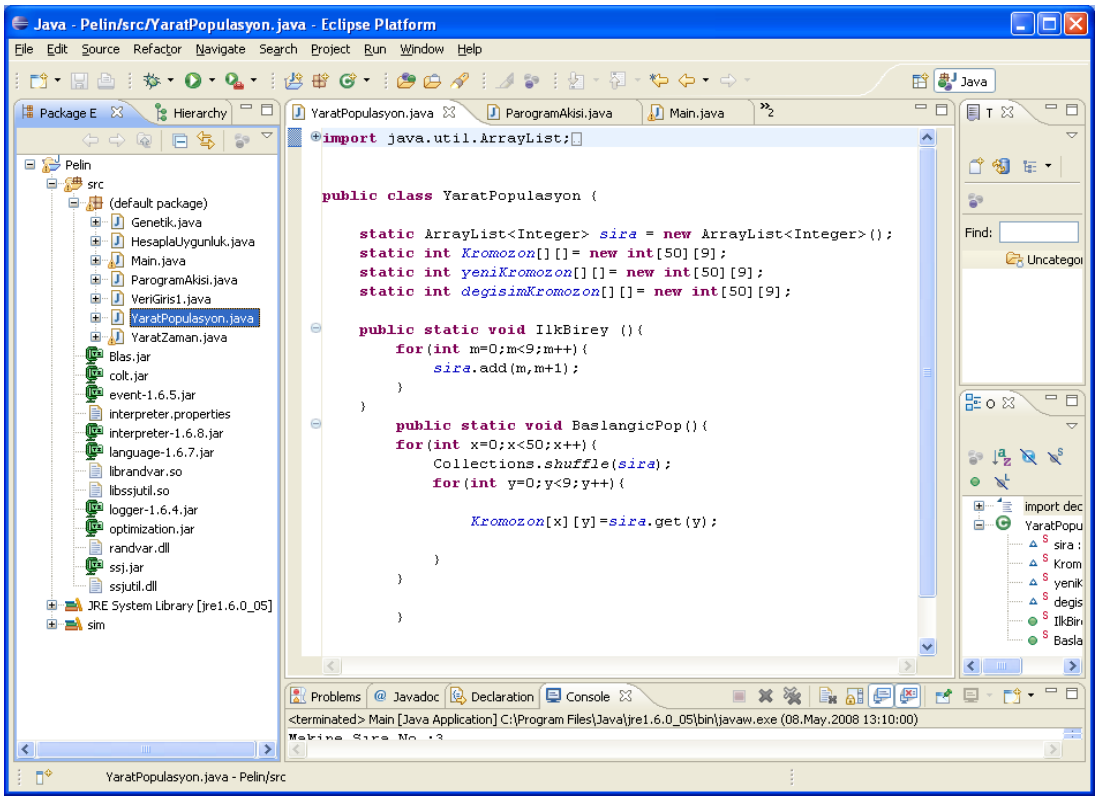
Burada "*public class*" ile yeni bir sınıf yaratılmaktadır. Java kod diline göre bir isimlendirmedir. Zamanlara yeni bir değer atanmaktadır (*new double*). Buradaki 4 ve 9 sayıları ise makine ve iş sayılarını göstermektedir. İnternet üzerinden bulunan kütüphaneden yararlanılarak "*random stream (rasgele akım)*" denilen obje belirtilmektedir.

Ayrıca bu kütüphaneden "*MRG31k3p*" denilen algoritma alınarak kullanılmaktadır. "*ok15*" denilen rasgele bir değişken, üçgensel sayılara bağlı olarak veri girişleri yapıldıktan sonra atanmaktadır. Bu değişken ismi istenilen bir şekilde atanabilir. "*m*" ve "*n*" değerleri sırasıyla iş ve makine sayılarını göstermektedir. Üçgensel dağılıma uygun zamanlar (*Zamanlar*) matrisine yazılmıştır yani kopyalanmıştır. Kodlama bitirilirken "}" işareti java kod dizininde kullanılmaktadır.

7.2.2.3 Popülasyon Yaratma

Makine sırası "*ArrayList (liste)*" (Java kod dizininde kullanılmaktadır) kullanılarak, genetik algoritmada liste türüne döndürülmüştür. Bu liste hiç birşeyin ikinci kere yazılmasına izin vermemektedir. Liste türüne döndürülmesi rasgele sıra türetme imkanı sağlamaktadır. Bu işlem "*shuffle*" koduyla tekrarlama işlemini sağlayarak, popülasyonların daha net bir şekilde ve birbirinden ayrı bir şekilde yaratılmasını sağlamaktadır.

İlk bireyler böylece rasgele sıralanmış olarak, kromozomlara dağıtılmaktadır. Daha sonra bu kromozomlar dizi türüne çevrilerek işlem yapılmasına olanak sağlanmaktadır.



Şekil 7.12 Yarat Popülasyon kod dizininin gösterimi

Yarat popülasyon kod dizini ise şöyle verilmektedir:

```
import java.util.ArrayList;
import java.util.Collections;
public class YaratPopulasyon

static ArrayList<Integer> sira = new ArrayList<Integer>();
        static int Kromozon[][]= new int[50][9];
        static int yeniKromozon[][]= new int[50][9];
        static int degisimKromozon[][]= new int[50][9];

public static void IlkBirey (){
            for(int m=0;m<9;m++){
                sira.add(m,m+1);
```

```
public static void BaslangicPop(){
        for(int x=0;x<50;x++){

Collections.shuffle(sira);
            for(int y=0;y<9;y++){
                Kromozon[x][y]=sira.get(y);
            }
```

Burada "*static int*" değeri içinde kromozom özellikleri atanmaktadır. Bunlardan 50 değeri popülasyon büyüklüğünü, 9 ise iş sayısını vermektedir. "*shuffle*" kodu ile daha öncede bahsedildiği gibi kromozomlara değer atanmaktadır. Kodlama bitirilirken " }" işareti java kod dizininde kullanılmaktadır.

7.2.2.4 Uygunluk Hesabının Oluşturulması

Öncelikle kromozom hattından tüm sıralar 1. makineye atanır. Daha sonra atanan sıradaki son işlem 2. makineye atanır. Eğer 1. makinanın toplam işlem süresinden çıkartılan işlemin süresi 2. makinedeki toplam iş süresi ve yeni atanan işlemin iş süresinin toplamından büyükse, atama gerçekleşir.

Böylece son iterasyona kadar bu mantıkla makineler arasında atanamayacak iş kalıncaya kadar devam edilir.

Çizelge 7.3 İşlerin anlatılan yöntemle makinelere atanması

	1. iş	**2. iş**	**3. iş**	**4. iş**	**5. iş**
1. Makine	17.13621741	13.01888095	22.72256488	31.16449805	15.00140835
2. Makine	7.42249719	8.033658058	11.88192769	14.85940375	9.030038543
3. Makine	4.489361494	5.448947471	6.258963435	7.500446588	2.819597217
4. Makine	4.406990018	3.576886047	5.913337355	6.950689877	2.522155308

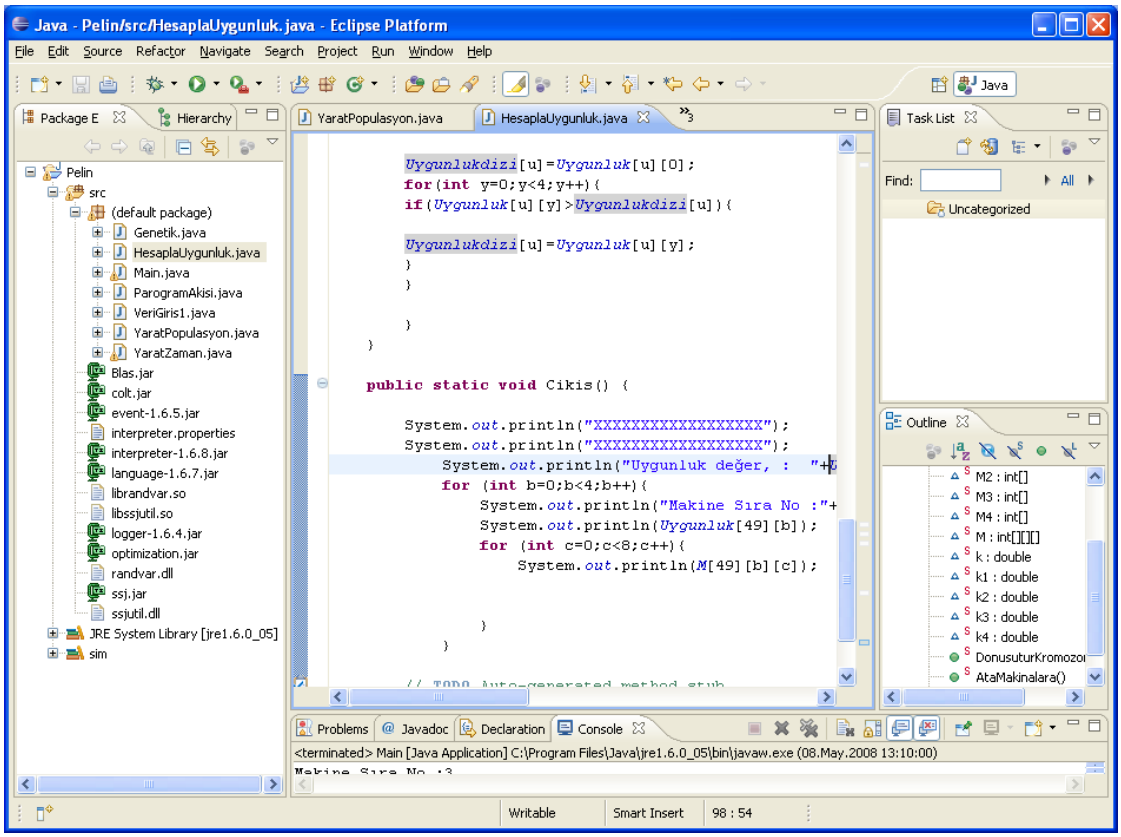
6. iş	**7. iş**	**8. iş**	**9. iş**	**1. Makinedeki İşlerin Toplam Zamanı**
18.77600011	20.98977	25.47013934	**13.30456639**	177.584045
10.04862769	11.21946565	14	**5.960615033**	
3.915396394	6.061577707	3.265259793	**4.604460363**	
5.164093079	3.251657547	5.454699031	**2.284688555**	

Burada dört makineye ayrı ayrı işler atanmaktadır. 1. makinedeki işlerin toplam zamanı 177.584' tür. Bütün işler ilk başta 1. makineye atanır.
177.584-13.304 =164.28 yeni işlem süresidir. 1. makine 9.işdeki 13.304 değeri silinir. Ardından 2. makinede yeni süre 5.969 olur.

164.28-25.470 = 138.81 değeri 1. makinenin işlem yükü olmaktadır. 25.470 değeri silinir.
14+5.960= 19.96 değeri 2. makinede yeni süre olmaktadır.

138.81-20.98 = 117.83 değeri 1. makinede kalan iş yükünü temsil etmektedir. Aynı şekilde 2. makinede iş yükü 11.219 kadar artmıştır.

İşlemler bu şekilde yapılıp bitirildikten sonra son durum incelenmelidir. Değerler silinip aşağıya atanırken alttaki değer daha çoksa artık atama kesilmek zorundadır. İşlem böylelikle sonlandırılmaktadır.



Şekil 7.13 Uygunluk Hesabı kod dizininin gösterimi

Bu noktada yazılan kodlamalar şöyledir;

```
public static void DonusuturKromozonu(){

        for(int m=0;m<50;m++){
        for (int k=0;k<9;k++){
            M[m][0][k]=YaratPopulasyon.Kromozon[m][k];
            M[m][1][k]=0;
            M[m][2][k]=0;
            M[m][3][k]=0;

    public static void AtaMakinalara(){
    for(int u=0;u<50;u++){

        for(int t=0;t<4;t++){
            for (int o=8;o>0;o--){
                for(int z=0;z<4;z++){
```

```
                    for (int p=0;p<8;p++){
                        Uygunluk[u][0]=0;
                        Uygunluk[u][1]=0;
                        Uygunluk[u][2]=0;
                        Uygunluk[u][3]=0;
                        for(int w=0;w<4;w++){
                            for(int q=0;q<8;q++){
                                if(M[u][w][q]>0){

    Uygunluk[u][w]+=YaratZaman.Zamanlar[w][M[u][w][q]-1];

                                    if(M[u][t][o]>0 &&M[u][t][o+1]==0
&&M[u][z][p]<=0){

                                        if((Uygunluk[u][t]-
YaratZaman.Zamanlar[t][M[u][t][o]-
1])>(Uygunluk[u][z]+YaratZaman.Zamanlar[z][M[u][t][o]-1])){

                                            for(int e=0;e<9;e++){
                                                if(M[u][z][e]==0){
                                                M[u][z][e]=M[u][t][o];
                                                M[u][t][o]=0

            Uygunlukdizi[u]=Uygunluk[u][0];
            for(int y=0;y<4;y++){
            if(Uygunluk[u][y]>Uygunlukdizi[u])
            Uygunlukdizi[u]=Uygunluk[u][y];

            }

"public static void AtaMakinalara(){
    for(int u=0;u<50;u++){ "
```

Burada "*public static void*" ile makinelere atanan değerler belirtilmektedir.

Buradaki "*u*" değeri popülasyon sayısını, "*w*" değeri ise makine sayısını vermektedir. Böylelikle yaratılan zamanlar makinelere atanmaktadır.
"YaratZaman.*Zamanlar*[t][*M*[u][t][o]- "

"*o*" makinedeki işlerin sayısını belirtmektedir. [o+1] ile iş sayıları artarda sıralanmaktadır. Uygunluk değerleri "*if*" komutu ile yazılmaktadır. İlk başta yazılan "*for"* komutu ile,

```
"for(int m=0;m<50;m++){
        for (int k=0;k<9;k++){
            M[m][0][k]=YaratPopulasyon.Kromozon[m][k];
            M[m][1][k]=0;
            M[m][2][k]=0;
            M[m][3][k]=0;"
```

ard arda gelen makine sayıları belirtilmektedir.

Uygunluk değeri "*uygunlukdizi*" dizisine aktarılır. Bu uygunluk dizi değeri her bir kromozomun uygunluk değeri olur. Uygunluk değeri makineler içinde maksimum tamamlanma süresine eşit olmaktadır.

7.2.2.5 Genetik İşlemler

"Genetik işlemler" kısmı ile ayrı bir sınıf açılmaktadır. Bu sınıf genetik işlem sınıfı olarak adlandırılmaktadır. İlk popülasyon oluştuğunda kromozomlardaki uygunluk değerine bakılarak, bu popülasyondaki en iyi %10 kromozom korunmaktadır. Böylece değerler diğer nesillere direkt aktarılmaktadır. Geriye kalan %90 lık kesim çaprazlama yapılarak, diğer nesile aktarılmaktadır. Bu işlem "elitizm" olarak adlandırılmaktadır. Bu işlemde en iyi kromozomlar korunarak uygun bireylerin oluşması için çaprazlama ortamı sağlanmaktadır [1].

Seçim yöntemi olarak geliştirilmiş birçok yöntem bulunmakla beraber, rulet çemberi, turnuva ve elitist seçim yöntemleri en yaygın kullanılanlardır. Burada ise kromozomlar seçilirken "turnuva seçim kuralı" uygulanmıştır. "*Turnuva Seçim Yöntemi"* nde yerine koyarak ya da koymadan rastgele *t* adet birey seçilir ve bu büyüklüğe turnuva genişliği adı verilir. Bu

gruptaki en iyi birey, yeni popülasyona kopyalanır. Bu işlem kullanıcı tarafından önceden kararlaştırılan çevrim sayısı kadar tekrarlanır.

Tek nokta çaprazlama, genetik algoritmanın kullandığı en basit çaprazlamadır. Rastgele seçilen kromozom çiftine çaprazlama uygulanır. Tek nokta çaprazlama işlemi için kromozomda çaprazlama yapılacak bölge kullanıcı tarafından rastgele seçilebilir. Oluşan yeni birey ebeveynlerin bazı özelliklerini alarak her ikisinin kopyası olacaktır. Burada ise kromozomlar çaprazlanırken "*iki noktalı çaprazlama*" kullanılmıştır. İki nokta çaprazlamada iki nokta arasında kalan alt dizilerin değiştirilmesiyle iki yeni birey elde edilir. Mesela, iki tane kromozom bulunmaktadır. Bu iki sayı kümesi ana bireyleri oluşturmaktadır:

12**.**34567**.**89 => 1.Birey
23.54167.98 => 2. Birey

Çaprazlamada ilk bireyin ilk noktaya kadarki sayı kümesi alınmalıdır. Yukarıda 1-2 alınmaktadır. Daha sonra ilk bireyin ikinci noktadan sonraki sayı kümesi alınmaktadır. Burada da 8-9 alınmalıdır. Sonra ilk çocuğu oluştururken 1-2-8-9 sayı kümesi başa yazılır. Eksik kalan rakamlar ikinci bireyden sırayla alınır. Örneğin, ikinci bireye bakıldığında 2 bulunmakta ancak 3, 5, 4, 6, 7 nin bulunmadığı görülmektedir.

Sonuç olarak 1-2-8-9-3-5-4-6-7 birinci çocuğu oluşturmaktadır.

Daha sonra ikinci bireye bakılır ve oradaki noktaların değerlendirmesi yapılır. Burada ise 2-3-9-8 alınmaktadır. Bu rakamlar ikinci çocuğa direkt aktarılır. Eksik kalan rakamlar 1. bireyden tamamlanır. Böylece ikinci çocuğumuzda oluşturulmuş olunur.

Bu sıra, 2-3-9-8-1-4-5-6-7 olarak dizilmektedir. Bu dizide ikinci çocuk sırasını oluşturmaktadır.

Burada en önemli nokta, noktaların yerlerinin rassal olarak her bir kromozomda üretilmesidir.

En son olarak "mutasyon" işlemine geçilir. Burada her bir kromozom için 1' den 0' a kadar rassal bir değer üretilmektedir. Bu değer 0.001 dir. Eğer 0.001 den küçük olursa mutasyon

gerçekleşmektedir. Mutasyonda herhangi iki işlem sırası yer değiştirilmektedir. Mutasyon oranı da her bir kromozom için 1/1000 dir.

7.2.2.6 Çıkış Alma

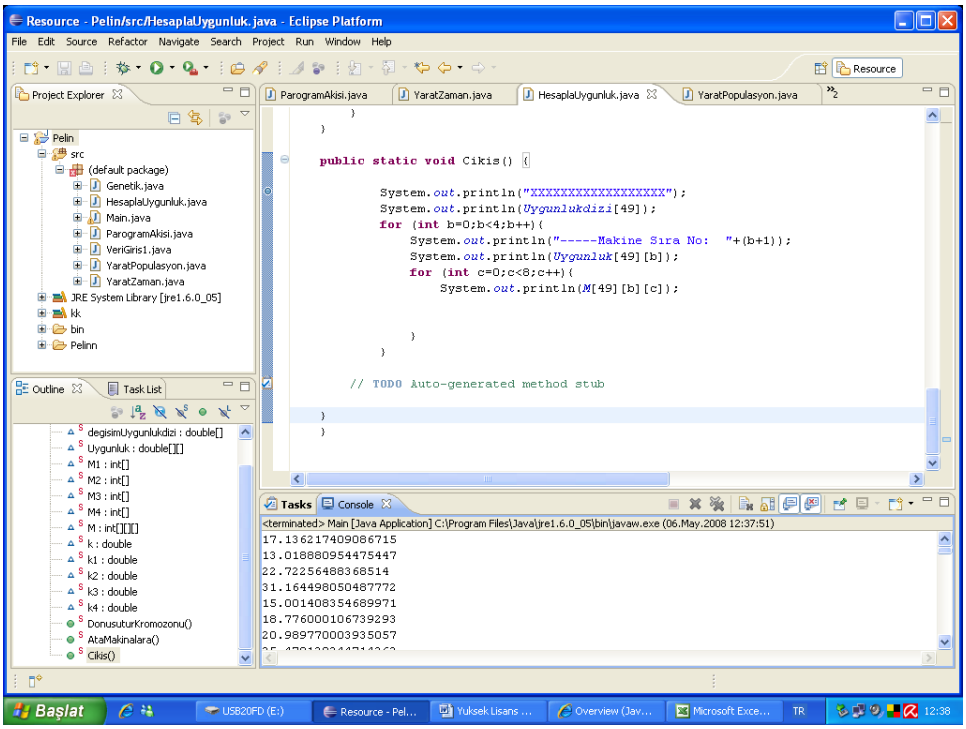
Her bir iterasyondaki en iyi sıra ve uygunluk değeri, uygun komutla konsolda görünmesi sağlanmaktadır. Böylelikle genetik algoritma işlemi tamamlanmaktadır. Uygun çıktılar konsolda gözükmektedir.

```
public static void Cikis() {

        System.out.println("XXXXXXXXXXXXXXXXXXX");
        System.out.println(Uygunlukdizi[49]);
        for (int b=0;b<4;b++){

            System.out.println("-----Makine Sıra No:  "+(b+1));
            System.out.println(Uygunluk[49][b]);

            for (int c=0;c<8;c++){
                System.out.println(M[49][b][c]);
```

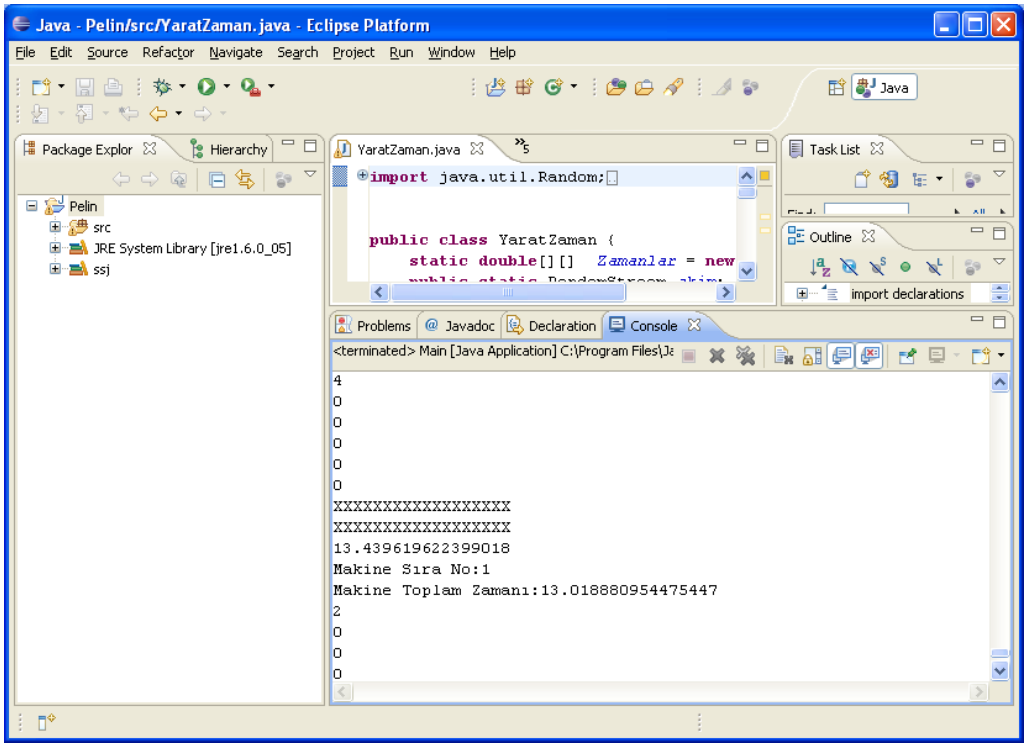
"public static void Cikis()" değeri ile yeni bir değer çıkış değerine atanmaktadır. Böylece sistem çıktıları "*println*" ile gönderilmektedir.

"for (**int** b=0;b<4;b++){" komutu ile makine sayıları ayrı ayrı belirtilmektedir.

"System.*out*.println(*M*[49][b][c]);" buradaki 49 değeri en iyi değer olmaktadır. Popülasyon büyüklüğü 50 alındığı anda, en iyi değer çıkış alma işlemine 49 yazılmaktadır.



Şekil 7.14 Java program ekranı ve yazılan kod düzeni.



Şekil 7.15 Makine toplam zamanlarının bulunuşu ile ilgili ekran görünümü.

Program çalıştırıldıktan sonra, bulunan sonuçlar için iterasyon sayıları şöyledir. Öncelikle 100, sonra 250, 500, 1000, 1500 ve 3000.

Çizelge 7.4 n= 100 ve n= 250 numaralı iterasyon sonuçları

İterasyon No : 250	**iterasyon No : 100**
Uygunluk değer, : 13.439619622399018	Uygunluk değer, : 14.489614898363623
Makine Sıra No :1	Makine Sıra No :1
	Makine Toplam Zamanı:
13.018880952	13.018880954475420
0	0
0	0
0	0
0	0
0	0
0	0
0	0
Makine Sıra No :2	Makine Sıra No :2
13.383112221	11.881927693
9	0
0	0
0	0
0	0
0	0
0	0
0	0
Makine Sıra No :3	Makine Sıra No :3
13.439619626	14.489614965
3	5
8	8
0	1
0	0
0	0
0	0
0	0
Makine Sıra No :4	Makine Sıra No :4
12.724502737	12.487035984
5	7
4	9
0	0
0	0
0	0
0	0
0	0

100 iterasyonda uygunluk değeri 14.489 olarak bulunmuş ve atamalar 1.makineye 2.iş, 2.makineye 3.iş, 3.makineye sırayla 6,5,8,1. işler ve 4.makineye sırayla 4,7,9. işler olarak gerçekleştirilmiştir. 250. iterasyonda 1. makineye 2.iş, 2. makineye 1.iş ve 9.iş, 3. makineye 6. iş, 3. iş, 8. iş, 4. makineye 7.iş, 5.iş, 4.iş atanmaktadır.

Çizelge 7.5 500 ve 1000 iterasyon sonuçları

iterasyon No : 500	**iterasyon No : 1000**
Uygunluk değer, : 13.439619622399018	Uygunluk değer, : 13.439619622399018
Makine Sıra No :1	Makine Sıra No :1
13.018880952	13.018880952
0	0
0	0
0	0
0	0
0	0
0	0
0	0
Makine Sıra No :2	Makine Sıra No :2
13.383112221	13.383112221
9	9
0	0
0	0
0	0
0	0
0	0
0	0
Makine Sıra No :3	Makine Sıra No :3
13.439619626	13.439619626
3	3
8	8
0	0
0	0
0	0
0	0
0	0
Makine Sıra No :4	Makine Sıra No :4
12.724502737	12.724502737
5	5
4	4
0	0
0	0
0	0
0	0
0	0

250 iterasyondan sonra bütün iterasyonlarda uygunluk değerleri aynı bulunmaktadır. Yani en iyi değere 250. iterasyonda ulaşılmıştır. 500. iterasyondan sonraki tüm iterasyonlar tekrar olarak kabul edilebilir. Burada 500. iterasyonda 1. makineye 2. iş, 2. makineye 1.iş ve 9.iş, 3. makineye 6.iş, 3.iş, 8.iş atanırken, 4. makineye 7. iş, 5. iş ve 4. iş atanmaktadır. Aynı şekilde 1000. iterasyonda ise 1. makineye 2.iş, 2. makineye 1. iş, 9.iş, 3. makineye 6.iş, 3.iş, 8. iş, 4. makineye 7.iş, 5.iş ve 4.iş atanmaktadır. Burada görülmektedir ki, 500. ve 1000.

iterasyonlarda her bir makineye atanan iş sayıları birbirlerinin aynısı olmaktadır. Bu şekilde atamalar tamamlanmaktadır.

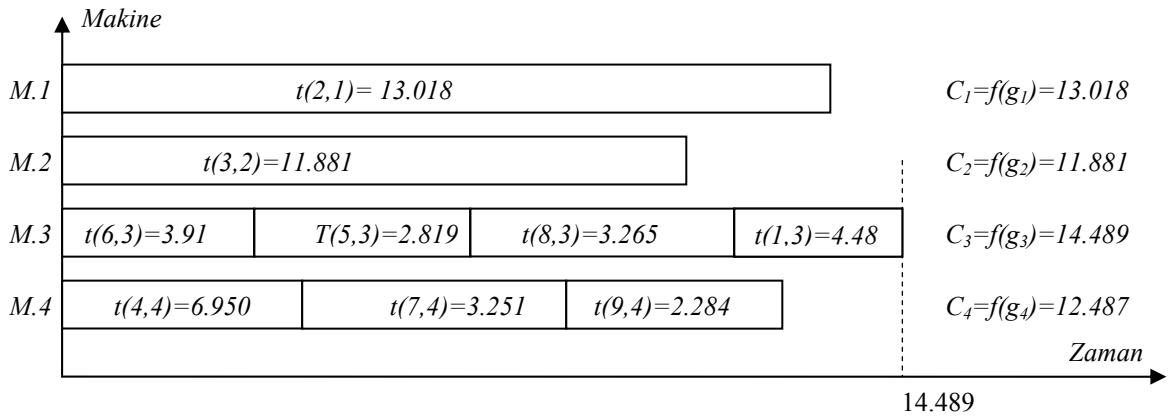
Çizelge 7.6 1500 ve 3000 iterasyon sonuçları

iterasyon No : 1500	**iterasyon No : 3000**
Uygunluk değer, : 13.439619622399018	Uygunluk değer, : 13.439619622399018
Makine Sıra No :1	Makine Sıra No :1
13.018880952	13.018880952
0	0
0	0
0	0
0	0
0	0
0	0
0	0
Makine Sıra No :2	Makine Sıra No :2
13.383112221	13.383112229
9	1
0	0
0	0
0	0
0	0
0	0
0	0
Makine Sıra No :3	Makine Sıra No :3
13.439619623	13.439619623
6	6
8	8
0	0
0	0
0	0
0	0
0	0
Makine Sıra No :4	Makine Sıra No :4
12.724502737	12.724502737
5	5
4	4
0	0
0	0
0	0
0	0
0	0

1500. iterasyonda ise makinelere atanmalar şöyle olmaktadır. En uygun değer 13.439 olmaktadır. Ve bu değer 3. makineye atanan işlerle olmaktadır. Buna göre, 1. makineye 2.iş, 2. makineye 1. iş, 9.iş, 3. makineye 3. iş, 6. iş, 8. iş, 4. makineye 7. iş, 5. iş, 4. iş atanmaktadır.

3000. iterasyonda yeniden en uygun değer 13.439 olmaktadır. Buna göre 1. makineye 2. iş, 2. makineye 9. iş, 1. iş, 3. makineye 3. iş, 6. iş, 8. iş, 4. makineye 7. iş, 5. iş, 4. iş atanmaktadır. Buradan 1500 ve 3000. iterasyonlara baktığımızda 2 makineye atanan işlerin 1. ve 9. işler oldukları görülmektedir. 1500. iterasyonda 2. makineye atanan işlerin sırasıyla 1 ve 9. işler olduğu görülürken, 3000. iterasyonda 2. makineye atanan işlerin sırayla 9. ve 1. işler olduğu görülmektedir.

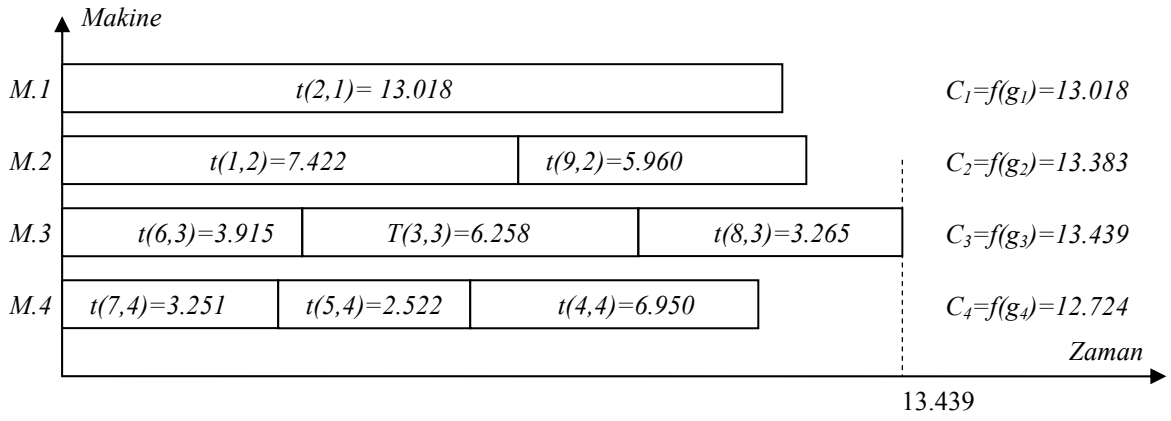
İterasyon numarası **n=100** için Gantt şemasını oluşturursak şu şekli elde etmekteyiz. Burada yukarıdaki grafiklerden de görüleceği üzere 1.makineye 2. iş, 2. makineye 3. iş, 3. makineye 6. iş, 5.iş, 8.iş ve 1. iş atanmakta iken 4. makineye sırayla 4.iş, 7.iş ve 9. iş atanmaktadır.



Şekil 7.16 Genetik algoritma kullanılarak iş sıralamada n=100 için oluşturulan Gantt şeması

Çıkan sonuç içinde en uygun değer 14.489 değeri ile 3. makineye atanan toplam iş proses değerlerinden elde edilmektedir.

Diğer yandan n= 500 iterasyon için yapılan genetik algoritma sonucunda, 1. makineye 2.iş, 2. makineye 1.iş ve 9.iş, 3.makineye 6.iş, 3.iş ve 8.iş atanmakta iken, 4.makineye sırayla 7.iş, 5.iş ve 4.iş atanmaktadır. Yapılan Gantt şeması ve en uygun proses zamanları ise aşağıdaki şekilde gösterilmektedir.



Şekil 7.17 Genetik algoritma kullanılarak iş sıralamada n=500 için oluşturulan Gantt şeması

Yapılan uygulamada en uygun değere yani en uygun proses zamanına 250. iterasyonda ulaşılmaktadır.

7.2.3 On Makine Otuz İş İçin Genetik Algoritma Uygulaması

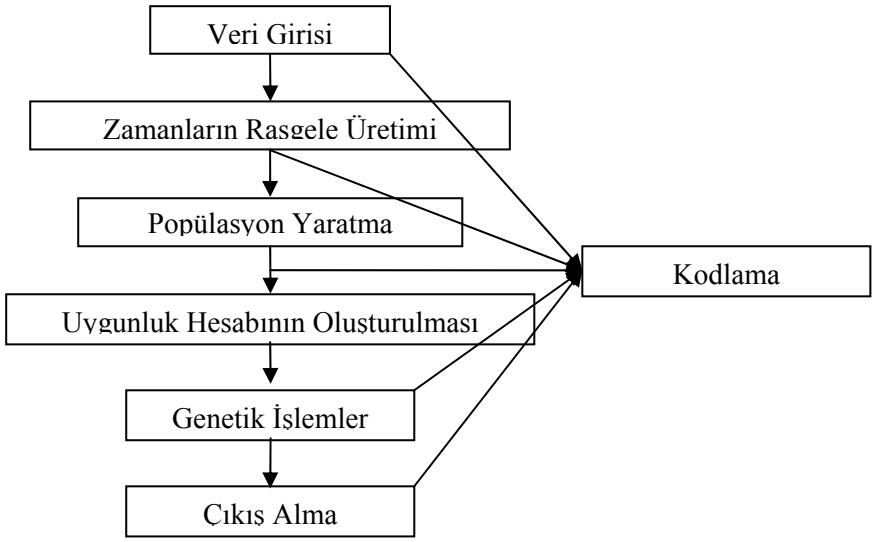
7.2.3.1 Modelin Tanımı

Burada bir diğer senaryo üretilmektedir. Buna göre 4 makine 9 iş için oluşturulan programın çalışmasının ardından şimdi de 10 makine 30 iş için gerekli veriler programa girilmiş ve program çalıştırılmıştır. Veriler ilk olarak excelde yazılmış bulanık değerlere dökülmüş ve ardından programa yazılmıştır.

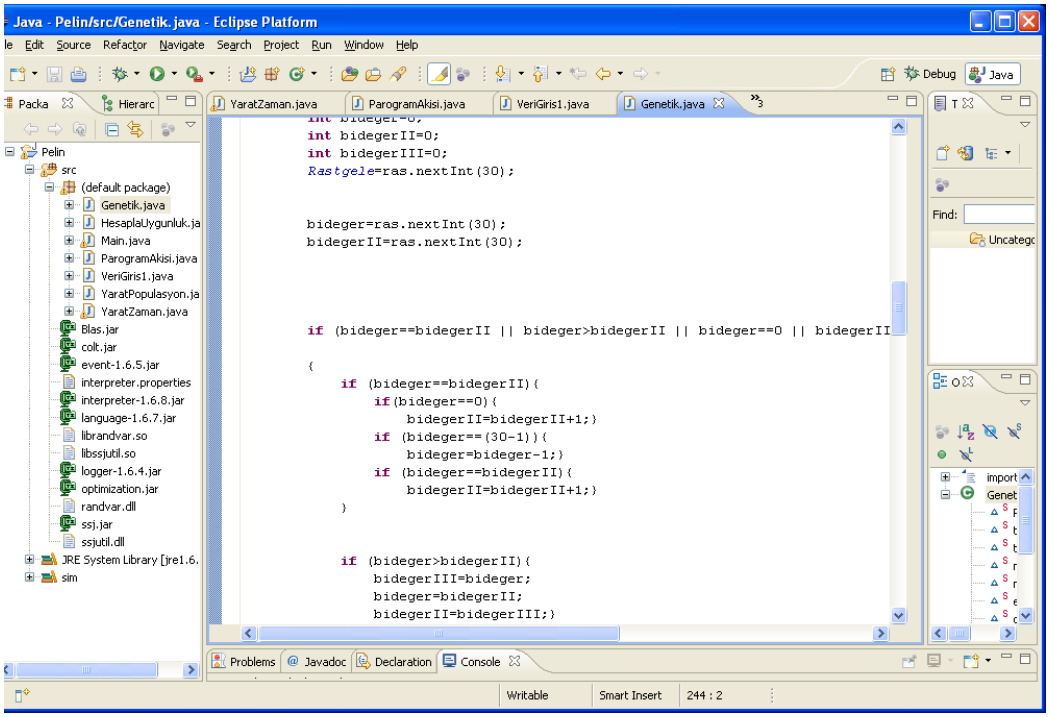
Benzer olmayan paralel makinelerde yapılan ikinci uygulamada, makine ve iş sayıları arttırılmıştır. Buna göre yapılan uygulamada veriler arttırılmış ve programa girilen kodlamalar daha detaylı bir hal almıştır.

Metodoloji şu şekildedir;

1. Veri girişi.
2. Zamanların rasgele üretilmesi.
3. Popülasyon yaratma.
4. Uygunluk hesabının oluşturulması.
5. Genetik işlemler.
6. Çıkış alma.



Şekil. 7.18 İkinci uygulamanın metodolojisi



Şekil 7.19 Eclipse de oluşturulan genetikJava ekranı

Bu veriler ise şöyledir:

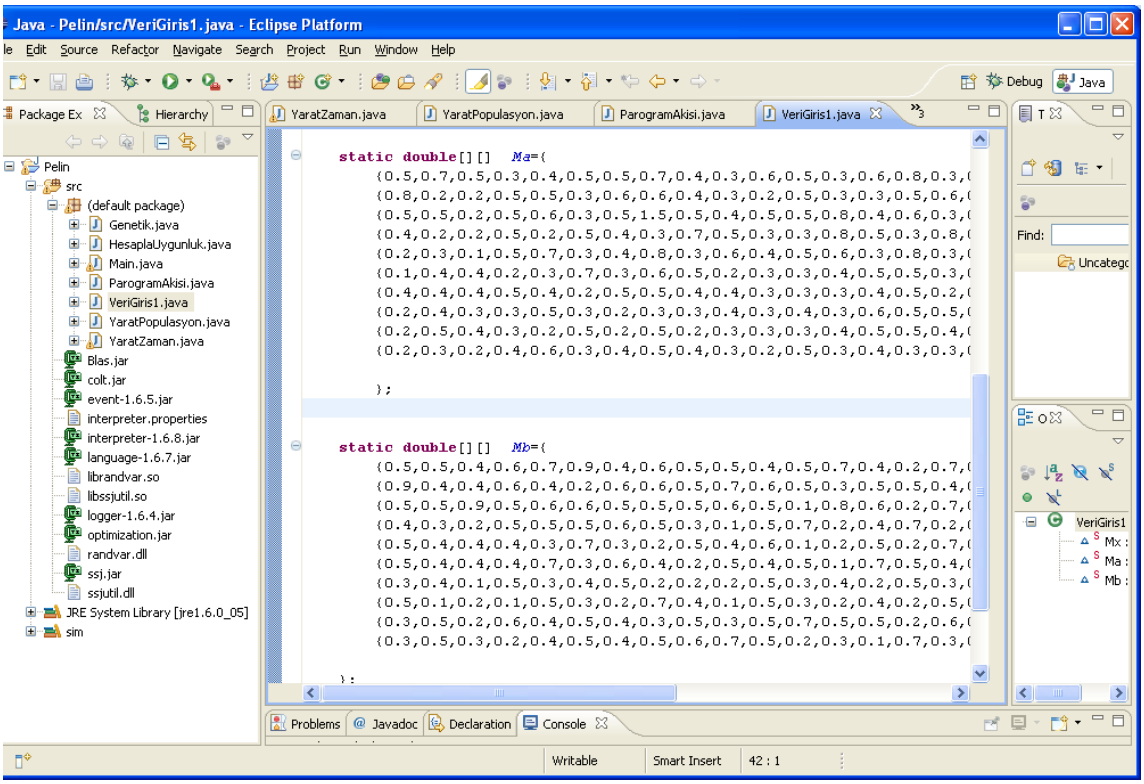
7.2.3.2 Programa Girilen Veriler

Programa ilk örnekte olduğu gibi yeniden veriler girilmektedir. Ancak bu sefer programa girilecek veriler oldukça fazla olmaktadır. Buna göre uygun veriler şu şekilde kod düzenine dökülmektedir:

```
static double[][] Mx={
        {17,13,23,31,15,19,21,25,13,18.3,23.6,13.5,25.3,5.6,17.8,20.3,8.5,13.6,11.2,3.2,13.5,1
8.4,21.2,17.6,15.6,14.7,10.4,4.9,17.8,22.3},

        {7,8,12,15,9,10,11,14,6,3.3,18.2,11.5,10.7,14.3,5.5,4.6,8.7,11.8,24.3,21,3.4,18.3,11.6,
4.3,22,16.7,14.7,13.6,24.6,18.5},

        {4.5,5.5,6,7.5,3,4,6.5,3.5,4.5,8.4,3.5,9.9,20,6.4,11.8,12.3,4.8,5.7,22.2,11.3,8.5,3.6,9.3,
12.7,25.3,15.3,22.3,16.7,3.2,17.4},

        {4.6,3.7,5.8,7.0,2.2,5.0,3.4,5.3,2.7,5.9,7.3,20.3,2.8,3.5,10.3,11.8,25,8.3,7.4,23.4,6.9,3.
6,10.4,6.7,11.7,4.7,13.6,4.9,6.6,16.3},

        {18.2,13.6,20.1,16.5,10.7,24.3,17.4,13.8,20.3,17.6,13.4,11.9,16.8,12.3,7.8,12.3,20.3,1
4.7,15.8,8.9,4.5,17.4,6.2,7.9,23.5,6.6,5.9,2.3,13.4,12.5},

        {8.1,9.4,17.6,11.2,9.3,13.7,6.3,9.6,19.8,10.2,14.3,22.3,8.9,20,16.5,17.3,6.6,5.8,4.7,16.
2,24.3,20.4,15.8,3.5,11.4,16.8,15.5,8.8,18.9,20.3},

        {6.7,5.6,3.9,9.5,8.7,4.2,5.5,7.8,3.8,7.8,5.3,4.7,2.5,3.8,14.5,11.2,4.8,6.6,11.3,18.5,4.3,1
8.9,14.2,4.8,3.4,13.5,17.7,9.6,4.2,18.5},

        {4.8,2.9,5.8,2.3,4.5,6.7,3.8,11.3,20.3,3.9,2.5,14.7,22.8,5.6,6.8,13.5,12.4,20.3,4.1,2.4,5.
4,7.6,8.9,16.8,9.8,14.8,8.7,18.8,5.8,6.5},

        {17.2,10.5,23.8,17.3,13.4,23.5,19.2,10.7,23.5,14.7,22.5,16.3,11.4,12.5,15.8,13.4,18.2,
11.8,13.2,15.9,10.2,4.6,20.2,17.7,5.7,17.9,11.5,22.6,14,7.8},
```

```
        {3.2,8.3,12.2,3.8,5.6,12.3,6.6,8.5,12.4,23.3,18.2,16.8,10.3,7.9,4.3,8.7,3.9,15.4,16.6,13.
4,12.4,11.8,23.3,15.6,12.5,20.2,12.8,5.5,13.6,17.3},
        };
static double[][] Ma={
        {0.5,0.7,0.5,0.3,0.4,0.5,0.5,0.7,0.4,0.3,0.6,0.5,0.3,0.6,0.8,0.3,0.5,0.6,0.2,0.2,0.5,0.4,0.2
,0.4,0.6,0.4,0.4,0.6,0.5,0.3},

        {0.8,0.2,0.2,0.5,0.5,0.3,0.6,0.6,0.4,0.3,0.2,0.5,0.3,0.3,0.5,0.6,0.4,0.6,0.3,0.5,0.4,0.3,0.6
,0.3,0.6,0.4,0.5,0.4,0.4,0.4},

        {0.5,0.5,0.2,0.5,0.6,0.3,0.5,1.5,0.5,0.4,0.5,0.5,0.8,0.4,0.6,0.3,0.3,0.3,0.2,0.3,0.5,0.3,0.3
,0.2,0.3,0.3,0.3,0.4,0.2,0.4},

        {0.4,0.2,0.2,0.5,0.2,0.5,0.4,0.3,0.7,0.5,0.3,0.3,0.8,0.5,0.3,0.8,0.5,0.3,0.4,0.4,0.7,0.6,0.4
,0.4,0.2,0.4,0.3,0.5,0.3,0.3},

        {0.2,0.3,0.1,0.5,0.7,0.3,0.4,0.8,0.3,0.6,0.4,0.5,0.6,0.3,0.8,0.3,0.3,0.7,0.3,0.5,0.5,0.4,0.2
,0.5,0.5,0.6,0.3,0.3,0.4,0.5},

        {0.1,0.4,0.4,0.2,0.3,0.7,0.3,0.6,0.5,0.2,0.3,0.3,0.4,0.5,0.5,0.3,0.6,0.8,0.7,0.2,0.3,0.4,0.3
,0.5,0.4,0.6,0.5,0.5,0.4,0.3},

        {0.4,0.4,0.4,0.5,0.4,0.2,0.5,0.5,0.4,0.4,0.3,0.3,0.3,0.4,0.5,0.2,0.8,0.6,0.3,0.5,0.3,0.5,0.2
,0.5,0.4,0.5,0.3,0.3,0.2,0.5},

        {0.2,0.4,0.3,0.3,0.5,0.3,0.2,0.3,0.3,0.4,0.3,0.4,0.3,0.6,0.5,0.5,0.4,0.3,0.1,0.4,0.4,0.4,0.5
,0.4,0.4,0.4,0.3,0.4,0.2,0.3},

        {0.2,0.5,0.4,0.3,0.2,0.5,0.2,0.5,0.2,0.3,0.3,0.3,0.4,0.5,0.5,0.4,0.2,0.4,0.2,0.5,0.2,0.4,0.2
,0.5,0.6,0.5,0.5,0.6,0.5,0.3},

        {0.2,0.3,0.2,0.4,0.6,0.3,0.4,0.5,0.4,0.3,0.2,0.5,0.3,0.4,0.3,0.3,0.9,0.4,0.4,0.4,0.4,0.8,0.2
,0.4,0.5,0.2,0.2,0.5,0.4,0.3},
```

```
        };
static double[][] Mb={
        {0.5,0.5,0.4,0.6,0.7,0.9,0.4,0.6,0.5,0.5,0.4,0.5,0.7,0.4,0.2,0.7,0.5,0.4,0.7,0.6,0.4,0.6,0.6
,0.4,0.4,0.3,0.6,0.1,0.2,0.7},

        {0.9,0.4,0.4,0.6,0.4,0.2,0.6,0.6,0.5,0.7,0.6,0.5,0.3,0.5,0.5,0.4,0.3,0.2,0.7,0.8,0.5,0.5,0.4
,0.4,0.6,0.3,0.3,0.4,0.3,0.3},

        {0.5,0.5,0.9,0.5,0.6,0.6,0.5,0.5,0.5,0.6,0.5,0.1,0.8,0.6,0.2,0.7,0.2,0.3,0.8,0.5,0.5,0.4,0.7
,0.3,0.4,0.7,0.5,0.3,0.4,0.4},

        {0.4,0.3,0.2,0.5,0.5,0.5,0.6,0.5,0.3,0.1,0.5,0.7,0.2,0.4,0.7,0.2,0.5,0.4,0.6,0.6,0.1,0.4,0.6
,0.3,0.3,0.3,0.4,0.1,0.4,0.3},

        {0.5,0.4,0.4,0.4,0.3,0.7,0.3,0.2,0.5,0.4,0.6,0.1,0.2,0.5,0.2,0.7,0.7,0.3,0.2,0.1,0.5,0.6,0.7
,0.1,0.5,0.4,0.1,0.3,0.4,0.5},

        {0.5,0.4,0.4,0.4,0.7,0.3,0.6,0.4,0.2,0.5,0.4,0.5,0.1,0.7,0.5,0.4,0.4,0.2,0.3,0.4,0.7,0.6,0.2
,0.5,0.3,0.2,0.5,0.1,0.1,0.5},

        {0.3,0.4,0.1,0.5,0.3,0.4,0.5,0.2,0.2,0.2,0.5,0.3,0.4,0.2,0.5,0.3,0.2,0.4,0.5,0.5,0.5,0.1,0.6
,0.2,0.5,0.5,0.3,0.4,0.3,0.5},

        {0.5,0.1,0.2,0.1,0.5,0.3,0.2,0.7,0.4,0.1,0.5,0.3,0.2,0.4,0.2,0.5,0.6,0.7,0.4,0.6,0.4,0.4,0.1
,0.2,0.2,0.2,0.3,0.2,0.2,0.5},

        {0.3,0.5,0.2,0.6,0.4,0.5,0.4,0.3,0.5,0.3,0.5,0.7,0.5,0.5,0.2,0.6,0.4,0.2,0.5,0.1,0.3,0.4,0.3
,0.3,0.3,0.1,0.5,0.4,0.5,0.2},

        {0.3,0.5,0.3,0.2,0.4,0.5,0.4,0.5,0.6,0.7,0.5,0.2,0.3,0.1,0.7,0.3,0.1,0.5,0.4,0.4,0.5,0.2,0.3
,0.4,0.5,0.5,0.2,0.5,0.4,0.5},
        };
```



Şekil 7.20 10 makine 30 iş problemi için oluşturulan veri giriş ekranı

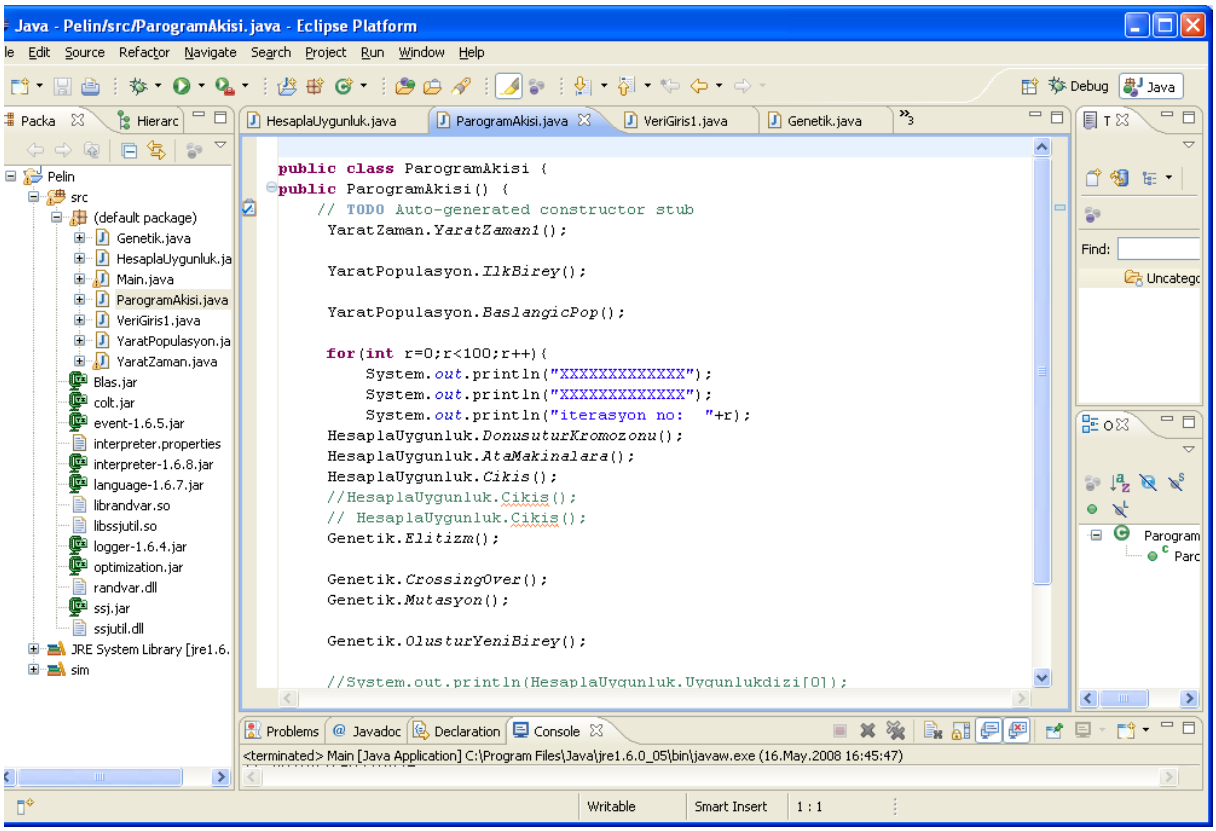
7.2.3.3 Zamanların Rasgele Üretilmesi

Program Akışı içinde uygunluk değerlerinden önce public class diyerek yeni bir değere zamanlar atanacaktır. Kullanılan kod düzeni şöyledir;

Yarat Zaman. Yarat Zamanı ();

Yarat Populasyon. Ilk Birey ();

Yarat Populasyon. Baslangic Pop ():



Şekil 7.21 10 makine 30 iş probleminde oluşturulmuş genel kod düzeni

7.2.3.4 Popülasyon Yaratma

10 makine 30 iş problemi için popülasyon yaratma işlemine gidilmektedir. Makine sırası "*ArrayList (liste)*" kullanılarak, genetik algoritmada liste türüne döndürülmüştür. Liste türüne döndürülmesi rasgele sıra türetme imkanı sağlamaktadır. Bu işlem "*shuffle*" koduyla tekrarlama işlemini sağlayarak, popülasyonların daha net bir şekilde ve birbirinden ayrı bir şekilde yaratılmasını sağlamaktadır.

İlk bireyler böylece rasgele sıralanmış olarak, kromozomlara dağıtılmaktadır. Daha sonra bu kromozomlar dizi türüne çevrilerek işlem yapılmasına olanak sağlanmaktadır. Popülasyon büyüklüğü 100 alınmaktadır.

7.2.3.5 Uygunluk Hesabının Oluşturulması

Öncelikle kromozom hattından tüm sıralar 1. makineye atanır. Daha sonra atanan sıradaki son işlem 2. makineye atanır. Eğer 1. makinanın toplam işlem süresinden çıkartılan işlemin süresi 2. makinedeki toplam iş süresi ve yeni atanan işlemin iş süresinin toplamından büyükse,

atama gerçekleşmektedir. Böylece son iterasyona kadar bu mantıkla makineler arasında atanamayacak iş kalıncaya kadar devam edilmek zorunda kalınacaktır.

7.2.3.6 Genetik İşlemler

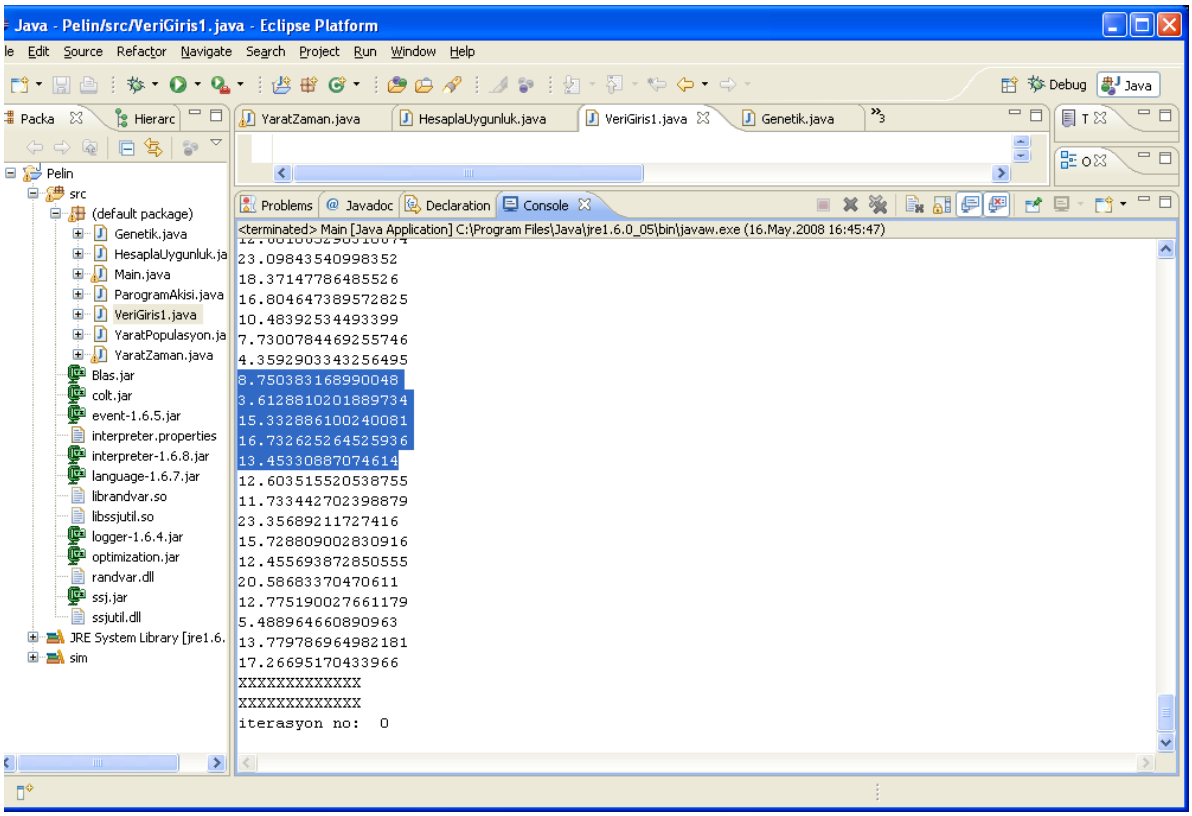
İlk popülasyon oluştuğunda kromozomlardaki uygunluk değerine bakılarak, bu popülasyondaki en iyi %10 kromozom korunmaktadır. Bu oran diğer nesillere direkt aktarılmaktadır. Geriye kalan %90 lık kesim çaprazlama yapılarak, diğer nesile aktarılır. Bu işlem "elitizm" olarak adlandırılmaktadır. Bu şekilde en iyi kromozomlar korunarak uygun bireylerin oluşması için çaprazlama ortamı sağlanmaktadır.

Uygulamanın her ikisinde de kromozomlar seçilirken "*turnuva seçim kuralı*" uygulanmıştır. Turnuva Seçim Yönteminde yerine koyarak ya da koymadan rastgele *t* adet birey seçilir ve bu büyüklüğe turnuva genişliği adı verilir. Bu gruptaki en iyi birey, yeni popülasyona kopyalanır. Bu işlem kullanıcı tarafından önceden kararlaştırılan çevrim sayısı kadar tekrarlanır. Kromozomlar çaprazlanırken "*iki noktalı çaprazlama metodu*" kullanılmıştır. İki nokta çaprazlamada iki nokta arasında kalan alt dizilerin değiştirilmesiyle iki yeni birey elde edilir.

7.2.3.7 Çıkış Alma

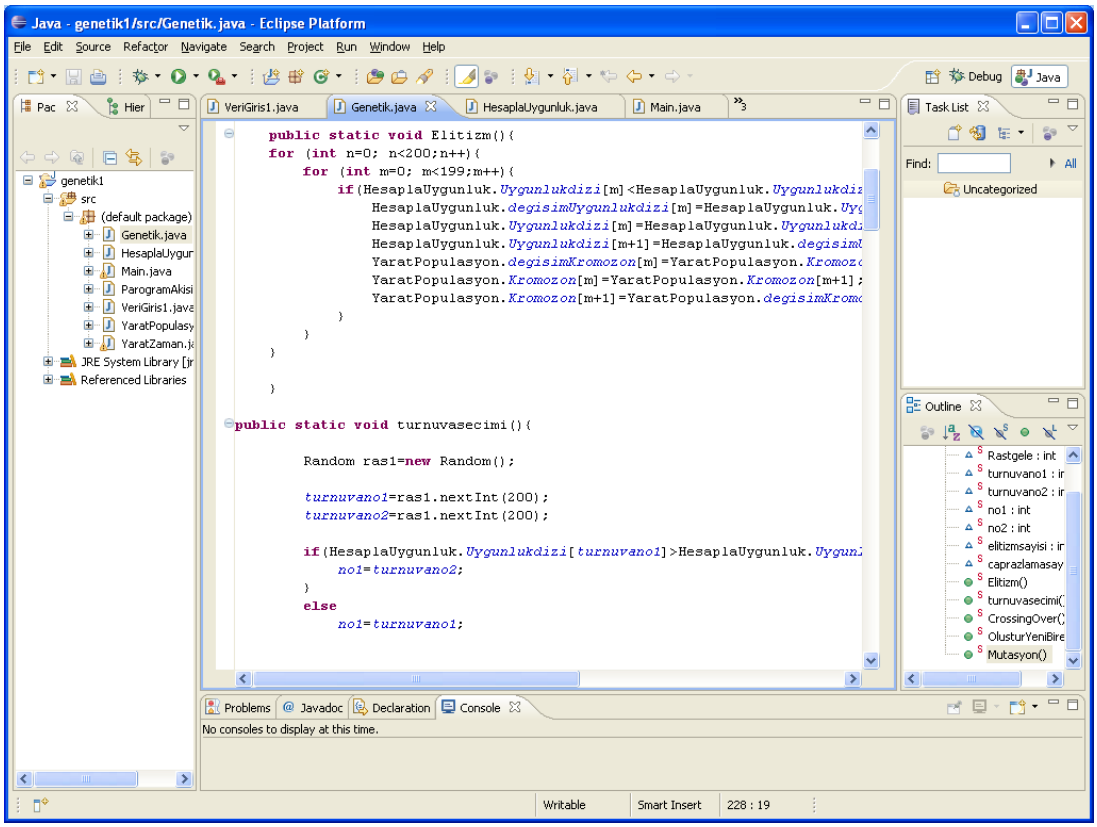
Bu işlemle her bir iterasyondaki en iyi sıra ve uygunluk değerinin, uygun komutla konsolda görünmesi sağlanmaktadır. Böylelikle genetik algoritma işlemi tamamlanmaktadır. Sonuç olarak uygun çıktılar konsolda gözükmektedir. Böylelikle genetik algoritma sonlandırılmaktadır.

Veriler girilip, uygun kodlamalar yazıldıktan sonra, program çalıştırıldığında ilk iterasyonda ortaya çıkan ekran resmi ise şöyledir;



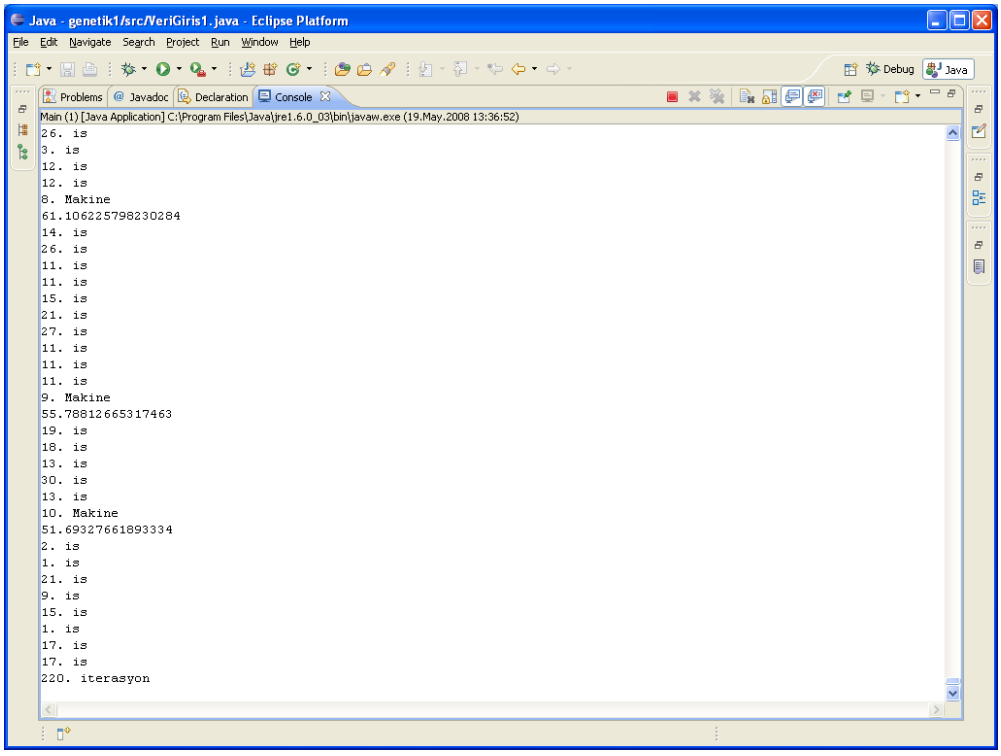
Şekil 7.22 Programın çalışmasıyla çıkan ekran görüntüsü

Sonuçta, 10 makine 30 iş sıralama problemi için program çalıştırılmış ve 1000 iterasyon yaptırılmıştır. Buna göre program belirli bir iterasyondan sonra tekrarlanmakta yani uygunluk değeri sabitlenmekte ve en uygun değere ulaşmış olmaktadır. Yapılan ikinci uygulamada sırayla 100, 250, 500, 1000. iterasyonlar denenmiş ve sonuçlar karşılaştırılmıştır. Buna göre 500. iterasyonda en uygun değere ulaşıldığı görülmektedir.




```
    public static void Elitizm(){
    for (int n=0; n<200;n++){
        for (int m=0; m<199;m++){
            if(HesaplaUygunluk.Uygunlukdizi[m]<HesaplaUygunluk.Uygunlukdizi
                HesaplaUygunluk.degisimUygunlukdizi[m]=HesaplaUygunluk.Uyg
                HesaplaUygunluk.Uygunlukdizi[m]=HesaplaUygunluk.Uygunlukdi
                HesaplaUygunluk.Uygunlukdizi[m+1]=HesaplaUygunluk.degisimU
                YaratPopulasyon.degisimKromozon[m]=YaratPopulasyon.Kromozo
                YaratPopulasyon.Kromozon[m]=YaratPopulasyon.Kromozon[m+1];
                YaratPopulasyon.Kromozon[m+1]=YaratPopulasyon.degisimKromo
            }
        }
    }

    }

public static void turnuvasecimi(){

        Random ras1=new Random();

        turnuvano1=ras1.nextInt(200);
        turnuvano2=ras1.nextInt(200);

        if(HesaplaUygunluk.Uygunlukdizi[turnuvano1]>HesaplaUygunluk.Uygunl
            no1=turnuvano2;
        }
        else
            no1=turnuvano1;
```

Şekil 7.23 Programın genetik kod bölümünün görüntüsü



Şekil 7.24 Program çalışırken ortaya çıkan ekran görüntüsü

100. iterasyonda makinelere atanan iş sayıları şöyle gösterilmektedir;

1. makineye 22.iş, 14.iş, 17.iş, 12.iş, 29.iş, 27.iş, 20.iş atanmaktadır.

2. makineye 16. iş, 24.iş, 21.iş, 10.iş, 13.iş, 28.iş, 11.iş ve 15.iş atanmaktadır.

3.makineye 19.iş, 18.iş, 5.iş, 1.iş, 30.iş ve 7. iş atanmaktadır.

4.makineye 9.iş, 6.iş, 2.iş, 26.iş, 25.iş, 4.iş, 8.iş, 23.iş ve 3.iş atanmaktadır.

5.makineye 17.iş, 15.iş, 12.iş, 23.iş, 8.iş atanmaktadır.

6.makineye 4.iş, 20.iş, 7.iş, 24.iş, 10.iş, 17.iş atanmaktadır.

7.makineye 8.iş, 24.iş, 21.iş, 25.iş, 13.iş, 26.iş, 3.iş, 12.iş atanmaktadır.

8.makineye 14.iş, 26.iş, 11.iş, 11.iş, 15.iş, 21.iş, 27.iş, 11.iş, 11.iş, 11.iş atanmaktadır.

9.makineye 19.iş, 18.iş,13.iş, 30.iş, 13.iş atanmaktadır.

10.makineye 2.iş, 1.iş, 21.iş, 9.iş, 15.iş, 1.iş, 17.iş, 17.iş atanmaktadır.

500. iterasyonda makinelere atanan iş sayıları şöyle gösterilmektedir;

1. makineye 22.iş, 19. iş, 14. iş, 17. iş, 29. iş, 27. iş, 20. iş atanmaktadır.

2. makineye 4. iş, 25. iş, 11. iş, 15. iş, 8. iş atanmaktadır.

3. makineye 18. iş, 1. iş, 30. iş, 23. iş, 5. iş, 7. iş, 3. iş, 12. iş atanmaktadır.

4. makineye 9. iş, 6. iş, 2. iş, 26. iş, 28. iş, 13. iş, 10. iş, 21. iş, 16. iş, 24. iş atanmaktadır.

5. makineye 17. iş, 15. iş, 12. iş, 23. iş, 8. iş atanmaktadır.

6. makineye 4. iş, 20. iş, 7. iş, 24. iş, 10. iş, 17. iş atanmaktadır.

7. makineye 8. iş, 24. iş, 21. iş, 25. iş, 13. iş, 26. iş, 3. iş, 12. iş, 12. iş atanmaktadır.

8. makineye 14. iş, 26. iş, 11. iş, 11. iş, 15. iş, 21. iş, 27. iş, 11. iş, 11. iş, 11. iş atanmaktadır.

9. makineye 19. iş, 18. iş, 13. iş, 30. iş, 13. iş atanmaktadır.

10. makineye 2. iş, 1. iş, 21. iş, 9. iş, 15. iş, 1. iş, 17. iş, 17. iş atanmaktadır.

Çizelge 7.7 10 makine 30 iş sıralama probleminin program çalıştırılınca 100. ve 250. iterasyon sonuçları

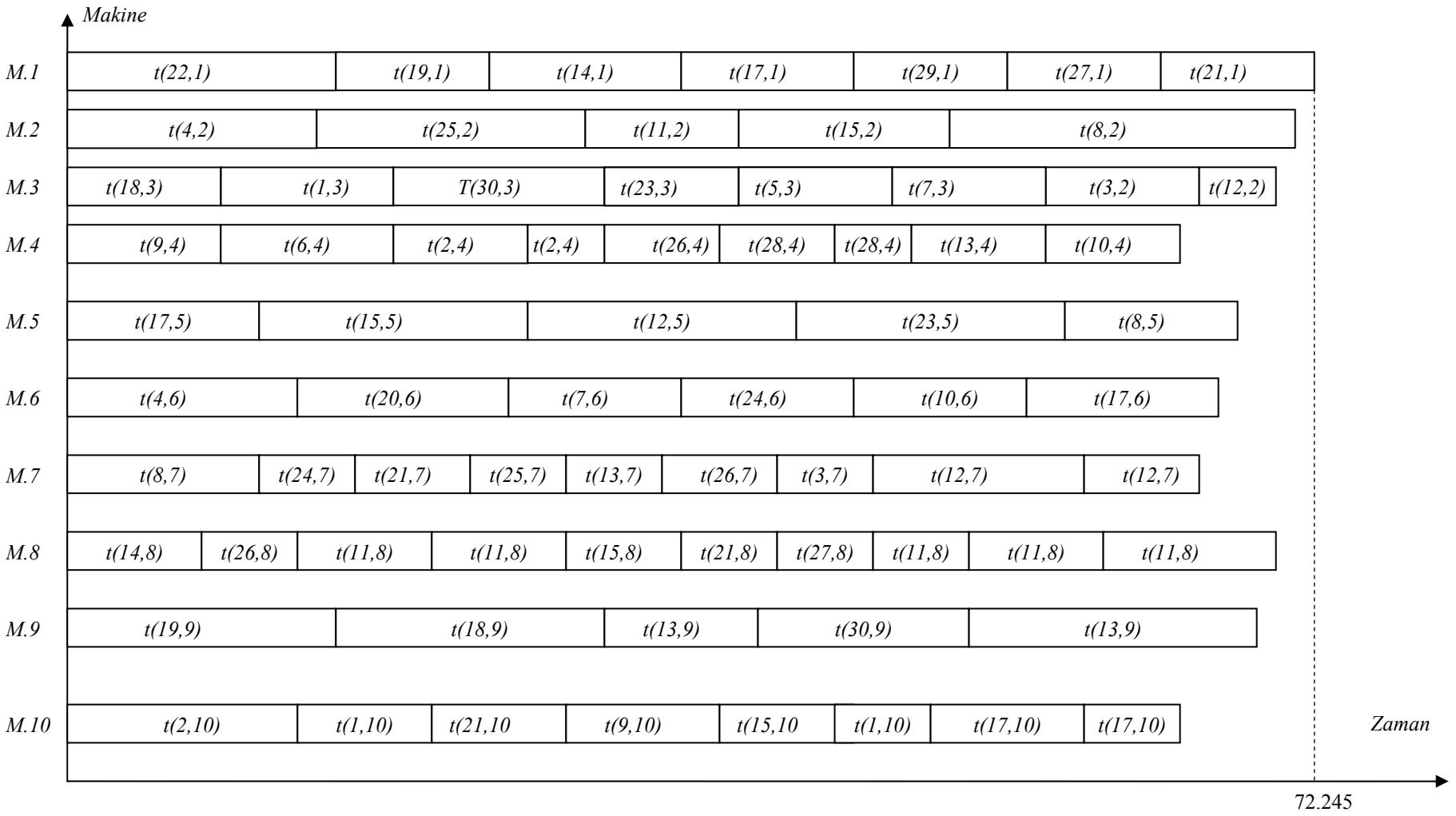
100. İterasyon									
En Uygun Değer: 77.3488684953164									
1. Makine	2. Makine	3. Makine	4. Makine	5. Makine	6. Makine	7. Makine	8. Makine	9. Makine	10. Makine
77.3488685	63.51120296	59.7791312	55.86806121	59.83681677	54.09369135	49.97718764	61.1062258	55.78812665	51.693277
22. iş	16. iş	19. iş	9. iş	17. iş	4. iş	8. iş	14. iş	19. iş	2. iş
14. iş	24. iş	18. iş	6. iş	15. iş	20. iş	24. iş	26. iş	18. iş	1. iş
17. iş	21. iş	5. iş	2. iş	12. iş	7. iş	21. iş	11. iş	13. iş	21. iş
12. iş	10. iş	1. iş	26. iş	23. iş	24. iş	25. iş	11. iş	30. iş	9. iş
29. iş	13. iş	30. iş	25. iş	8. iş	10. iş	13. iş	15. iş	13. iş	15. iş
27. iş	28. iş	7. iş	4. iş		17. iş	26. iş	21. iş		1. iş
20. iş	11. iş		8. iş			3. iş	27. iş		17. iş
	15. iş		23. iş			12. iş	11. iş		17. iş
			3. iş			12. iş	11. iş		
							11. iş		

250. İterasyon									
En Uygun Değer: 75.5086928345087									
1. Makine	2. Makine	3. Makine	4. Makine	5. Makine	6. Makine	7. Makine	8. Makine	9. Makine	10. Makine
75.50869283	74.43596889	62.33108055	54.31801431	59.83681677	54.09369135	49.97718764	61.1062258	55.78812665	51.693277
22. iş	4. iş	18. iş	9. iş	17. iş	4. iş	8. iş	14. iş	19. iş	2. iş
19. iş	25. iş	1. iş	6. iş	15. iş	20. iş	24. iş	26. iş	18. iş	1. iş
14. iş	11. iş	30. iş	2. iş	12. iş	7. iş	21. iş	11. iş	13. iş	21. iş
17. iş	15. iş	23. iş	26. iş	23. iş	24. iş	25. iş	11. iş	30. iş	9. iş
29. iş	8. iş	5. iş	28. iş	8. iş	10. iş	13. iş	15. iş	13. iş	15. iş
27. iş		7. iş	13. iş		17. iş	26. iş	21. iş		1. iş
20. iş		3. iş	10. iş			3. iş	27. iş		17. iş
		12. iş	21. iş			12. iş	11. iş		17. i
			16. iş			12. iş	11. iş		
			24. iş				11. iş		

Çizelge 7.8 10 makine 30 iş sıralama probleminin program çalıştırılınca 500. ve 1000. iterasyon sonuçları

500. İterasyon									
En Uygun değer: 72.2457617803043									
1. Makine	2. Makine	3. Makine	4. Makine	5. Makine	6. Makine	7. Makine	8. Makine	9. Makine	10. Makine
72.2457617803043	74.43596889	62.33108055	54.31801431	59.83681677	54.09369135	49.97718764	61.1062258	55.78812665	51.693277
22. is	4. is	18. is	9. is	17. is	4. is	8. is	14. is	19. is	2. is
19. is	25. is	1. is	6. is	15. is	20. is	24. is	26. is	18. is	1. is
14. is	11. is	30. is	2. is	12. is	7. is	21. is	11. is	13. is	21. is
17. is	15. is	23. is	26. is	23. is	24. is	25. is	11. is	30. is	9. is
29. is	8. is	5. is	28. is	8. is	10. is	13. is	15. is	13. is	15. is
27. is		7. is	13. is		17. is	26. is	21. is		1. is
20. is		3. is	10. is			3. is	27. is		17. is
		12. is	21. is			12. is	11. is		17. is
			16. is			12. is	11. is		
			24. is				11. is		

1000. İterasyon									
En Uygun değer: 72.2457617803043									
1. Makine	2. Makine	3. Makine	4. Makine	5. Makine	6. Makine	7. Makine	8. Makine	9. Makine	10. Makine
72.24576178	66.28241788	48.58935892	48.41673203	57.81267823	54.25634214	45.89128224	49.330667	39.53597348	45.668911
28. is	9. is	21. is	19. is	29. is	7. is	12. is	2. is	13. is	28. is
30. is	16. is	27. is	2. is	16. is	19. is	25. is	1. is	21. is	15. is
20. is	14. is	11. is	22. is	23. is	19. is	11. is	4. is	26. is	7. is
25. is	24. is	17. is	13. is	12. is	14. is	6. is	26. is		28. is
12. is	1. is	3. is	7. is	8. is	5. is	1. is	19. is		17. is
18. is	23. is	29. is	15. is		8. is	11. is	15. is		1. is
	10. is		8. is			5. is	2. is		8. is
	4. is		6. is			13. is	29. is		8. is
			26. is			29. is	29. is		
			5. is						



Şekil 7.25 Genetik algoritma kullanılarak iş sıralama probleminde n=500 için oluşturulan Gantt şeması

8. SONUÇLAR

Bu çalışmada, bir Endüstri Mühendisliği konusu olan "İş Sıralama" üzerinde Genetik Algoritma yöntemi incelenmektedir ve bu yöntemle benzer olmayan paralel makineler üzerinde (non-identical) iş sıralama problemine çözümler sunan bir programla uygulama yapılmaktadır. Burada yapılan çalışma sadece iş sıralamada genetik algoritma yaklaşımının kullanımı değildir. Aynı zamanda işlerin makinelere atanan proses zamanlarının bulanık zaman esaslı olarak ele alınarak genetik algoritma uygulamasına gidilmesi de araştırılmaktadır.

Günlük yaşantımızda, kesin olduğunu düşündüğümüz ancak gerçekte kesin olmayan durumlarla karşılaşırız. Bu durumların sistematik bir biçimde öngörülebilmesi ancak bazı kabullerin yapılmasından sonra mümkün olmaktadır. Birçok sosyal, ekonomik ve teknik olayda da belirsizlik ve dolayısıyla karmaşıklık bulunmaktadır. Bu belirsizliklerin analiz edilmesi Zadeh tarafından geliştirilen bulanık mantık teorisi kapsamında mümkündür.

Bu çalışmada, Java Eclipse Europa bilgisayar programı genetik algoritma için kullanılırken, 4 makine 9 iş, iş sıralama problemi ile beraber 10 makine 30 iş iş sıralama problemi incelenmiş ve işlerin makinelerdeki en uygun proses zamanları çeşitli iterasyonlar yapılarak araştırılmıştır. 100, 250, 500, 1000, 1500 ve 3000 kez iterasyon yapılmıştır. Buna göre en uygun zamanlamaya 250. iterasyonda ulaşılmaktadır. 250. iterasyonda 1. makineye 2.iş, 2. makineye 1.iş ve 9.iş, 3. makineye 6. iş, 3. iş, 8. iş, 4. makineye 7.iş, 5.iş, 4.iş atanmaktadır. Bulunan en iyi değer 13.439 uygunluk değeridir.

250 iterasyondan sonra bütün iterasyonlarda uygunluk değerleri aynı bulunmaktadır. Yani en iyi değere 250. iterasyonda ulaşılmıştır. 500. iterasyondan sonraki tüm iterasyonlar tekrar olarak kabul edilebilir. Burada 500. iterasyonda 1. makineye 2. iş, 2. makineye 1.iş ve 9.iş, 3. makineye 6.iş, 3.iş, 8.iş atanırken, 4. makineye 7. iş, 5. iş ve 4. iş atanmaktadır. Aynı şekilde 1000. iterasyonda ise 1. makineye 2.iş, 2. makineye 1. iş, 9.iş, 3. makineye 6.iş, 3.iş, 8. iş, 4. makineye 7.iş, 5.iş ve 4.iş atanmaktadır. Burada görülmektedir ki, 500. ve 1000. iterasyonlarda her bir makineye atanan iş sayıları birbirlerinin aynısı olmaktadır. Bu şekilde atamalar tamamlanmaktadır.

10 makine 30 iş iş sıralama problemi içinde de aynı şekilde 100, 250, 500, 1000 iterasyon incelenmiş ve sonuçlar kaydedilmiştir. Bu noktada 500. iterasyonda uygun değere ulaşıldığı görülmektedir.

Aslında 3000 iterasyona gidilen ilk örnekte iterasyon sayıları bu noktada fazlalık arz etmiştir. Gene de programın çalışabilirliğinin test edilmesi açısından bu konu önemli olmaktadır. İkinci uygulamada program tarafından en iyi değer, 72.245 uygunluk değeri olarak hesaplanmaktadır.

KAYNAKLAR

Alagöz O., Azizoğlu M., (2003), "Rescheduling of identical parallel machines under machine eligibility constraints", European Journal of Operational Research, 149, 2.

Alidaee B., Rosa D., (1997), "Scheduling parallel machines to minimize total weighted and unweighted tardiness", Computers Ops Res., 24, 1.

Anglani A., Grieco A., Guerriero E., Musmanno R., (2005), "Robust scheduling of parallel machines with sequence-dependent set-up costs", European Journal of Operational Research 161, 704–720.

Armentano V.A., Filho M.F., (2007), "Minimizing total tardiness in parallel machine scheduling with setup times: An adaptive memory-based GRASP approach", European Journal of Operational Research, 183, 1.

Arslan, S., Tuncer, T., Karcı, A., (2005), "Çoklu-Dizi Hizalama Problemi İçin Genetik Algoritma", Bilgisayar Mühendisliği Bölümü Mühendislik Fakültesi, Fırat Üniversitesi, Elazığ.

Başlıgil, H., (1988), "Sıralama ve Programlama", Yıldız Üniversitesi, 1. Baskı, İstanbul.

Baykal, N., Beyan, T., (2004), "Bulanık Mantık İlke ve Temelleri", Bıçaklar Kitabevi, No:9, Ankara.

Baksak, M., Erol V., (2004), "Sipariş Tipi Atölyelerde İş Sıralama Problemi İçin Bir Genetik Algoritma Uygulaması", Yöneylem Araştırması/Endüstri Mühendisliği - XXIV Ulusal Kongresi, Gaziantep – Adana.

Balin, S., Başlıgil, H., Alcan, P., (2007), "Paralel Makinelerin Genetik Algoritma Kullanılarak Çizelgelenmesi", 2. Ulusal Sistem Mühendisliği Sempozyumu, Yıldız Teknik Üniversitesi, Makine Fakültesi, Endüstri Müh. Bölümü, Yıldız-İstanbul.

Balin, S., Başlıgil, H., Alcan, P., (2008), "Non-identical parallel machine scheduling using genetic algorithm", Computers & Industrial Engineering, Yıldız Teknik Üniversitesi, İstanbul.

Balasubramanian J., Grossmann I.E., (2003), "Scheduling optimization under uncertainty/an alternative approach", Computers and Chemical Engineering 27, 469/490.

Bank J., Werner F., (2001), "Heuristic Algorithms for Unrelated Parallel Machine Scheduling with a Common Due Date, Release Dates, and Linear Earliness and Tardiness Penalties", Mathematical and Computer Modelling, 33, 14-19.

Baptiste P., Timkovsky V.G., (2001), "On preemption redundancy in schedulingunit processing time jobs on two parallel machines", Operations Research Letters, 28, 2.

Beraldi P., Ghiani G., Grieco A., Guerriero E., (2007), "Rolling-horizon and fix-and-relax heuristics for the parallel machine lot-sizing and scheduling problem with sequence-dependent set-up costs", Computers & Operations Research, 234-112.

Centeno G., Armacost R.L., (1997), "Parallel machine scheduling with release time and machine eligibility restrictions", Computers ind. Engng, 33, 1-2.

Chand S., McClurg T., Ward J., (2000), "A model for parallel machine replacement with capacity expansion", European Journal of Operational Research, 121, 1.

Chang P., Chen S., Lin K., (2005), "Two-phase sub population genetic algorithm for paralel machine-scheduling problem". Expert Systems with Applications, 29; 1-7.

Chen J.F., Wu T.H., (2006), "Total tardiness minimization on unrelated parallel machine scheduling with auxiliary equipment constraints", Omega, 34, 2.

Chiu N.C., Fang S.C., Lee Y.S., (2000), " Sequencing parallel machining operations by genetic algorithms", Computers & Industrial Engineering 36, 259-280.

Cochran J. K., Horng S.M., Fowler J.W., (2003), "A multi-population genetic algorithm to solve multi-objective scheduling problems for parallel machines", Computers & Operations Research 30, 1087–1102.

Dubois D., Fargier H., Fortemps P., (2003), "Fuzzy scheduling: Modelling flexible constraints vs. coping with incomplete knowledge", European Journal of Operational Research 147 , 231–252.

Dunstall S., Wirth A., (2005), "Heuristic methods for the identical parallel machine flowtime problem with set-up times", Computers & Operations Research, 32, 11.

Engin, O., Fığlalı, A., (2002), "Akış Tipi Çizelgeleme Problemlerinin Genetik Algoritma Yardımı İle Çözümünde Uygun Çaprazlama Operatörünün Belirlenmesi", Doğuş Üniversitesi Dergisi, 2002/6, 27-35.

Gharbi A., Haouari M., (2007), "An approximate decomposition algorithm for scheduling on parallel machines with heads and tails", Computers & Operations Research, 34, 15.

Gairing M., Monien B., Woclaw A., (2007), "A faster combinatorial approximation algorithm for scheduling unrelated parallel machines", Theoretical Computer Science, 380, 12.

Ghirardi M., Potts C.N., (2005), "Makespan minimization for scheduling unrelated parallel machines: A recovering beam search approach", European Journal of Operational Research, 165, 1.

Gökay, G., Taşkın, E. Ç., (2002), "Genetik Algoritmalar Ve Uygulama Alanları", Uludağ Üniversitesi, İktisadi ve İdari Bilimler Fakültesi Dergisi, Cilt XXI, Sayı 1, s. 129-152.

Gözütok, S., Özdemir O. N., (2004), "Genetik Algoritma Yöntemi İle Su Şebekelerinde Hidrolik Kalibrasyonun Geliştirilmesi", Gazi Üniv. Müh. Mim. Fak. Der., Cilt 19, No 2, 125-130.

Gupta J.N.D., Ho J.C., (2001), "Minimizing makespan subject to minimum flowtime on two identical parallel machines", Computers & Operations Research, 28, 1-4.

Gupta J.N.D, Torres A., (2005), "Generating efficient schedules for identical paralel machines involving flow-time and tardy jobs", European Journal of Operational Research, 167,1-4.

Hall N.G., Potts C.N., Sriskandarajah C., (2000), "Parallel machine scheduling with a common server", Discrete Applied Mathematics, 102, 1.

Hong T., Huang C.M., Yu K.M., (1998), "LPT scheduling for fuzzy tasks", Fuzzy Sets and Systems 97, 277 286.

İşçi, Ö., Korukoğlu, S., (2003), "Genetik Algoritma Yaklaşımı ve Yöneylem Araştırmasında Bir Uygulama", Yönetim ve Ekonomi, Yıl:2003 Cilt:10 Sayı :2.

Jou C., (2005), "A genetic algorithm with sub-indexed partitioning genes and its application to production scheduling of parallel machines", Computers & Industrial Engineering 48, 39–54.

Kahraman, C., Cebeci, U., Ruan, D., (2003), "Multi-attribute Comparison of Catering Service Companies Using Fuzzy AHP: The Case of Turkey", Int. J. Production Economics 87 (2004), 171–184.

Kıyak, E., Kahvecioğlu, A., (2003), "Bulanık Mantık Ve Uçuş Kontrol Problemine Uygulanması", Havacılık Ve Uzay Teknolojileri Dergisi, Cilt 1 Sayı 2 (63-72).

Kogan K., (2006), "Optimal scheduling of parallel machines with constrained resources", European Journal of Operational Research, 170, 1.

Koulamas C., Kyparisis G.J., (2007), "An improved delayed-start LPT algorithm for a partition problem on two identical parallel machines", European Journal of Operational Research, 1-2.

Koulamas C., Kyparisis G.J., (2000), "Scheduling on uniform parallel machines to minimize maximum lateness", Operations Research Letters, 26, 1.

Koulamas C., Kyparisis G.J., (2007), "A note on the two-stage assembly flow shop scheduling problem with uniform parallel machines", European Journal of Operational Research, 182, 1-3.

Kyparisis G.J., Koulamas C., (2006), "Flexible flow shop scheduling with uniform parallel machines", European Journal of Operational Research, 168, 3.

Laguna M., Lino P., Perez A., Quintanilla S., Valls V., (2000), "Minimizing weighted tardiness of jobs with stochastic interruptions in parallel machines", European Journal of Operational Research, 127, 1.

Leonardi S., Raz D., (2006), "Approximating total flow time on parallel machines", Journal of Computer and System Sciences, 73, 1-2.

Liao C.J., Lin C.H., (2003), "Makespan minimization for two uniform parallel machines. Int. J. Production Economics", 84, 4.

Liao L., Sheen G., (2007), "Parallel machine scheduling with machine availability and eligibility constraints", European Journal of Operational Research, 3-2.

Liaw C.F., Lin Y.K., Cheng C.Y., Chen M., (2003), "Scheduling unrelated parallel machines to minimize total weighted tardiness", Computers & Operations Research, 30, 1.

Litoiu M., Tadei R., (2001), "Fuzzy schedulingwith application to real-time systems", Fuzzy Sets and Systems 121, 523–535.

Liu M., Wu C., (2003), "Scheduling algorithm based on evolutionary computing in identical parallel machine production line", Robotics and Computer Integrated Manufacturing, 19, 6-7.

Logendran R., McDonell B., Smucker B., (2007), "Scheduling unrelated parallel machines with sequence-dependent setups", Computers & Operations Research, 34, 1-17.

Lopez M, Carvalho J.M., (2007), "A branch-and-price algorithm for scheduling parallel machines with sequence dependent setup times", European Journal of Operational Research, 176; 1.

Mandel M., Mosheiov G., (2001), "Minimizing maximum earliness on parallel identical machines", Computers & Operations Research, 28, 2-4.

Meyr H., (2002), "Simultaneous lotsizing and scheduling on parallel machines", European Journal of Operational Research, 139, 1.

Min L., Cheng W., (1999), "A genetic algorithm for minimizing the makespan in the case of scheduling identical parallel machines", Artificial Intelligence in Engineering 13, 399–403.

Mok P.Y., Kwong C.K., Wong W.K., (2007), "Optimisation of fault-tolerant fabric-cutting schedules using genetic algorithms and fuzzy set theory", European Journal of Operational Research 177, 1876–1893.

Mokotoff E., Chretienne P., (2002), "A cutting plane algorithm for the unrelated parallel machine scheduling problem", European Journal of Operational Research, 141-345.

Mosheiov G., (2001), "A common due-date assignment problem on parallel identical machines. Computers & Operations Research", 28, 13.

Moukrim A., Quilliot A., (2005), "Optimal preemptive scheduling on a fixed number of identical parallel machines", Operations Research Letters, 33, 1.

Muşdal H., (2005), "Performans Değerlendirme İçin Bulanık Yaklaşım", Bitirme Projesi, Yıldız Teknik Üniversitesi, İstanbul.

Nabiyev V.V., (2005), "Yapay zeka, problemler-yöntemler-algoritma", Seçkin Kitabevi, 2. Baskı, Ankara.

Nessah R., Yalaoui F., Chu C., (2006), "A branch-and-bound algorithm to minimize total weighted completion time on identical parallel machines with job release dates", Computers & Operations Research, 1.

Nezhad S. S., Assadi R. G., (2008), "Preference ratio-based maximum operator approximation and its application in fuzzy flow shop scheduling", Applied Soft Computing 8, 759–766.

Omar M.K., Teo S.C., (2006), "Minimizing the sum of earliness/tardiness in identical parallel machines schedule with incompatible job families: An improved MIP approach", Applied Mathematics and Computation, 283-224.

Öndemir Ö., (2004), "Hata Türü ve Etkilerinin Bulanık Kümeler Yaklaşımıyla Analizi", Yüksek Lisans Tezi, YTÜ Fen Bilimleri Enstitüsü, İstanbul.

Özdağlar, D., Benzeden, E., Kahraman, A.M., (2006), "Kompleks Su Dağıtım Şebekelerinin Genetik Algoritma ile Optimizasyonu", İMO Teknik Dergi, 3851 -3867, Yazı 253.

Park M., Kim Y., (1997), "Search Heuristics for a Parallel Machine Scheduling Problem with Ready Times and Due Dates", Computers ind. Engng, 33, 2-3-4.

Park Y., Kim S., Lee Y., (2000), "Scheduling jobs on parallel machines applying neural network and heuristic rules", Computers & Industrial Engineering, 38, 13.

Peng J., Liu B., (2004), "Parallel machine scheduling models with fuzzy processing times", Information Sciences, 166, 1.

Rocha P., Ravettia M., Mateus G., Pardalos P., (2006), "Exact algorithms for a scheduling problem with unrelated paralel machines and sequence and machine-dependent setup times", Computers & Operations Research, 154-234.

Santos E.E., (2002), "Optimal and Efficient Algorithms for Summing and Prefix Summing on Parallel Machines", Journal of Parallel and Distributed Computing, 62, 1.

Shabtay D., Kaspi M., (2006), "Parallel machine scheduling with a convex resource consumption function. European Journal of Operational Research", 173, 3-13.

Shim S.O., Kim Y.D., (2006), "A branch and bound algorithm for an identical parallel machine scheduling problem with a job splitting property". Computers & Operations Research, 134-222.

Silva C., Magalhaes J.M., (2006), "Heuristic lot size scheduling on unrelated parallel machines with applications in the textile industry", Computers & Industrial Engineering, 50, 1-4.

Sun H., Wang G., (2003), "Parallel machine earliness and tardiness scheduling with proportional weights", Computers & Operations Research, 30, 1.

Şen, Z., (2004), "Mühendislikte Bulanık (Fuzzy) Mantık ile Modelleme Prensipleri", Su Vakfı Yayınları, İstanbul.

Tahar D.N., Yalaoui F., Chu C., Amodeo A., (2005), "A linear programming approach for identical parallel machine scheduling with job splitting and sequence-dependent setup times", Int. J. Production Economics, 99, 1.

Tanyıldızı, H., Yazıcıoğlu, S., (2006), "Bulanık Mantık Metodu ile Tekil Yükler Altında İki Açıklıklı Kirişlerin Plastik Göçme Yük Faktörü Değerinin Bulunması", İMO Teknik Dergi, 3961-3971, Yazı 262.

Toledo F.M.B., Armentano V.A., (2006), "A Lagrangian-based heuristic for the capacitated lot-sizing problem in parallel machines", European Journal of Operational Research, 175, 1.

Torres A, Lo´pez F, Ho J. Scheduling uniform parallel machines subject to a secondary resource to minimize the number of tardy jobs. European Journal of Operational Research 2007; 179; 1-13.

Uçaner, M.,. Özdemir, O. N., (2002), "Genetik Algoritmalar İle İçme Suyu Şebekelerinde Ek Klorlama Optimizasyonu", Gazi Üniv. Müh. Mim. Fak. Der., Cilt 17, No 4, 157-170.

Weng M.X., Lu J., Ren H., (2001), "Unrelated parallel machine scheduling with setup consideration and a total weighted completion time objective", Int. J. Production Economics, 70, 1.

Webster S., Azizoglu M., (2001), "Dynamic programming algorithms for scheduling parallel machines with family setup times", Computers & Operations Research, 28, 1.

Yager, R., Filev, D., (1994), "Essentials of Fuzzy Modeling and Control", John Wiley&Sons Inc., Canada.

İNTERNET KAYNAKLARI

[1] http://www.mmo.org.tr/muhendismakina/arsiv/2001/ekim/Genetik_Algoritma.htm

[2] http://www.yapay-zeka.org

ÖZGEÇMİŞ

Doğum tarihi	12.05.1981	
Doğum yeri	İstanbul	
İlkokul	1988–1992	Ümraniye Zübeyde Hanım İlkokulu
Ortaokul	1992-1995	Üsküdar Cumhuriyet Lisesi
Lise	1995–1999	Haydarpaşa Lisesi
Lisans	2000–2004	Yıldız Teknik Üniversitesi Kimya Metalurji Fakültesi Metalurji Malzeme Mühendisliği Bölümü
	2002-2006	Yıldız Teknik Üniversitesi Makine Fakültesi Endüstri Mühendisliği Bölümü
Yüksek Lisans	2006–2008	Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Endüstri Mühendisliği Anabilim Dalı