GENETİK PROGRAMLAMA İLE ÖNCELİK KURALLARI VE ÇİZELGELEME ALGORİTMALARININ KEŞFİ

Mete Alikalfa

DOKTORA TEZİ

İstatistik Anabilim Dalı

Haziran 2013

DISCOVERING PRIORITY RULES AND SCHEDULING ALGORITHMS USING GENETIC PROGRAMMING

Mete Alikalfa

DOCTORAL DISSERTATION

Statistics

June 2013

GENETİK PROGRAMLAMA İLE ÖNCELİK KURALLARI VE ÇİZELGELEME ALGORİTMALARININ KEŞFİ

Mete Alikalfa

Eskişehir Osmangazi Üniversitesi
Fen Bilimleri Enstitüsü
Lisansüstü Yönetmeliği Uyarınca
İstatistik Anabilim Dalı
Yöneylem Araştırması Bilim Dalında
DOKTORA TEZİ
Olarak Hazırlanmıştır

Danışman: Prof. Dr. Muzaffer Kapanoğlu

Haziran 2013

ONAY

İstatistik Anabilim Dalı Doktora öğrencisi Mete ALİKALFA'nın DOKTORA tezi olarak hazırladığı "Genetik Programlama ile Öncelik Kuralları ve Çizelgeleme Algoritmalarının Keşfi" başlıklı bu çalışma, jürimizce lisansüstü yönetmeliğin ilgili maddeleri uyarınca değerlendirilerek kabul edilmiştir.

Danışman **:** Prof. Dr. Muzaffer KAPANOĞLU

İkinci Danışman **: -**

Doktora Tez Savunma Jürisi:

Üye : Prof. Dr. Muzaffer KAPANOĞLU

Üye : Prof. Dr. Şenol ERDOĞMUŞ

Üye : Prof. Dr. Harun TAŞKIN

Üye : Doç. Dr. H. Kıvanç AKSOY

Üye : Yrd. Doç. Dr. Nihat ADAR

Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ..... tarih ve ..... sayılı kararıyla onaylanmıştır.

Prof. Dr. Nimetullah BURNAK

Enstitü Müdürü

ÖZET

Günümüzün verimlilik odaklı dünyasında çizelgeleme önemli ekonomik fırsatlara sahip kritik bir problemdir. Diğer yandan, çizelgeleme problemleri birçok optimizasyon probleminin belkemiği olarak değerlendirilebilir. Bu problemlerin fırsatları ve zorluğu yıllar içinde birçok araştırmacının ilgisini çekmiştir. Çizelgeleme problemlerinin çeşitliliği ve karmaşıklığı sonucunda birçok çizelgeleme kuralı ve algoritma ortaya çıkmıştır. Bunların çoğu kabul edilebilir hesaplama sürelerinde yaklaşık sonuçlar bulmakta, bazıları ise optimum çözüm sunmaktadır.

Bu doktora tezinde, çizelgeleme problemlerinin çözümü amacıyla çizelgeleme kuralları ve algoritmalarının otomatik keşfi ile ilgili bir yapı sunulmaktadır. Bu çalışmanın altında yatan motivasyon, tam otomatik çizelgeleme sistemlerinin olanaklı hale getirilmesidir. Bu tür sistemlerin hayata geçirilebilmesi endüstriyel sistemlerden lojistik ve servis sistemlerine kadar geniş bir alanda, anında ve önemli bir katkı sağlayabilecektir. Uzun vadede, çizelgeleme kurallarının ve algoritmaların otomatik keşfi, çizelgelemenin ötesinde herhangi bir problem için algoritmik çözümlerin keşfine olanak sağlayabilecektir.

Çizelgeleme kurallarının ve algoritmaların otomatik keşfi çoklu ifade genetik programlama tekniği kullanılarak ve öncül alan bilgileri verilmeden gerçekleştirilmiştir. Geliştirilmiş öğrenme sistemi, çizelgeleme kuralları ve algoritmaları çok basit operatörleri ve problemle ilgili nitelikleri kullanarak sıfırdan keşfetmiştir. Deneysel sonuçlarımız keşfettiğimiz çizelgeleme kuralları ve algoritmaların literatürdekilerden çok daha iyi olduğunu göstermektedir. Buna ek olarak, aynı yaklaşımla iyi bilinen iki algoritma da keşfedilmiştir. Elde ettiğimiz sonuçlar araştırmacıları çizelgeleme problemlerini yeni bir doğrultuda ele almak bakımından cesaretlendirecektir: Problemlerin kendilerini çözmek yerine yeni çizelgeleme kuralları ve algoritmaların keşfinin tercih edilmesi.

Anahtar Kelimeler: Genetik Programlama, Keşif, Öncelik Kuralları, Çizelgeleme Algoritmaları

SUMMARY

In today's productivity-oriented world, scheduling is a critical problem having significant economic opportunities. On the other hand, scheduling problems can be considered as the backbone of a wide range of combinatorial optimization problems. The opportunities as well as challenges of the scheduling problems have attracted many researchers over years. The variety and the complexity of scheduling problems have resulted in a list of dispatching rules and algorithms to address the practitioners needs mostly approximate solutions with acceptable computation times and few with optimal solutions.

This dissertation proposes the automated discovery of dispatching rules and algorithms for scheduling problems. The underlying motive of the study is to enable fully automated scheduling systems with no supervision. The realization of such systems may seem to have immediate and significant contributions on the applications of the industrial systems to logistics and service systems as well. In the long-term, the impact of the automated discovery of dispatching rules and algorithms may lead to discover algorithmic solutions solely on computational intelligence for any problem beyond scheduling.

The automated generation of dispatching rules and algorithms have been achieved by using a novel multi-expression genetic programming with no a priori domain knowledge. The developed multi-expression genetic programming have discovered the dispatching rules and algorithms from scratch by using very simple operators and attributes related to the problem in hand. Experimental results show that automatically discovered priority rules and scheduling algorithms by our genetic programming framework outperforms the literature. Moreover, the two well-known algorithms have been discovered by the same approach. We expect the results will encourage researchers towards a new direction in dealing with scheduling problems: Discovering a new dispatching rule or algorithm might be preferred over solving the problem itself.

Keywords: Genetic Programming, Discovery, Priority Rules, Scheduling Algorithms

TEŞEKKÜR

Doktora çalışmalarında, gerek derslerimde ve gerekse tez çalışmalarında, bana danışmanlık ederek, beni yönlendiren ve her türlü olanağı sağlayan danışmanım Prof. Dr. Muzaffer Kapanoğlu'na ve aileme teşekkür ederim.

İÇİNDEKİLER

Sayfa

İÇİNDEKİLER (devam)

Sayfa

İÇİNDEKİLER (devam)

Sayfa

İÇİNDEKİLER (devam)

Sayfa

ŞEKİLLER DİZİNİ

ŞEKİLLER DİZİNİ (DEVAM)

ÇİZELGELER DİZİNİ

ÇİZELGELER DİZİNİ (DEVAM)

BÖLÜM 1
GİRİŞ

Günümüz üretim sistemlerinde, üretim planlama ve çizelgeleme önemli bir karar verme sürecidir. Çizelgeleme, ekonomik olarak çok önemli olan ama problemin yapısı gereği eniyilenmesi oldukça zor olan bir işlemdir. Genel anlamda, bir amaca bağlı olarak bir takım kaynakları (tezgah vb.) kullanarak bazı görevleri (iş vb.) yerine getirme problemidir. Çizelgeleme problemleri; üretim sistemleri, işletim (bilgisayar) sistemleri, internet servisleri, lojistik gibi birçok farklı alanda ortaya çıkmaktadır. Dolayısıyla çizelgeleme alanında yapılacak her türlü iyileştirme birçok uygulama alanı bulacaktır (Bruns, 1997).

Bu çalışmada tek tezgahlı üretim sistemlerinde çizelgeleme (single machine scheduling) konusu ile ilgilenilecektir. Tek tezgahlı çizelgeleme problemleri, sıralanacak olan işlerin işlem sürelerinin deterministik ve baştan belli olduğu durumda temel çizelgeleme problemlerinden birini oluşturmaktadır. Bu problemler birçok farklı amaç (performans ölçütü) ve çeşitli çözüm tekniklerinin incelenebilmesi için bir ortam sunmaktadır. Buna bağlı olarak çizelgeleme kavramının kapsamlı şekilde anlaşılabilmesi ve incelenebilmesi için bir yapı taşıdır. Daha karmaşık çizelgeleme problemlerinin davranışlarının gerektiği şekilde anlaşılabilmesi için öncelikle bu karmaşık sistemlerin parçalarının anlaşılması gereklidir. Bütün çizelgeleme problemlerinin ortak ve en temel parçası ise tek tezgah çizelgeleme problemi olduğundan bu problem türü çalışmalarımızın ana odak noktası olarak seçilmiştir.

Bu çalışmada ele alınacak olan problemler ikiden fazla işe sahip olduğu için "NP-Zor" problem olarak sınıflandırılmaktadır (Lageweg, et al., 1977). İlgili problemler, çizelgeleme işlemi başlamadan işlem görecek bütün işlerin bilinmesi nedeniyle statik ve bütün parametrelerin bilinmesi nedeniyle de belirli (deterministic) çizelgeleme problemidir (Holthaus, 1997). Üzerinde çalışılan problemler statik ve

deterministik olmasına rağmen önerilen sistem ile bulunan çözüm yöntemleri dinamik ve / veya stokastik problemlere de uygulanabilir durumda olacaktır.

Üretim sistemlerinde çizelgeleme probleminin çözümü amacıyla kullanılabilecek teknikler eniyileme teknikleri ve sezgisel teknikler olmak üzere iki sınıfa ayrılırlar. Eniyileme teknikleri, en iyi çözümü garanti ederken büyük boyutlu problemlerde çözüme ulaşması çok uzun süre alan tekniklerdir. Buna karşın sezgisel teknikler en iyi çözümü garanti edemezler. Sezgisel tekniklerin kullanım nedeni ise eniyileme tekniklerine nazaran çok daha kısa sürede en iyi ya da en iyiye yakın çözümler sunmasıdır. Fakat sezgisel teknikler en iyi çözüme ulaşabilse bile bu çözüme ulaşmayı garanti etmemektedir. Eniyileme tekniklerine dal-sınır algoritması ve dinamik programlama örnek olarak verilebilir. Çizelgeleme problemleri için kullanılabilecek sezgisel tekniklere öncelik kuralları, yapay zeka teknikleri, yapay sinir ağları, meta sezgiseller (genetik algoritma, yasaklı arama, benzetimli tavlama vb.), bulanık mantık, otomatik öğrenme (machine learning) teknikleri ve melez yaklaşımlar örnek olarak verilebilir.

Üretim sistemlerinde çizelgeleme problemine kısa zamanda iyi çözümler vermesi amacıyla tasarlanmış olan öncelik kuralları, çizelgeleme problemlerine yaygın olarak uygulanmıştır. Literatürde öncelik kuralı terimi çizelgeleme kuralı, sıralama kuralı ve yönlendirme kuralı olarak da isimlendirilmiş (Panwalker, 1977; Blackstone, 1982; Baker, 1974) ve birçok farklı kural türetilmiştir (Pinedo, 1995).

Çok az sayıda özel problem ya da performans ölçütü için optimum çözüm veren öncelik kuralları ve çizelgeleme algoritmaları geliştirilmesine rağmen her koşulda iyi sonuç veren bir öncelik kuralı ya da çizelgeleme algoritması olmadığından, çözüm için hangi yöntemin kullanılacağı çizelgeleme ortamının durumuna ya da amaçlanan hedefe göre değişmektedir. Ayrıca, üretim sistemlerinin yapısından kaynaklanan bütün kısıtları temsil edebilen ve değişen ortama göre eylem sergileme yeteneği olan bir çizelgeleme sisteminin kurulması çizelgeleme ortamında meydana gelen alışılmadık veya öngörülmemiş olayların çözülebilmesine yardımcı olabilir. Tabii olarak, bu durumda bilgi tabanlı zeki yaklaşımlar kullanılması ile daha etkin çözümlerin üretilebileceği gözlenmektedir. Dolayısıyla geliştirilecek olan çizelgeleme sisteminin zeki davranışlar

göstermesi ya da arka planda yer alan bilgilerin bilgi madenciliği gibi tekniklerle ortaya çıkartılması hem problemin çözümü için bir avantaj olacak hem de daha sonraki problemlere uygulanmasını kolaylaştıracaktır.

Zeki davranışların en önemli temel özelliklerinden olan öğrenme, bir işi yapabilmek ya da bir duruma tepki verebilmek için bulunulan durumdan ve/veya çevreden gerekli olan bilgiyi elde etmek, elde edilen bu bilgiyi karar verme sürecinde kullanmak ve verilen kararların sonucunda mevcut bilgileri yeniden değerlendirebilmektir (Russell, 1995). Öğrenme ve karar verme, günümüze kadar üzerinde en çok ilgilenilen alanlardan biri olmuştur. Bu amaç doğrultusunda birçok algoritma geliştirilmiş ve çeşitli alanlara başarı ile uygulanmıştır. Basit problemler üzerinde elde edilen başarılar, bu tür algoritmaların daha karmaşık problemlere uygulanması konusunda araştırmacılara güven vermiştir.

Mevcut durumlardan öğrenme, karar verme ve elde ettiği bilgileri genelleştirerek veya onlardan yeni bilgiler ortaya çıkararak kullanma özelliklerine sahip tekniklerden birisi genetik programlama tekniğidir. Genetik programlama tekniği verilen yapı taşlarını (bilgi, bilgi yapısı vb.) kullanarak ürettiği yeni bilgi ve yapıları seçilen ortamlardaki performanslarına bağlı olarak değerlendiren ve bunları iyileştiren bir tekniktir. Genetik programlamanın kullandığı yapı taşları klasik değişkenler, sabitler ve operatörler (aritmetik, matematiksel vb.) olabileceği gibi diğer teknikleri de (karar ağaçları, sezgisel algoritmalar, yapay sinir ağları vb.) yapı taşı olarak kullanabilir. Genetik programlama belirli bir görevi yerine getirmek ya da bilgiyi öğrenmek için ortamla etkileşim halindedir. Bu etkileşim, genetik programlama ile oluşturulan bilgi, kural veya yapının ortama uygulanması ve uygulama sonucu ortamın sisteme gönderdiği geri besleme (feedback) ile olur. Bu geri besleme sayesinde genetik programlama uyguladığı bilgi, kural veya yapıların iyi ya da kötü olduğunu anlar ve onları geliştirmeye çalışır. Genetik programlamanın asıl amacı ortamdan aldığı geri beslemeyi mümkün olduğu kadar iyileştirmektir.

Bu çalışmada yukarıda bahsedilen nedenlerden dolayı öncelik kuralları ve çizelgeleme algoritmalarını tek tezgahlı çizelgeleme problemleri için zeki bir şekilde geliştirme becerisine sahip olan bir çizelgeleme sistemi geliştirilmiştir. Çalışma iki

aşamadan oluşmaktadır. Birinci aşamada genetik programlama ile öncelik kurallarının birleştirilmesi yoluyla tek tezgahlı üretim sistemlerinde çizelgeleme probleminin çözümü için kullanılabilecek bir çizelgeleme sistemi geliştirilmiştir. Böylece öncelik kurallarının basitliği ve kolay uygulanabilirliği ile genetik programlamanın öğrenme olgusu birleştirilmiştir. Bu aşamada literatürde daha önce yapılan çalışmalarda araştırmacılar tarafından akademik çalışmalar sonucunda geliştirilen ya da çok az sayıda problem kullanılarak yapısı önceden belirlenerek geliştirilen öncelik kurallarının yerine çeşitli yöntemlerle bilgi madenciliği yapabilen zeki bir sistem ile öncelik kuralları oluşturulmuştur. İkinci aşamada ise çizelgeleme algoritmaları oluşturabilen bir zeki çizelgeleme sistemi geliştirilmiştir. Böylece çizelgeleme problemlerinin çözümünde daha esnek, birden çok problemin iç dinamiklerini kavrayabilen ve neden-sonuç ilişkisini gösterebilen çizelgeleme algoritmalarına ulaşılmaya çalışılmıştır.

Çalışmanın ikinci bölümünde tek tezgahlı üretim sistemleri ve genel karakteristikleri tanımlanmış ve söz konusu sistemde çizelgeleme problemi üzerinde durulmuştur. Üretim sistemlerinde çizelgeleme problemleri için kullanılan performans ölçütleri ve ilgili problemlerin çözümlerinde kullanılan yaklaşımlara kısaca değinilmiştir.

Üçüncü bölümde ilk olarak öncelik kuralları ve standart genetik programlama sistemi tanıtılmış ve genetik programlamanın yapısı, işleyişi ve bu yöntemin neden seçildiği üzerinde durulmuştur. Ardından bu çalışma kapsamında kullanılacak olan ve standart genetik programlamanın daha sonraki yıllarda geliştirilmiş bir şekli olan Çoklu İfade Genetik Programlama (Multi Expression Programming), genel yapısı ve işleyişi ele alınmıştır. Ardından literatürde bu konu ile ilgili daha önceden yapılmış olan çalışmalar listelenmiştir.

Dördüncü bölüm öncelik kurallarının çoklu ifade genetik programlama ile keşfi çalışmalarına ayrılmıştır. Bu bölümde ilk olarak geliştirilen çoklu ifade programlama tabanlı öğrenme sistemimizin genel yapısı, bileşenleri ve bu bileşenlerin çalışma yapıları verilmiştir. Ardından geliştirdiğimiz sistem ile öncelik kurallarının keşfinde kullanılacak olan test problemleri, elde ettiğimiz çözümler ve sonuçlar verilmiştir.

Beşinci bölüm çizelgeleme algoritmalarının çoklu ifade genetik programlama ile keşfi çalışmalarına ayrılmıştır. Bu bölümde ilk olarak geliştirilen çoklu ifade programlama tabanlı algoritma öğrenme sistemimizin genel yapısı, bileşenleri ve bu bileşenlerin çalışma yapıları verilmiştir. Ardından geliştirdiğimiz sistem ile çizelgeleme algoritmalarının keşfinde kullanılacak olan test problemleri, elde ettiğimiz çözümler ve sonuçlar verilmiştir.

Altıncı bölüm çoklu ifade genetik programlama ile çizelgeleme problemlerinin çözümünde kullanılmak amacıyla öncelik kuralları ve çizelgeleme algoritmalarının keşfi konusu ile ilgili sonuçlara ayrılmıştır.

BÖLÜM 2
TEK TEZGAHLI ÜRETİM SİSTEMLERİNDE ÇİZELGELEME

Bu bölümde tek tezgahlı üretim sistemleri ile ilgili genel karakteristikler tanımlanmakta ve tek tezgahlı üretim sistemlerinde çizelgeleme problemi üzerinde durulmaktadır. Ardından çizelgeleme problemlerinde kullanılan performans ölçütleri sunulmakta ve çizelgeleme problemlerinin çözümünde kullanılan çizelgeleme tekniklerine kısaca değinilmektedir.

2.1 Tek Tezgahlı Üretim Sistemleri

Üretim, bir fayda yaratmak amacıyla yeni bir fiziksel ürün veya hizmetin ortaya konması faaliyetidir (Kobu, 2006). Üretim süreci ise malzeme, sermaye, iş gücü, makine ve yönetim bilgisinin ürün ve/veya hizmete dönüşümüdür.

Üretim sistemleri birçok şekilde sınıflandırılabilmektedir. Üretim sistemleri üretimin akış tipine göre kesikli veya sürekli olmak üzere ikiye ayrılırlar. Kesikli üretim, siparişe göre üretim ya da partiler şeklinde üretim yapan sistemlerdir. Sürekli üretimde ise belirli bir tür ürün çok miktarda belirli bir akış hızında seri biçimde üretilir.

Üretim sistemlerini bir diğer sınıflandırma şekli ise çizelgeleme ortamına göre sınıflandırmadır. Üretim sistemleri çizelgeleme ortamına göre statik ve dinamik olarak sınıflandırılır. Eğer çizelgelenecek bütün işler, çizelgeleme işlemi başlamadan önce biliniyor, sistemde mevcut ve zaman içinde elimizdeki mevcut işlere yeni işler eklenmiyorsa, statik üretim sistemi statik olarak adlandırılır. Eğer zaman içinde yeni işlerin gelmesine olanak veriliyorsa, buna dinamik üretim sistemi denir.

Ayrıca üretim sistemleri belirli (deterministik) ve olasılıklı (stokastik) olarak da sınıflandırılır (French, 1982). Eğer işlerin süreleri ve diğer parametreler belirli ve sabit

ise bu problem deterministik, değil ise olasılıklı (stokastik) veya bulanıktır (posibilistik).

Üretim sistemlerinin en temel hali tek tezgahlı üretim yapılan sistemlerdir. Genel olarak, tezgah sayısının artması, üretimin akış tipi ve tezgah yerleşimine bağlı olarak diğer üretim sistemleri (akış tipi, atölye tipi vb.) ortaya çıkmaktadır.

Üretim sistemlerinde bir parçanın üretimi için çeşitli tezgahlarda çeşitli üretim aşamaları (rota) gereklidir. Tek tezgahlı bir üretim sisteminde bütün işler zorunlu olarak bir tek tezgahtan geçerken, birden fazla tezgahlı üretim sistemlerinde parçalar birden fazla rotaya sahip olabilmektedir. Yani her parça üretilmek için aynı sıra ile aynı tezgahlardan geçmeyebilir. Bu sebepten dolayı atölye ortamının herhangi bir andaki durumunu tam olarak izlemek veya kontrol etmek daha güçtür.

Bir üretim sistemi çizelgeleme probleminin çözüm uzayı, problemdeki tezgah ve iş sayılarının yanı sıra problemin çözümünde yapılan varsayımlara (işlerin tamamlanmadan tezgahtan ayrılabilmesi, tezgahların boş kalmasına izin verilmesi, bir tezgahta aynı anda sadece bir iş yapılabilmesi vb.) bağlı olarak değişmektedir (Pinedo, 1995).

Bu çalışmada ele alınan tek tezgahlı üretim sistemleri için kabul edilen varsayımlar aşağıdaki gibidir:

1. Bir iş üzerinde aynı anda birden fazla işlem yapılamaz.
2. Her iş mutlaka bitirilmelidir. Yani bir iş tamamen bitmeden tezgahtan ayrılamaz.
3. İşlerin taşıma süreleri sıfır kabul edilmektedir.
4. İşlerin tezgahlardaki hazırlık süreleri işin işlem görme sırasından bağımsızdır ve işlem sürelerine eklenmiştir.
5. Üretim yapılan tezgah kuyruğunda iş mevcut olduğu sürece boş kalamaz.
6. Bir tezgah aynı anda sadece bir tane işlem yapabilir.
7. Her bir işin belirli ve sabit bir işlem görme süresi vardır.

2.2 Üretim Sistemlerinde Çizelgeleme

Çizelgeleme, ekonomik olarak çok önemli ama gerçekleştirilmesi oldukça zor olan bir işlemdir. Genel anlamda çizelgeleme, bir amaca bağlı olarak bir takım kaynakları (tezgah vb.) kullanarak bazı görevleri (iş vb.) yerine getirme işlemidir. Üretim sistemlerinde çizelgeleme ise, bir ya da daha fazla performans ölçütünü göz önüne alacak biçimde sisteme gelen işleri tezgahlara atamak ve bu işlerin ilgili tezgahlardaki üretim sırasını belirlemektir.

Çizelgeleme problemleri; üretim sistemleri, bilgisayar işletim sistemleri, internet servisleri, ulaşım ve lojistik gibi birçok farklı alanda ortaya çıkmaktadır. Dolayısıyla çizelgeleme alanında yapılacak her türlü iyileştirme birçok uygulama alanı bulmaktadır (Bruns, 1997).

Bir üretim sisteminde tek tezgah çizelgelemede, işlerin ilgili tezgahtaki üretim sırasının belirlenmesi işlemi yapılmaktadır. Tek tezgahlı bir üretim sistemindeki çizelgeleme çözüm uzayı diğer üretim sistemlerinin çözüm uzayına göre daha küçüktür. Buna rağmen iş sayısı n olmak üzere mümkün olan çözüm sayısı n! olarak ortaya çıkmaktadır. n adet iş ve m adet tezgahın söz konusu olduğu atölye tipi üretim sistemlerinde ise çözüm uzayı $(n!)^m$ olmaktadır. Bu nedenle ikiden fazla iş ve/veya tezgah olduğu durumlarda ilgili çizelgeleme problemi literatürde NP-Hard olarak sınıflandırılmaktadır. Yani ilgili problemin çözümünü polinom zamanda bulabilecek bir algoritma mevcut değildir ve probleme iş eklendikçe çözüm uzayı üstel olarak artmaktadır (Lageweg, et al., 1977).

2.3 Performans Ölçütleri

Oluşturulan çizelgelerin ne kadar iyi olduğunun değerlendirilebilmesi amacıyla performans ölçütleri kullanılmaktadır. Literatürde birçok performans ölçütü mevcuttur ve bunlar teslim zamanına, tamamlanma zamanına ve stoklama ve tezgah kullanım

oranlarına bağlı performans ölçütleri olarak sınıflandırılmaktadır (French, 1982). Bu çalışmada işlerin teslim zamanına bağlı performans ölçütleri kullanılmıştır.

Performans ölçütlerinin açıklanabilmesi için gereken bazı tanımlar ve gösterimler Çizelge-2.1'de verilmiştir.

Çizelge-2.1: Kullanılan bazı gösterimler ve tanımları

Gösterim	**Ad**	**Tanım**
d_i	Due Date	i işinin teslim edilmesi gereken zamandır.
r_i	Ready Time	i işinin sistemde hazır olma zamanıdır. Bir iş ancak hazır olma zamanından sonra işlenmeye başlayabilir.
p_{ij}	Processing Time	i işinin j. tezgahta işlenmesi için gerekli süredir.
s_i	Slack	i işinin gecikmeden tamamlanabilmesi için kalan mevcut süredir. $s_i = d_i - r_i - T_{now}$ şeklinde hesaplanabilir. T_{now} hesaplamanın yapıldığı zamandır.
C_i	Completion Time	i işinin tamamlanma zamanıdır.
L_i	Lateness	i işinin tamamlanma zamanının teslim zamanından sapmasıdır. Bir iş teslim zamanından önce bitirilmişse negatif, teslim zamanından sonra bitirilmişse pozitif değer alır. $L_i = C_i - d_i$ şeklinde hesaplanabilir.
E_i	Earliness	i işinin tamamlanma zamanının teslim zamanından negatif yönde sapmasıdır. Yani işin erken bitirilmesidir. $E_i = \max \{ -L_i, 0 \}$ şeklinde hesaplanabilir.
T_i	Tardiness	i işinin tamamlanma zamanının teslim zamanından pozitif yönde sapmasıdır. Yani gecikmedir. $T_i = \max \{ L_i, 0 \}$ şeklinde hesaplanabilir.

Teslim zamanına bağlı performans ölçütleri hedeflenen teslim zamanlarından sapmaya bakmaktadır. Bu performans ölçütlerinden en çok kullanılanlar $\sum T$, $\overline{T}$ ve T_{max}'dır.

Teslim zamanlarından pozitif sapmaların (gecikme) toplamı olan $\sum T$ bu çalışmada kullanılan performans ölçütüdür ve $\sum_i (C_i - d_i)$ formülü ile hesaplanmaktadır. $\sum T$, $\overline{T}$ ve T_{max} ölçütleri işlerin erken bitirilmesi durumunda bir kazanç yoksa ve geç bitirilmesi durumunda bir cezaya katlanılıyorsa kullanılmaktadır. Bazı durumlarda işlerin ne kadar geç kaldığının bir önemi yoktur. Asıl önemli olan işin gecikmiş olmasıdır. Bu durumda geç kalmış işlerin sayılarının toplamı ölçütünün kullanılması gerekmektedir (French, 1982).

2.4 Çizelgeleme Yaklaşımları

Üretim sistemlerinin çizelgelenmesinde kullanılan yaklaşımlar Eniyileme Teknikleri ve Sezgisel Teknikler olarak ikiye ayrılırlar (Jones and Rabelo, 1998).

Eniyileme teknikleri, en iyi çözümü garanti ederken büyük boyutlu problemlerde çözüme ulaşması çok uzun zaman alan tekniklerdir. Sezgisel teknikler ise en iyi çözümü garanti edemezler. Buna karşın, sezgisel teknikler eniyileme tekniklerine nazaran çok daha kısa sürede en iyi ya da yeterli iyilikte çözümler sunmaktadırlar.

En iyi çözümü bulmayı garanti eden eniyileme teknikleri arasında dal-sınır algoritması (branch and bound) ve dinamik programlama (dynamic programming) örnek olarak verilebilir. Bu tür tekniklerin en temel halinde sayımlama yapılmaktadır. Yani mümkün olan her çözüm tek tek değerlendirilerek en iyi çözüm bulunmaktadır. Fakat çözüm uzayı büyüdükçe bu yaklaşımın çözüm bulması çok uzun süre almaktadır. Daha gelişmiş teknikler ise çeşitli yöntemlerle çözüm uzayını indirgeyerek çözüme daha kısa sürede ulaşmaya çalışmaktadırlar.

Sezgisel teknikler, eniyileme teknikleri gibi en iyi çözüme ulaşmayı garanti edemezler. Buna karşın en iyi ya da en iyiye yakın sonuçları çok daha kısa sürede elde edebilirler. Bu yüzden büyük boyutlu problemlerde eniyileme tekniklerine nazaran kullanımları çok daha yaygındır. Sezgisel tekniklere örnek olarak öncelik kuralları (priority rules), genetik algoritmalar (genetic algorithms), genetik programlama (genetic

programming), yapay sinir ağları (atrificial neural networks), bulanık mantık (fuzzy logic), otomatik öğrenme (machine learning) teknikleri ve melez (hybrid) yaklaşımlar verilebilir.

BÖLÜM 3

GENETİK PROGRAMLAMA

Genetik programlama, genetik algoritma tekniğinin bir uzantısı olarak geliştirilmiştir. Genetik algoritma, belirli sayıda aday çözüm üzerinde seçim, çaprazlama ve mutasyon operatörlerini kullanarak daha iyi çözümler elde etmeye çalışan bir sezgisel arama tekniğidir. İlk genetik programlama uygulaması 1992 yılında John Koza (Koza, 1992) tarafından genetik algoritma kullanılarak çeşitli görevleri yerine getiren bilgisayar programlarının geliştirilmesi ile ortaya çıkmıştır. Genetik programlama, bu çalışmada ele alınan üretim sistemlerinde çizelgeleme problemine kısa zamanda iyi çözümler üretmek amacıyla tasarlanmış ve henüz bilinmeyen öncelik kurallarını ve yeni çizelgeleme algoritmalarını geliştirmek amacıyla kullanılmıştır. Genetik programlama, bilgi tipi (biçimi) belirlenmiş olan ve / veya hiçbir şekilde biçimin belirlenmediği ama bilginin yapı taşları (temel veriler) kümesinin belirlendiği bilgiler üzerinde çalışabilecek kapasitede bir tekniktir. Buna ek olarak genetik programlama ile gerekli olduğu durumlarda yeni yapı taşları otomatik olarak belirlenip çalışma esnasında eldeki yapı taşları kümesi, bilgi kümesi ve ilişkiler kümesi genişletilebilir. Genetik programlamanın bu becerileri yeni öncelik kurallarının geliştirilmesi amacıyla kullanıldığından dolayı bu bölümde ilk olarak öncelik kuralları ile ilgili bilgi verilecektir. Ardından genetik programlama ve çalışmamızda kullanılan bir özel genetik programlama olan çoklu ifade genetik programlama açıklanacaktır.

3.1 Öncelik Kuralları

Üretim sistemlerinde çizelgeleme problemine kısa zamanda iyi çözümler vermesi amacıyla tasarlanmış olan öncelik kuralları (priority rules), ilgili problemlere yaygın olarak uygulanmıştır. Literatürde öncelik kuralı terimi çizelgeleme kuralı

(scheduling rule), sıralama kuralı (sequencing rule) ve yönlendirme kuralı (dispatching rule) olarak da isimlendirilmiş (Panwalker and Iskander, 1977; Blackstone, et al., 1982; Baker, 1974) ve birçok farklı kural türetilmiştir (Panwalker and Iskander, 1977; Pinedo, 1995).

Bir öncelik kuralı tezgah kuyruğunda işlem görmek için bekleyen işlerden hangisinin önce işleme alınması gerektiğini belirleyen bir kuraldır. Öncelik kuralı işleme alınacak olan işin seçimi sırasında çeşitli bilgileri (tezgah bilgileri, iş bilgileri, sistem bilgileri vb.) kullanarak her bir işe bir öncelik değeri atayan bir fonksiyon kullanmaktadır. Öncelik değeri atama sürecinin ardından en öncelikli iş tezgahta işlem görmek için seçilmektedir.

Öncelik kuralları çeşitli şekillerde sınıflandırılmaktadır (Pinedo, 1995; French, 1982). İlk sınıflandırma şeklinde öncelik kuralları statik ve dinamik olarak ikiye ayrılırlar. Statik kurallar zamana bağlı olarak işlemeyen kurallardır. Bu tür kurallar sadece iş ve/veya tezgah bilgilerine bağlı olarak çalışır. Bu tür kurallara SPT (En Küçük İşlem Süresi – Shortest Processing Time) örnek olarak verilebilir. SPT kuralı tezgahın kuyruğunda bekleyen işler arasından o tezgahtaki işlem süresi en küçük olan işi diğerlerinden önce işleme almaktadır. Dinamik kurallar ise zamana bağlı olan kurallardır. Bu kurallara örnek olarak ise MST (En Küçük Bolluk – Minimum Slack Time) kuralı verilebilir. Bu kural zaman içinde bir işi en önce işleme alabileceği gibi, aynı işi en son işleme de alabilir. Söz konusu kuralda parçalar işlenip zaman ilerledikçe işlerin öncelikleri değişiklik gösterebilir.

Öncelik kurallarının sınıflandırılmasında kullanılan ikinci bir yöntem ise, kuralları kullandığı bilgiye göre sınıflandırmaktır. Bu sınıflandırmaya göre kurallar yerel (local) ve bütünsel (global) olmak üzere iki kısma ayrılırlar (Pinedo, 1995). Yerel kurallar önceliği belirlenecek olan işin yer aldığı kuyruktan ya da ilgili kuyruğun tezgahından gelen bilgileri kullanırlar. Birçok öncelik kuralı bu sınıfa girmektedir. Bütünsel kurallar ise diğer tezgahlardan gelen bilgilere de bakarlar. Örneğin bir işin bir sonra ziyaret edeceği tezgahtaki işlem süresine bakan bir kural bütünsel bir kuraldır.

Öncelik kuralları çok basit olabileceği gibi birçok bilgiyi bünyesinde barındıran komplike öncelik kuralları da oluşturulmuştur. Komplike öncelik kuralları genel olarak

birçok parametrenin ağırlıklandırılması ile belirlenen değerleri işlere öncelik değeri olarak atamaktadır. Bahsedilen komplike öncelik kurallarına bir örnek aşağıda verilmiştir. Burada Z_j, j. işin ilgili tezgahta işlenme önceliği değeridir.

$$Z_j = w_1 \cdot DueDate + w_2 \cdot PTime + w_3 \cdot Slack$$

Komplike öncelik kurallarının içerisinde barındırdığı *basit* öncelik kuralının davranışına bir ölçüde sahip olduğu kabul edilmektedir. Fakat bu tür öncelik kurallarının en önemli sorunları arasında ağırlıkların (w_1, w_2 vb.) ve formülasyonunun belirlenmesi ile genelde uygulandığı probleme özgü olması belirtilebilir (Pinedo, 1995).

İzleyen kısımda genel olarak kullanılan bazı öncelik kuralları ve açıklamaları verilmektedir. Bu açıklamalar sırasında kullanılan Z_{ij}, i işinin j. tezgahındaki öncelik değerini belirtmektedir ve her öncelik kuralında farklı tanımlanmaktadır.

FIFO (First in First Out – İlk Giren İlk Çıkar): Bu öncelik kuralı ilgili tezgahta işlenecek olan işi, işlerin üretim sistemine geliş zamanlarına göre seçer. Sisteme geliş zamanı en erken olan iş tezgahta işlenecek olan ilk iştir. İlgili tezgahın kuyruğundaki işlere öncelik değeri ataması sırasında kullanılan denklem aşağıdaki gibidir.

$$Z_{ij} = r_i$$

Burada r_i i işinin sisteme geliş zamanıdır.

EDD (Earliest Due Date - En Erken Teslim Zamanı): Bu öncelik kuralı ilgili tezgahta işlenecek olan işi, işlerin teslim zamanlarına bağlı olarak seçer. Teslim zamanı en erken olan iş tezgahta işlenecek olan ilk iştir. İlgili tezgahın kuyruğundaki işlere öncelik değeri ataması sırasında kullanılan denklem aşağıdaki gibidir.

$$Z_{ij} = d_i$$

Burada d_i i işinin teslim zamanıdır.

MST (Minimum Slack Time - En Küçük Bolluk): Bu öncelik kuralı ilgili tezgahta işlenecek olan işi, işlerin bolluklarına göre seçer. Bolluğu en küçük olan iş tezgahta

işlenecek olan ilk iştir. İlgili tezgahın kuyruğundaki işlere öncelik değeri ataması sırasında kullanılan denklem aşağıdaki gibidir.

$$Z_{ij} = s_i = d_i - rt_i - t$$

Burada s_i i işinin bolluğu olmak üzere, rt_i i işinin kalan işlem süresi ve t ilgili andaki süredir.

SPT (Shortest Processing Time - En Küçük İşlem Süresi): Bu öncelik kuralı ilgili tezgahta işlenecek olan işi, işlerin o tezgahtaki işlem sürelerine göre seçer. İlgili tezgahtaki işlem süresi en küçük olan iş tezgahta işlenecek olan ilk iştir. İlgili tezgahın kuyruğundaki işlere öncelik değeri ataması sırasında kullanılan denklem aşağıdaki gibidir.

$$Z_{ij} = p_{ij}$$

Burada p_{ij} i işinin j tezgahındaki işlem süresidir.

CR (Critical Ratio - Kritik Oran): Bu öncelik kuralı ilgili tezgahta işlenecek olan işi, işlerin kritik oranlarına göre seçer. En küçük kritik orana sahip olan iş ilgili tezgahta işlem görecek olan ilk iştir. İlgili tezgahın kuyruğundaki işlere öncelik değeri ataması sırasında kullanılan denklem aşağıdaki gibidir.

$$Z_{ij} = \frac{d_i - t}{rt_i}$$

MON (Montagne Kuralı): Bu öncelik kuralı ilgili tezgahta işlenecek olan işi, izleyen denkleme bağlı olarak seçer. En küçük kritik orana sahip olan iş ilgili tezgahta işlem görecek olan ilk iştir.

$$Z_{ij} = \frac{p_{ij}}{p_{ij} - d_i}$$

Öncelik kurallarının üretim sistemi çizelgeleme problemlerine uygulanması ile ilgili ilk çalışmalar; Jackson (1955), Smith (1956), Giffler ve Thompson (1960) ve Gere

(1966) tarafından gerçekleştirilmiştir. Bu çalışmalar öncelik kurallarının genel esasları ve önemleri hakkında yapılmıştır. Öncelik kuralları ile ilgili en çok bilinen ve en kapsamlı çalışma Panwalker ve Iskander (1977) tarafından yapılmıştır. Bu çalışmada 113 adet öncelik kuralı açıklanmış, incelenmiş ve sınıflandırılmıştır. Daha sonraki yıllarda bu öncelik kuralları da dâhil olmak üzere birçok öncelik kuralı Blackstone et al. (1982), Haupt (1989) ve Bhaskaran ve Pinedo (1991) tarafından ayrıntılı olarak incelenmiştir. Bu çalışmalar sonucu oluşan genel kanı, bir öncelik kuralının diğerlerine tek başına üstünlük sağlamadığıdır. Chang ve diğerleri (1996) ise bir atölye tipi üretim sisteminde çizelgeleme probleminin doğrusal programlama modeli ile çözülmesi sonucu elde edilen sonuçları kullanarak 42 adet öncelik kuralının performansları üzerine çalışma yapmıştır. Bu çalışmadaki analizler, SPT tabanlı öncelik kurallarının genel olarak iyi, LPT (En Uzun İşlem süresi) tabanlı öncelik kurallarının ise genel olarak kötü performans sergilediğini ortaya koymuştur.

Basit öncelik kurallarının bütün problem türlerinde tek başına kullanılmasının genel olarak düşük performans sergilediği görüldüğü için, daha komplike öncelik kurallarının oluşturulması yoluna gidilmiştir (Pinedo, 1995). Komplike öncelik kurallarının oluşturulmasında kullanılan en yaygın yöntem öncelik kurallarının belli ağırlıklar kullanılarak birleştirilmesidir. Yani öncelik kuralları belli katsayılarla ağırlıklandırılmış ve yeni komplike öncelik kuralları oluşturulmuştur. Bu stratejinin kullanıldığı ilk çalışmalar, Crowston ve diğerleri (1963) ile Fisher ve Thompson (1963)'ın çalışmalarıdır. Lawrence (1984) ise çalışmasında 10 adet öncelik kuralının performanslarını, bu öncelik kurallarının rastgele birleştirilmesi ile oluşturulmuş komplike öncelik kuralları ile karşılaştırmıştır. Çalışması sonucunda komplike öncelik kurallarının basit öncelik kurallarına göre daha iyi bir performans sergilediğini fakat komplike öncelik kurallarının çözümü bulması için daha fazla zamana ihtiyaç duyabileceğini göstermiştir. Öncelik kurallarının kullanıldığı daha farklı çalışmalar arasında ise Dorndorf ve Pesch (1995)'in her bir tezgahta hangi öncelik kuralının uygulanması gerektiğini belirlemek için genetik algoritma tekniğini kullandıkları çalışma ile Grabot ve Geneste (1994)'in aynı amaç için bulanık mantık tekniğini kullandıkları çalışmalar sayılabilir. Son 30 yıl boyunca, benzetim teknikleri ile (Montazer and Van Wassenhove, 1990) öncelik kurallarının çoğunun performansı

üzerine çalışılmıştır. Bu çalışmalar "Bir performans ölçütünü eniyilemek için hangi öncelik kuralını seçmeliyim?" sorusunun cevabını aramaktadır. Bu amaçla yapılan birçok çalışma olmasına karşın hangi durumlarda hangi öncelik kuralının kullanılması gerektiği belirlenememiştir. Fakat performans ölçütüne bağlı olarak hangi öncelik kuralının seçileceği problemi üzerinde çalışılan araştırma alanı hala oldukça aktiftir (Jones and Rabelo, 1998).

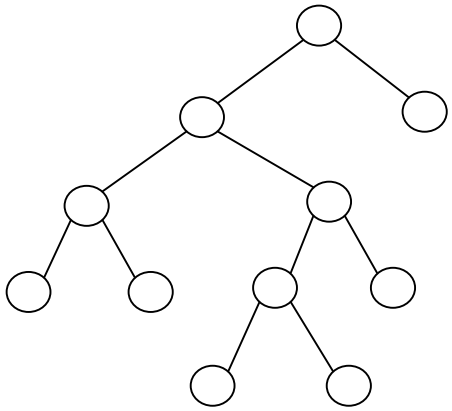
3.2 Standart Genetik Programlama

Genetik programlama, genetik algoritma tekniğinin bir uzantısı olarak geliştirilmiştir. Genetik algoritma, belirli sayıda aday çözüm üzerinde seçim, çaprazlama ve mutasyon operatörlerini kullanarak daha iyi çözümler elde etmeye çalışan bir sezgisel arama tekniğidir. İlk genetik programlama uygulaması 1992 yılında John Koza (Koza, 1992) tarafından genetik algoritma kullanılarak çeşitli görevleri yerine getiren bilgisayar programlarının geliştirilmesi ile ortaya çıkmıştır. Genetik programlama ile genetik algoritmaların arasındaki temel farklılık, çözüm yapısının gösterimi (kromozom) ve bu gösterimin taşıdığı anlamda ortaya çıkmaktadır. Genetik algoritmalar genelde sabit uzunluktaki dizilere sahip çözümler üzerinde çalışır. Standart genetik programlama ise genellikle bilgisayar programlarını ya da bilgisayar programı şeklinde gösterilebilecek algoritmaları barındıran ağaç yapısına sahip çözümler üzerinde çalışır. Genetik programlama, problemin yüksek seviyeli bir ifadesini kullanarak otomatik olarak çalışan bir bilgisayar programı oluşturulmasını sağlayan bir metottur ve bir problemi çözmek ya da çözümüne yaklaşmak amacıyla önceden boyut ve şeklinin belirlenmesine ihtiyaç duyulmayan her türlü veri / bilgi yapısını ve içeriğini bulmak için bir yol sağlar. Şekil 3.1'de standart genetik programlama (ağaç yapılı) ve genetik algoritmada (ikili kodlamalı) yaygın olarak kullanılan iki çözüm gösterimi verilmiştir.

a)

1	0	0	1	0	1	1	1	0	1	1	0	0	0	1	1	1	0	1

b)



Şekil 3.1: a) Genetik algoritma gösterimi b) Genetik programlama gösterimi

Hem genetik algoritma hem de genetik programlama teknikleri mevcut çözümleri geliştirmek ve yeni çözümleri aramak amacıyla çaprazlama ve mutasyon gibi operatörler kullanmaktadırlar. Genetik programlamada çaprazlama ve mutasyon operatörleri ağaç yapısına uygun olarak yapılandırılmışlardır. Çaprazlama operatörü ile iki adet çözüm üzerinde rastgele iki adet düğüm seçilerek çözümler arasında bilgi aktarımı gerçekleştirilir. Mutasyon operatörüyle ise rastgele bir düğüm seçilip değiştirilerek yeni çözümler bulma işlemi gerçekleştirilir. Dolayısıyla genetik algoritmada çaprazlama ve mutasyon işlemleri ile genler seçilirken, genetik programlamada genler yerine düğümler seçilmektedir. Genetik programlamadaki seçim operatörünün çalışma mantığı ise genetik algoritmadakinden farklı değildir ve daha iyi çözümleri sonraki iterasyonlara taşıma işlemini yerine getirmektedir.

Daha sonraki yıllarda birçok genetik programlama çeşidi (gen denklem programlama – GEP, doğrusal genetik programlama – LGP, çoklu ifade programlama – MEP vb.) geliştirilmiştir. Bunlar genel olarak kromozom yapısı ağaç (tree) şeklinde olanlar ve doğrusal (linear) şekilde olanlar olmak üzere ikiye ayrılabilir. Doğrusal

yapılı kromozomların gösterimi ve geçerliliğinin tespiti ağaç yapılı kromozoma göre daha az maliyetli olmaktadır.

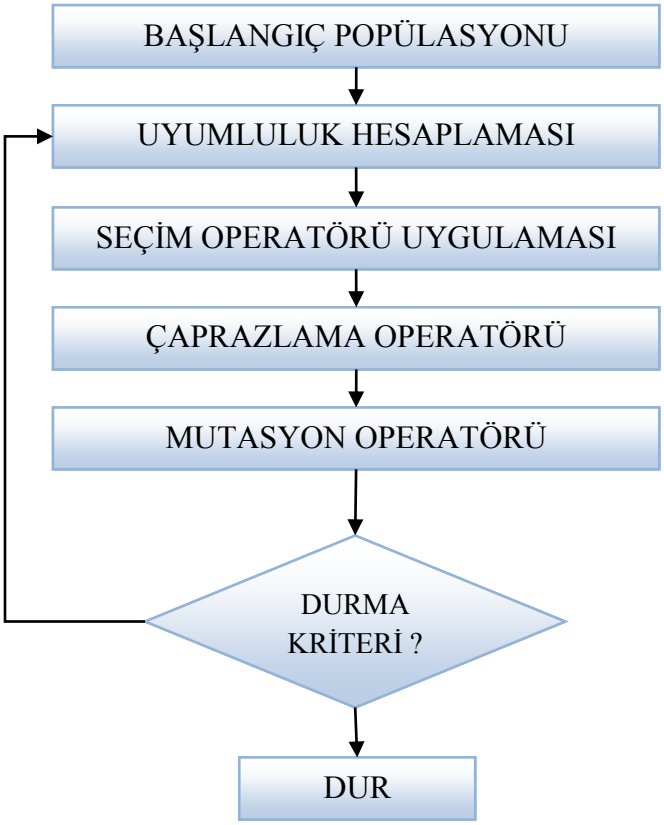
Genel olarak ağaç yapılı kodlamaya sahip standart genetik programlama ile doğrusal kodlamaya sahip genetik programlama arasında iki ana fark bulunmaktadır. Birincisi, doğrusal genetik programlamada kullanılan kromozomda barınan bilgiler kromozomun yapısı gereği tekrarlı kullanıma uygun durumdadır. Standart genetik programlamada ise kromozom ağaç yapısı şeklinde olduğundan, aynı bilgilerin tekrarlı kullanılabilmesi sadece bilginin aynı şekilde tekrardan oluşturulabilmesi ile mümkündür. Bu da daha önceden oluşturulmuş bir bilginin ya da bilgi yapısının aynı şekilde başka yerde kullanımını oldukça zorlaştırmaktadır. İkinci fark ise doğrusal genetik programlama kromozomunda bulunan kullanılmayan bilgilerin kolayca belirlenebilmesi ve giderilebilmesidir. Standart genetik programlamada bütün bilgiler tek bir bütünsel yapı şeklinde tutulup kök düğüme bağlı olduğundan gereksiz ve kullanılmayan bilgilerin tespiti ekstra çabalarla yapılabilmektedir.

3.2.1 Standart genetik programlama akışı

Standart genetik programlama; seçim (selection), çaprazlama (crossover) ve mutasyon (mutation) gibi genetik operatörleri ağaç yapısı şeklindeki çözümlere uygulayarak popülasyondaki (çözümler kümesi) çözümleri uyumluluk değerini eniyileyecek şekilde geliştirmeye çalışan bir algoritmadır (Koza, 1992; Koza, 1994).

Genetik programlama işlemlere çözüm kümesini rastgele üreterek başlar. Bu rastgele çözüm kümesini ise problemin terminal ve fonksiyon kümelerini kullanarak oluşturur. Çözüm kümesi içerisindeki çözümler (kromozom - birey) rastgele oluşturulduğundan farklı şekillerdedir. Bu işlemin ardından durma kriteri sağlanana kadar tekrarlı şekilde devam edecek işlemler gelir. Bunlar uyumluluk değerlendirmesi, seçim, çaprazlama ve mutasyondur. Popülasyon içerisindeki her bir bireyin eldeki problemi ne kadar iyi çözebildiğini bulmak amacıyla uyumluluk değerlendirmesi yapılır. Bireyler daha sonra uyumluluk değerlerine göre seçilerek çeşitli genetik

operasyonlara sokulmaktadır. Bazı durumlarda diğerlerinden daha az uygun bireylerin seçilme durumu da matematiksel olarak imkân dâhilindedir. Seçilen bireyler üzerinde çaprazlama ve mutasyon operatörleri belirli olasılıklarla uygulanarak bireyler bir sonraki popülasyona aktarılır. Yeni popülasyonun bu şekilde oluşturulduktan ve bir iterasyon tamamlandıktan sonra işlemler tekrar ettirilir. Birçok iterasyondan sonra, bir ya da daha fazla çözüm ortaya çıkartılır ve genetik programlama işlemleri bitirilir. Şekil 3.2'de genetik programlama çalışması esnasındaki adımlar gösterilmektedir.



Şekil 3.2: Standart genetik programlama akış diyagramı

Bu genetik programlama adımları daha ayrıntılı olarak ilerleyen kısımlarda incelenecektir.

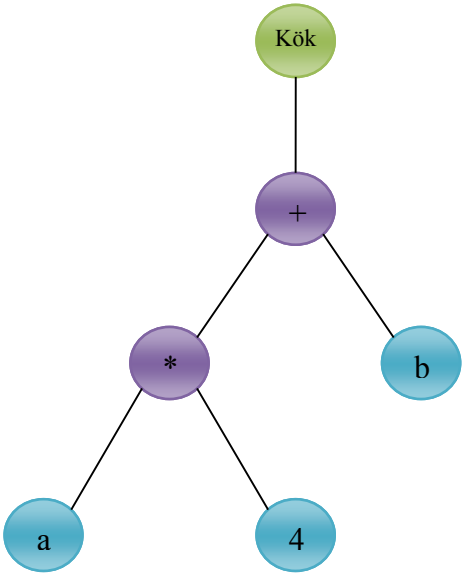
3.2.2 Standart genetik programlama bileşenleri

Standart bir genetik programlama algoritmasının çalışması için kullanılan bileşenler aşağıda açıklanmıştır. Bu bileşenlerin her biri problemin yapısına bağlı olarak her zaman kullanılmayabilir.

3.2.2.1 Fonksiyon ve terminal kümeleri

Terminal kümesinde bulunan terminal öğeleri problemin tanımlanmasında doğrudan etkili olan değişken, parametre ve sabitlerdir. Her problemin kendine has terminal yapısı vardır ve terminallerin seçiminde kullanılan kesin kurallar bulunmamaktadır. Fonksiyon kümesindeki öğeler ise terminal öğelerini belli bir işlem mantığına göre birleştirmeye yararlar. Bazı problemlerde ilgili fonksiyonlar yalnızca toplama, çıkarma, çarpma ve bölme gibi aritmetik operatörlerden oluşabilir. Diğer problemlerde ise çok daha karmaşık operatörler kullanılabilir. Terminal ve fonksiyon kümelerinin tanımlanması genetik programlamanın çalışmasından önce probleme özel olarak yapılması gereken bir iştir.

Fonksiyonların ve terminallerin hiyerarşik bir biçimde anlamlı bir yapı oluşturulacak şekilde birleştirilmesi sonucu çözümleri kodlayan kromozomlar oluşturulur. Standart genetik programlamada kullanılan kromozomların yapısı bir kökten çıkan (root) ve daha sonra dallanan bir ağaca (tree) benzer. Terminal öğeleri ağaç yapısının dallanmalarının sonlandığı noktalara yerleştirilir. Fonksiyonlar ise terminaller ile kök arasındaki noktalarda yer alırlar. Şekil 3.3'de matematiksel fonksiyon ve terminallerden oluşturulmuş bir kromozomun ağaç yapısı gösterilmektedir ve 4*a+b'yi kodlamaktadır.



Şekil 3.3: Bir bireyin ağaç yapısı (Fonksiyonlar: + ve *; Terminaller: a, b ve 4)

Daha farklı problem türlerinde özelleştirilmiş fonksiyon ve terminaller kullanılabilir. Örneğin, problem bir robotun belli bir alanı en az adımla dolaşması olsaydı, genetik programlama içerisinde ileri gitmek, 90° dönmek, 180° dönmek gibi hareketler tanımlanabilirdi.

3.2.2.2 Uyumluluk fonksiyonu

Popülasyondaki çözümlerin ele alınan problemi ne kadar iyi çözebildikleri uyumluluk fonksiyonu ile hesaplanır. Elde edilen uyumluluk değerine göre aday çözümlerin izleyen nesillere aktarılma olasılıkları artar ya da azalır. Bir aday çözümün uyumluluk değeri ne kadar iyiyse sonraki nesillere aktarılma olasılığı o kadar artmaktadır.

Genetik programlamanın arama uzayında yön tayini yapmasını sağlayan tek mekanizma uyumluluk değeridir. Bu nedenle genetik programlamanın çözümünü

arayacağı probleme uygun bir uyumluluk fonksiyonu seçilmesi gerekmektedir. Literatürde genel olarak aşağıdaki uyumluluk fonksiyonları tek başına ya da birleştirilmiş şekilde kullanılmaktadır.

Fayda Tabanlı (Benefit-based): Uyumluluk fonksiyonunun alınan faydayla doğru orantılı olmasıdır.

Hata Tabanlı (Error-based): Uyumluluk fonksiyonunun toplam hataya göre ters orantılı olmasıdır.

Maliyet Tabanlı (Cost-based): Uyumluluk fonksiyonunun kullanılan kaynakla (zaman, yer, para, ağacın yaprak sayısı vb.) ters orantılı olmasıdır.

Cimrilik Tabanlı (Parsimony-based): Uyumluluk fonksiyonunun bireylerin sadeliği ile doğru orantılı olmasıdır.

3.2.2.3 Kontrol parametreleri

Genetik programlamanın çalışması sırasında kullanılan parametrelerdir. Bu parametrelerin doğru olarak belirlenmesi iyi çözümlere yakınsama miktarını ve hızını değiştirmektedir. Bu parametrelerin değerleri daha önce yapılmış doğruluğu ispatlanmış deneylerden ya da deneme-yanılma yöntemi ile bulunabilir.

Popülasyon büyüklüğü: Genetik programlamada her bir nesilde oluşturulacak olan birey (kromozom) sayısıdır.

Çaprazlama olasılığı: Çaprazlama operatörünün aday çözümler üzerinde kullanılma olasılığıdır.

Mutasyon olasılığı: Mutasyon operatörünün aday çözümler üzerinde kullanılma olasılığıdır.

Birey seçim yöntemi: Bir sonraki nesilde yaşamlarını sürdürecek bireylerin belirlenmesi işleminde kullanılacak olan birey seçme yöntemini ifade eder. Bu yöntemler turnuva seçimi, rulet seçimi ve sıralama seçimi gibi yöntemlerdir.

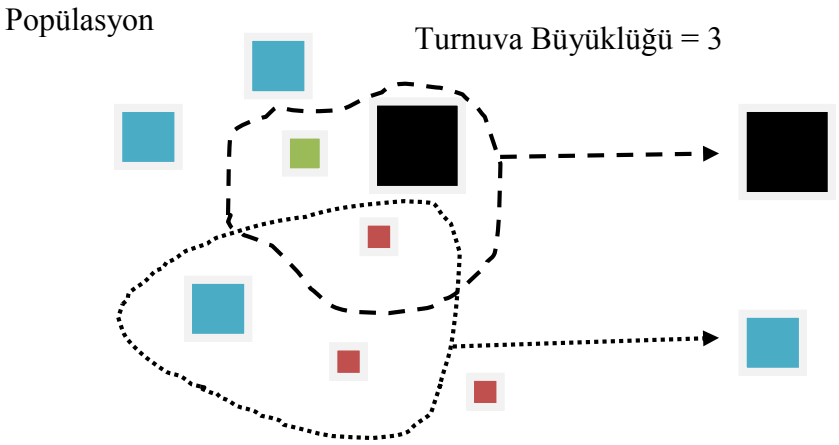
Maksimum iterasyon/nesil sayısı: Genel olarak uygulanan bir durma kriteri olan maksimum nesil (jenerasyon) sayısıdır.

3.2.2.4 Seçim (Selection) operatörü

Bir sonraki nesile aktarılacak olan kromozomların seçimini yapan operatördür. Bu kromozomlar üzerinde bir sonraki nesile aktarılmadan önce çaprazlama ve mutasyon işlemleri de olasılıklarına göre yapılabilmektedir. Literatürde birçok seçim operatörü mevcuttur ve her birinin kendine has avantaj ve dezavantajları vardır (Hancock, 1994). Aşağıda en çok kullanılan seçim yöntemlerinden olan turnuva seçimi ve rulet seçimi teknikleri açıklanmıştır (Maza and Tidor, 1991).

Turnuva Seçimi (Tournament Selection)

Turnuva seçim tekniğinde öncelikle popülasyon içerisinden rastgele seçilen kromozomlar ile bir alt küme oluşturulur ve alt kümedeki en iyi uyumluluk değerine sahip kromozom bir sonraki nesile yerleştirilmek amacıyla seçilir. Bu alt kümenin büyüklüğü popülasyonun büyüklüğüne göre belirlenir. Örneğin 100 bireye sahip bir popülasyonda turnuva seçim büyüklüğü için 3 yeterli iken birey sayısı 5000 olan bir popülasyon için yeterli olmayacaktır. Bu işlem bir sonraki nesil için seçilen bireylerin sayısı popülasyon sayısına eşit olana kadar devam eder. Şekil 3.4'de turnuva seçimi yönteminin işleyişi görülmektedir.

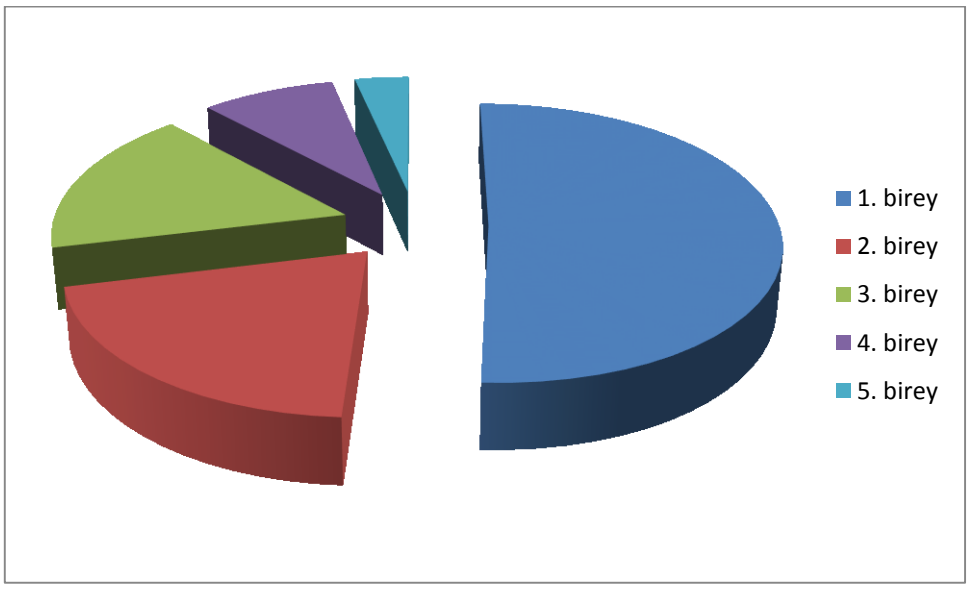


Şekil 3.4: Turnuva seçiminin işleyişi (Maza and Tidor, 1991)

Rulet Seçimi (Fitness-Proportionate Selection)

Uyumluluk oranı seçim tekniği olarak da bilinir. Bu seçim tekniğinde uyumluluk değeri en yüksek olan kromozomun seçilme sansı daha yüksektir. Rulet seçimi tekniğinin dezavantajı, her seçimde popülasyonda bulunan en iyiyi seçme olasılığı yüksek olduğundan bölgesel maksimumlara takılma olasılığının diğer tekniklere göre daha yüksek olmasıdır. Fakat genetik programlama problem için aranan çözümü en iyi birey ile en kötü bireyin çaprazlamasında bulacak ise rulet seçim tekniğinde bu çok zor olacaktır.

Şekil 3.5'de her bir bireyin seçilme olasılığının uyumluluk değerleri ile orantılı olduğu rulet seçim tekniği gösterimi verilmektedir. Burada uyumluluk değeri daha yüksek olan bireylerin seçilme olasılıklarının daha yüksek olduğu görülmektedir.



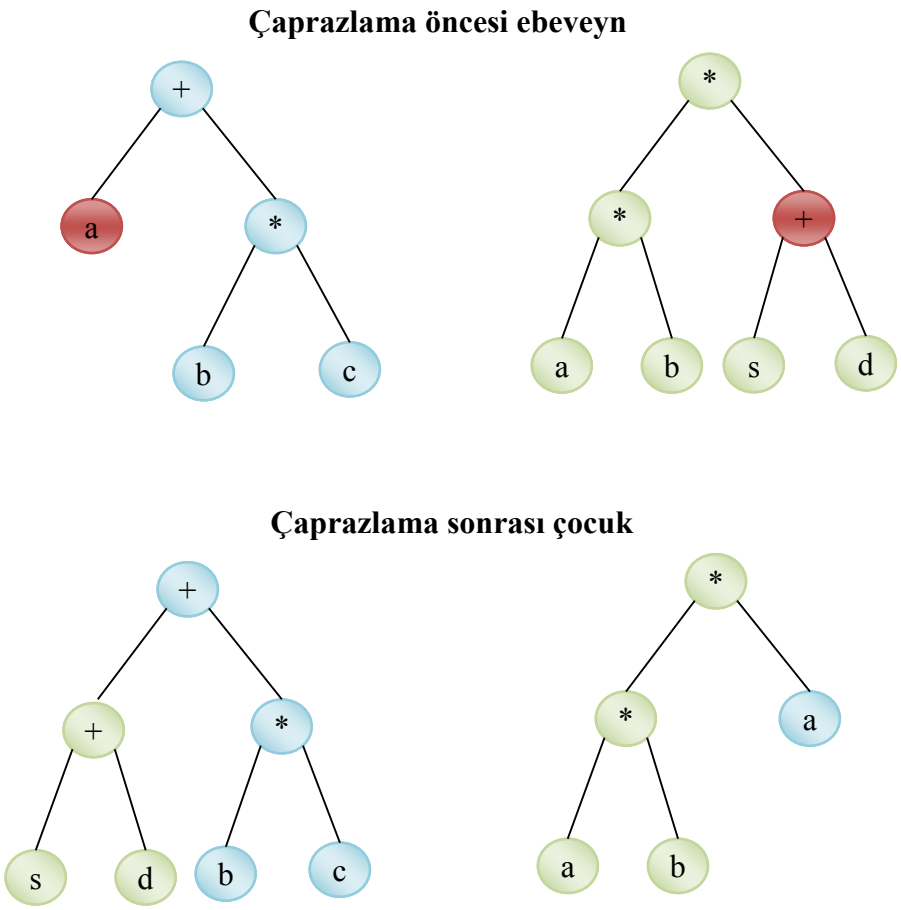
Şekil 3.5: Rulet seçim tekniği işleyişi

Bu seçim yöntemlerine ek olarak genetik programlamada bazen elitizm de kullanılır. Elitizm popülasyondaki en iyi uyumluluk değerine sahip birey(ler)in doğrudan yeni nesile (çaprazlama ve mutasyon olmadan) aktarılmasıdır. Uygulanması durumunda genellikle tek bir çözüm elit çözüm olarak diğer nesile aktarılır.

3.2.2.5 Çaprazlama (Crossover) operatörü

Çaprazlama operatörü kromozomlar arasında bilgi alışverişine imkan veren ve bu yolla yeni çözümlerin oluşturulmasını sağlayan bir operatördür. Seçim operatörü ile oluşturulan yeni popülasyondaki kromozomlar çaprazlama operatörü için ebeveyn adaylarıdır. Çaprazlama operatörü bu adaylardan bir kısmını eşleştirmek üzere seçer ve bunları eşleme havuzuna koyar. Bu seçim işleminde genetik programlama kontrol parametrelerinden birisi olan çaprazlama olasılığından (P_c) faydalanılır. Bu olasılık kullanıcı tarafından tanımlanır ve çaprazlama operatörüne tabi tutulacak beklenen kromozom sayısını (P_c*n) verir. Eşleme havuzuna seçilen ebeveynler eşleştirilip

çaprazlamaya tabi tutulurlar ve sonuç olarak yeni çözümleri kodlayan çocuk kromozomlar elde edilir. Bu çocuk kromozomlar popülasyonda ebeveynlerinin yerine konulurlar. Şekil 3.6'de çaprazlama operatörünün örnek bir işlemi gösterilmektedir. Ebeveyn kromozomlar üzerindeki çaprazlama noktaları kırmızı ile işaretlenmiştir.

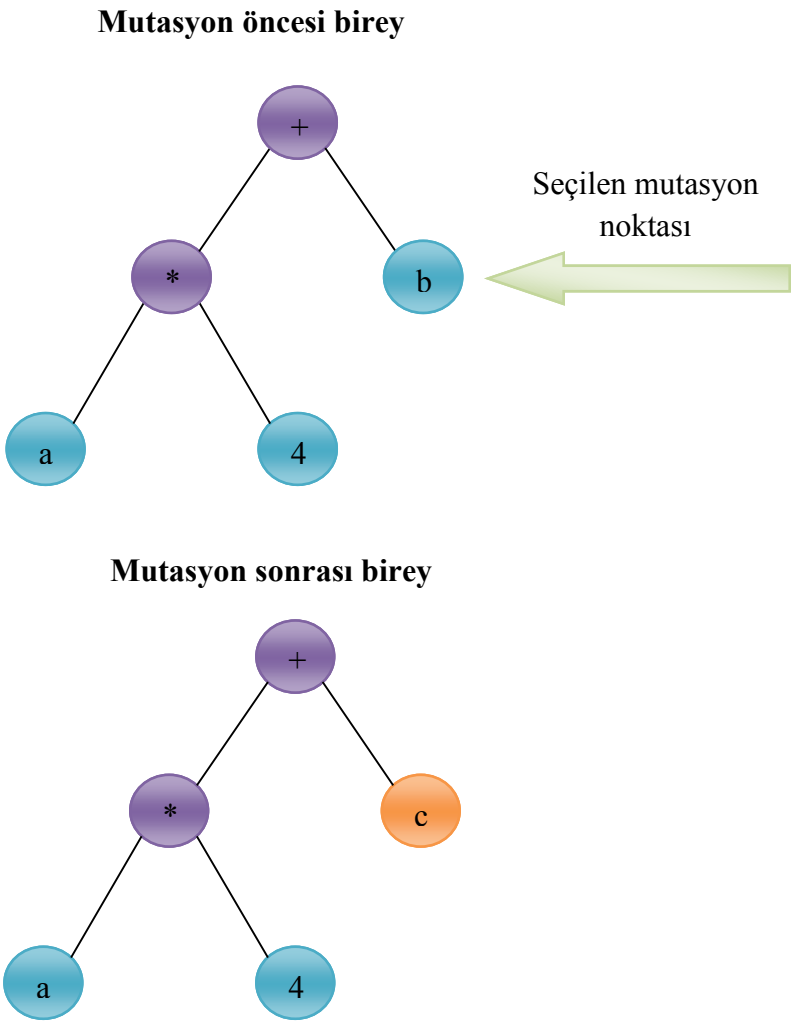


Şekil 3.6: Çaprazlama operatörü örnek işlemi

3.2.2.6 Mutasyon (Mutation) operatörü

Mutasyon operatörü genetik programlama sürecinin son safhasını oluşturur. Seçim ve çaprazlama operatörlerinden sonra yeni kromozomlardan oluşan yeni bir

popülasyon elde edilir. Genetik programlama işleyişinde sadece seçim ve çaprazlama operatörleri kullanılırsa birkaç iterasyon sonra popülasyondaki tüm kromozomlar birbirine benzemeye başlarlar. Bunun nedeni ise popülasyondaki kromozomların içerdiği bilgilerin hep aynı kalmasıdır. Eğer popülasyondaki tüm kromozomlarda belirli bir pozisyondaki genin değeri aynı ise, seçim ve çaprazlama operatörleri yoluyla o pozisyondaki genin değeri farklı olan bir çocuk kromozom oluşturmak mümkün değildir. Bu da çözüm arama işlemi esnasında tüm çözüm uzayının taranmasını engeller. Mutasyon operatörü bu eksiği gidermek amacıyla devreye girmektedir. Mutasyon operatörü çözüm arama işleminin incelenmemiş bölgelere yayılmasını sağlayan bir operatördür. Ayrıca genetik programlamanın yerel en iyiye takılmasını önleme görevini de bu operatör üstlenmiştir. Mutasyon işlemini gerçekleştirmek için bir mutasyon olasılığına (P_m) ihtiyaç duyulur. Bu olasılık genetik programlama kontrol parametrelerinden birisi olup kullanıcı tarafından tanımlanır. Şekil 3.7'de mutasyon operatörünün örnek bir işlemi gösterilmektedir.



Şekil 3.7: Mutasyon operatörü örnek işlemi

3.2.2.7 Durma kriteri

Durma kriteri genetik programlama sürecinin hangi koşul ya da koşullarda durması gerektiğini belirleyen bir yapıdır. Durma kriteri maksimum nesil sayısı olabileceği gibi probleme özgü başarı değerleri ya da arama süresi de olabilir. Genel olarak uygulanan durma kriteri maksimum nesil sayısıdır.

3.3 Çoklu İfade Programlama

Çoklu İfade Programlama (Multi Expression Programming – MEP) ya da Çoklu Çözüm Programlama (Multi Solution Programming – MSP) 2002 yılında Oltean ve Dumitrescu tarafından geliştirilmiş bir doğrusal genetik programlama tekniğidir. Çoklu ifade programlama tekniğinin diğer genetik programlama tekniklerinden en büyük farkı çoklu çözümleri tek bir kromozom içerisinde kodlayabilmesidir. Ayrıca yapısı gereği standart ve özelleştirme gerekmeyen çaprazlama ve mutasyon operatörlerini kullanabilmektedir. Bu tekniğin bir diğer avantajı ise çaprazlama ve mutasyon operatörlerinin uygulanması sonucunda ortaya çıkan kromozomlar her zaman sözdizimsel olarak uygun yapıda olduğundan onarım fonksiyonuna ihtiyaç kalmamaktadır (Oltean, 2006, 2007).

Çoklu ifade gösterimi tekniğinin standart genetik programlama tekniğinden en büyük farkı çözümlerin kodlandığı kromozom gösterimidir. Çoklu ifade programlama kromozomları genetik algoritmalardaki kromozomlara benzer şekilde belli sayıda genden meydana gelmektedir ve barındırılan gen sayısı ilgili kromozomun uzunluğunu göstermektedir. Kromozom içerisindeki her bir gen, barındırdığı bilgiye bağlı olarak değişken uzunlukta olabilmektedir. Kromozom üzerindeki her bir genin ilk karakteri bir terminal ya da fonksiyon kümesi öğesi olmalıdır. Genin ilk karakterinin terminal öğesi içermesi durumunda gende başka bir bilgi bulunmasına gerek yoktur. Genin ilk karakterinin fonksiyon kümesinden bir öğe içermesi halinde ise fonksiyonun aldığı argüman sayısı kadar diğer genlere referans bulunmalıdır. Dolayısıyla fonksiyon argümanları diğer genlerin içerdiği bilgiler olmaktadır. Örnek olarak, eğer genin ilk karakteri fonksiyon kümesi öğelerinden çarpma fonksiyonu ise ilgili gende toplamda 3 karakterlik bilgi bulunmalıdır. İlk karakter çarpma fonksiyonu olmak üzere ikinci ve üçüncü karakterler diğer genlere birer referans içermelidir. Dolayısıyla bu çarpma fonksiyonu, ikinci ve üçüncü karakterlerdeki gen referansları ile gösterilen genlerdeki değerleri çarpmak için kullanılacaktır. Çoklu ifade programlama kromozomlarının oluşturulmasındaki kısıtlar ilk genin her zaman terminal öğesi içermesi ve bir gen içerisinde diğer genlere referans verilecekse referansların önceki indisli genlere olmasıdır.

Şekil 3.8'de bir çoklu ifade programlama kromozomuna örnek verilmiştir. Kromozomun oluşturulmasında T = {a, b, c, d} terminalleri ve F = {+, *} fonksiyonları kullanılmıştır.

	Kromozom		
Gen 1	a		
Gen 2	b		
Gen 3	+	1	2
Gen 4	c		
Gen 5	d		
Gen 6	+	4	2
Gen 7	*	3	4
Gen 8	*	2	5

Şekil 3.8: Örnek bir MEP kromozomu

MEP kromozomları her zaman baştan sona doğru okunan bir yapı ile oluşturulurlar. Örnekte 1, 2, 4 ve 5 numaralı genlerde terminal öğeleri (değişken) bulunmaktadır. 3, 6, 7 ve 8 numaralı genlerde fonksiyonlar bulunmaktadır ve fonksiyonların argümanları diğer genlere bir ya da daha fazla referans içermektedir. Örneğin 3. gende bulunan fonksiyon "**+ 1 2**" şeklinde belirtilmiştir. Bu fonksiyonun argümanları 1. ve 2. genlere referans içermektedir. Dolayısıyla ilgili fonksiyon gerçek şekliyle yazıldığında "a + b" şeklinde belirtilebilir. Şekil 3.7'de verilen kromozomun kodlanmış şekilde içerdiği bütün ifadeler Şekil 3.9'daki gibidir.

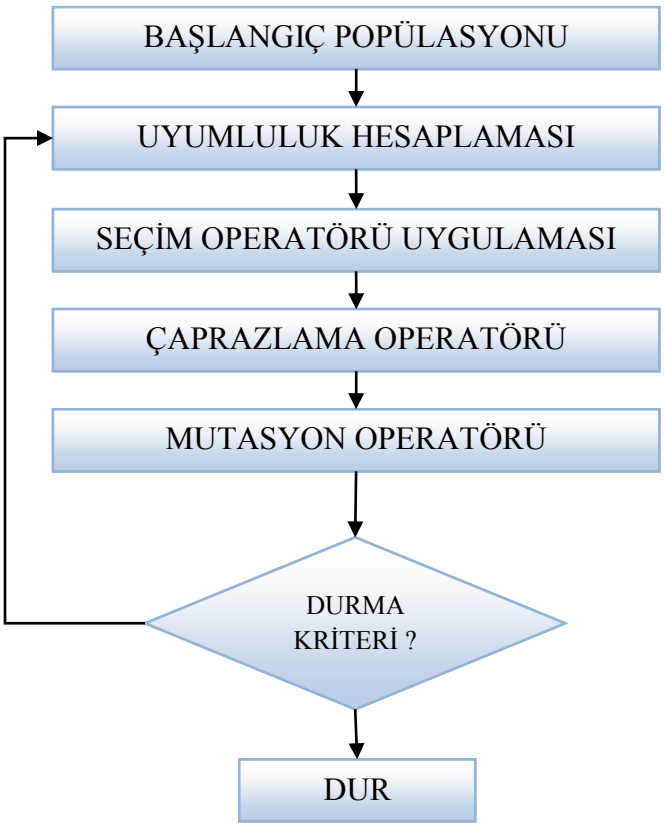
	İfade
Gen 1	a
Gen 2	b
Gen 3	a + b
Gen 4	c
Gen 5	d
Gen 6	c + b
Gen 7	(a + b) * c
Gen 8	b * d

Şekil 3.9: Örnek MEP kromozomunun içerdiği ifadeler

Her bir çoklu ifade programlama kromozomu gen sayısı kadar ifadeyi kodlanmış şekilde bünyesinde bulundurur. Her pozisyondaki ifade baştan sona doğru fonksiyonların gen referansları takip edilerek ortaya çıkartılır. Her bir çoklu ifade programlama kromozomunun uyumluluk değeri ise kromozomun barındırdığı ifadelerin uyumluluk değerlerinin en iyisidir.

3.3.1 Çoklu ifade programlama akışı

Çoklu ifade programlama tekniği standart genetik programlama ile aynı akış diyagramına sahiptir. Şekil 3.10'da çoklu ifade programlama tekniğinin çalışması esnasındaki adımlar gösterilmektedir.



Şekil 3.10: Çoklu ifade programlama akış diyagramı

Çoklu ifade programlama, problemin terminal ve fonksiyonlarından rastgele oluşan kromozomların (birey) popülasyonu ile başlar. MEP kromozomlarının oluşturulması standart genetik programlamaya göre daha kolaydır. Çünkü standart genetik programlama içerisinde kromozom oluşturulması ile ilgili birçok yöntem mevcuttur ve bu yöntemler genetik programlamanın çözüme yakınsama hızı üzerinde ciddi etkiye sahiptirler. Buna karşın çoklu ifade programlama içerisinde kromozom oluşturma işlemi sırasında tek parametre kromozomun gen sayısıdır. Çoklu ifade programlamada, genetik algoritma ve standart genetik programlama tekniklerinde kullanılabilecek bütün seçim operatörleri kullanılabilir. Çaprazlama ve mutasyon işlemleri içinse standart genetik programlama içerisindeki karmaşık operatörler yerine basit tek ya da çok nokta çaprazlama ile basit mutasyon operatörleri kullanılmaktadır.

Basit çaprazlama ve mutasyon operatörlerinin kullanımı ile fazladan işlem yapılmasına gerek kalmamaktadır.

3.3.2 Çoklu ifade programlama bileşenleri

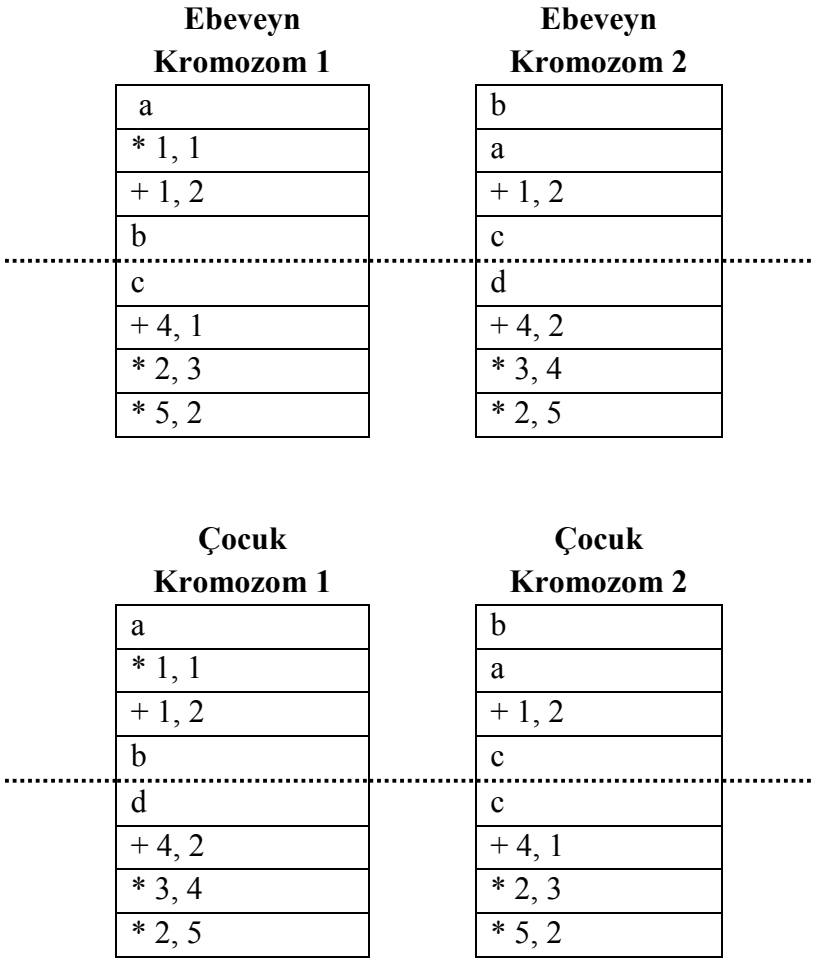
Çoklu ifade programlama tekniğinin çalışması için kullanılan bileşenler aşağıda açıklanmıştır. Bu bileşenlerin her biri problemin yapısına bağlı olarak her zaman kullanılmayabilir.

3.3.2.1 Seçim (Selection) operatörü

Bir sonraki nesile aktarılacak olan kromozomların seçimini yapan operatördür. Bu kromozomlar üzerinde bir sonraki nesile aktarılmadan önce çaprazlama ve mutasyon işlemleri de olasılıklarına göre yapılabilmektedir. Çoklu ifade programlamada kullanılabilir seçim operatörleri standart genetik programlama seçim operatörleri ile aynıdır.

3.3.2.2 Çaprazlama (Crossover) operatörü

Çaprazlama operatörü kromozomlar arasında bilgi alışverişine imkan veren ve bu yolla yeni çözümlerin oluşturulmasını sağlayan bir operatördür. Seçim operatörü ile oluşturulan yeni popülasyondaki kromozomlar çaprazlama operatörü için ebeveyn adaylarıdır. Çoklu ifade programlamada kullanılan çaprazlama operatörleri basit tek nokta ya da çok nokta çaprazlama operatörleridir. Tek nokta çaprazlamada ebeveyn kromozomlarda her birinde aynı pozisyonda bir çaprazlama noktası seçilir ve parçalar paylaşılarak çocuk kromozomlar oluşturulur. Tek nokta çaprazlama işlemine bir örnek çaprazlama noktası 5. Gen olmak üzere Şekil 3.11'de verilmiştir.



Şekil 3.11: Tek nokta çaprazlama örnek işlemi

3.3.2.3 Mutasyon (Mutation) operatörü

Mutasyon operatörü çoklu ifade programlama sürecinin son safhasını oluşturur ve çözüm arama işleminin incelenmemiş bölgelere yayılmasını sağlayan bir operatördür. Ayrıca çoklu ifade programlamanın yerel en iyiye takılmasını önleme görevini de bu operatör üstlenmiştir. Çoklu ifade programlamada basit mutasyon operatörü kullanılmaktadır. Basit mutasyon operatöründe kromozom üzerinde seçilen bir noktadaki bilgi başka bir bilgi ile değiştirilir. Basit mutasyon işlemine bir örnek

mutasyon noktası 5. gen olmak üzere Şekil 3.12'de verilmiştir ve mutasyon sonucu gen içerisindeki ifade "c" iken "d" olmuştur.

Mutasyon öncesi birey	**Mutasyon sonrası birey**
a	a
* 1, 1	* 1, 1
+ 1, 2	+ 1, 2
b	b
c	**d**
+ 4, 1	+ 4, 1
* 2, 3	* 2, 3
* 5, 2	* 5, 2

Şekil 3.12: Basit mutasyonun gen içeriği örnek işlemi

Gen içerisindeki bilgilerin değiştirilmesine ek olarak bir genin diğer genlere olan referansları üzerinde de mutasyon yapılabilmektedir. Gen referansının değiştirilmesini sağlayan bu mutasyon ile kromozomların barındırdığı ifadelerin içeriği yerine yapısı değiştirilmiş olmaktadır. Bu mutasyon işlemine bir örnek mutasyon noktası 3. genin 2. gen referansı olmak üzere Şekil 3.13'de verilmiştir ve mutasyon sonucu gen içerisindeki ifade "b+a" iken "b+a+a^2" olmuştur.

Mutasyon öncesi birey	**Mutasyon sonrası birey**
a	a
* 1, 1	* 1, 1
+ 1, 2	+ 1, 2
b	b
c	**c**
+ 4, 1	+ 4, **3**
* 2, 3	* 2, 3
* 5, 2	* 5, 2

Şekil 3.13: Basit mutasyonun gen referansı üzerindeki örnek işlemi

3.4 Yapılmış Olan Çalışmalar

Genetik programlama tekniğinin geliştirilmesinden günümüze üretim sistemlerinde çizelgeleme konusundaki uygulamaları çok az olmuştur. Son yıllarda geliştirilmiş olan çoklu ifade programlama tekniği ise üretim sistemlerinde çizelgeleme konusuna, bu doktora çalışması hariç, hiç uygulanmamıştır. Aşağıda üretim sistemlerinde kullanılabilecek çözümleri otomatik olarak oluşturan çalışmalar tarih sıralamasına göre verilmiştir.

- Nakasuka ve Yoshida (1992) tarafından deneysel veriler karar ağaçlarının otomatik oluşturulması için kullanılmıştır. İlgili veriler simülasyon sırasında karar ağaçlarının işin seçimi için kullanılacağı anda ulaşılabilecek verilerden türetilmektedir. Bu çalışmada mevcut öncelik kurallarının kullanılması yoluna gidilmiştir.
- Shaw, Park ve Raman (1992) tarafından simülasyonla birleşik öğrenme mekanizması kurulmuştur. Karar ağaçlarının yapılandırılmasında örüntüler kullanılmıştır. Bu örüntüler ise ID3 algoritması ve öncelik kuralları yardımıyla oluşturulmuştur.

- Piramuthu ve diğerleri (1993) ve Piramuthu, Raman ve Shaw (1994) tarafından bir önceki makalelerinde yaptıkları çalışmaları ID3 algoritmasını C4.5 algoritması ile değiştirerek yinelemişlerdir. Bu çalışmalarda önceki çalışmaya göre daha az karmaşık karar ağaçları oluşturulabilmiştir. Ayrıca bu çalışmada ilk kez öncül bilgiler kullanılarak yeni bilgilerin oluşturulması sağlanmıştır.
- Wang ve diğerleri (1995) tarafından öğrenme yeteneği olan bir çizelgeleme sistemi kurulmuştur. Bu çizelgeleme sisteminde karar ağaçları otomatik olarak yapay sinir ağlarına dönüştürülmektedir. Oluşturulan yapay sinir ağlarındaki bilgiler ardından karar kuralları oluşturulmasında kullanılmıştır. Karar kuralları ise çizelgeleme kurallarının oluşturulması sırasında kullanılmıştır.
- Chen ve Yih (1996) tarafından karar ağaçları ve öncelik kuralları için kullanılabilecek niteliklerin (attribute) seçimi ve yeni niteliklerin türetilmesi çalışması yapılmıştır.
- Lesh, Zaki ve Ogihara (1999) ve Deshpande ve Karypis (2002) tarafından basit niteliklerden daha karmaşık niteliklerin oluşturulması sağlanmıştır.
- Dimopoulos ve Zalzala (2001) tarafından tek tezgahlı üretim sistemleri için genetik programlama ile öncelik kuralı geliştirilmiştir. Bu öncelik kuralları geliştirilirken değişik teslim zamanları sıklıkları ile oluşturulmuş test ortamları ve gecikme temelli değerlendirme ölçütü kullanılmıştır.
- Olafsson (2003) tarafından veri madenciliği yöntemleri ve karar ağaçları üretim sistemlerinde öncelik kuralı oluşturmak için kullanılmıştır.
- Geiger, Uzsoy ve Aytug (2003) tarafından üretim sistemleri için genetik algoritma tekniği kullanılarak öncelik kuralları oluşturulmuştur.
- Geiger ve diğerleri (2006) tarafından genetik programlama tekniği hem statik hem de dinamik tek tezgah üretim sistemleri için öncelik kuralı oluşturulmasında kullanılmıştır. Oluşturulan öncelik kuralları EDD, SPT, MST, MDD, COVERT ve ATC öncelik kuralları ile karşılaştırılmıştır.
- Tay ve Ho (2008) tarafından genetik programlama ile atölye tipi üretim sistemlerinde birden fazla değerlendirme ölçütü için (ortalama gecikme, ortalama akış süresi ve işlerin tamamlanma süresi [makespan]) karmaşık öncelik kuralları oluşturulmuştur.

- Kapanoğlu ve Alikalfa (2004, 2005, 2006, 2008) tarafından genetik tabanlı makine öğrenmesi ve genetik algoritma ile atölye tipi üretim sistemlerinde gecikme tabanlı amaçlar göz önüne alınarak öncelik kuralları ve çizelgeleme politikaları oluşturulmuştur.
- Kapanoğlu ve Alikalfa (2010) tarafından çoklu ifade genetik programlama ile üretim sistemlerinde çizelgeleme problemlerinin çözümünde kullanılmak amacıyla öncelik kuralı keşif sistemi geliştirilmiştir. Geliştirilen öğrenme sistemi ile tek tezgahlı çizelgeleme problemleri için yapılan keşif çalışmalarından elde edilen sonuçlar ile literatürde kullanılan öncelik kurallarının karşılaştırılması yapılmıştır.
- Alikalfa ve Kapanoğlu (2010) tarafından çoklu ifade programlama tekniği ile tek tezgahlı çizelgeleme problemleri için toplam gecikme performans ölçütü göz önüne alınarak öncelik kuralları keşfedilmiştir. Keşfedilen öncelik kurallarının genelleştirilmesi ile öncelik kuralları çözüm kümesi bulunmuş ve literatürde kullanılan öncelik kuralları ile karşılaştırılması yapılmıştır.

BÖLÜM 4

ÖNCELİK KURALLARININ

GENETİK PROGRAMLAMA İLE KEŞFİ

Bu bölümde üretim sistemlerinde çizelgeleme problemlerine yönelik öncelik kurallarının geliştirilmesi için kullanılacak olan Çoklu İfade Programlama Tabanlı Öğrenme Sistemi (Multi Expression Programming Learning System - MELES) olarak adlandırdığımız yaklaşımımız, uygulamaları ve sonuçları ile ilgili bilgi verilecektir. MELES bütün üretim sistemlerine uygulanabilecek bir teknik olmasına rağmen uygulamalarımız literatürde kullanılan tek tezgah ve tek ölçütlü OR Library test problemleri üzerinde yapılmıştır.

4.1 Çoklu İfade Programlama Tabanlı Öğrenme Sistemi (MELES)

Çoklu ifade programlama tabanlı öğrenme sistemi (MELES) üretim sistemlerinde çizelgeleme problemlerinin çözümünde kullanılacak olan öncelik kurallarını sadece standart bilgi ve bilgi yapılarını kullanarak oluşturan bir öğrenme sistemidir. Öncelik kurallarının oluşturulması sırasında bir ya da daha fazla amacı göz önüne alarak yeni bilgi yapılarını geliştirir ve bu işlem sırasında aldığı ara karar ve bilgileri saklayarak ileri aşamalarda tekrar kullanabilir.

MELES amaçladığı bilgileri bulabilmek amacıyla çeşitli alt sistemlerden oluşmaktadır. Bu alt sistemler aşağıdaki gibidir:

a) Başlangıç Fonksiyon ve Terminal Kümeleri:

   Çoklu ifade programlama sistemi tarafından sadece başlangıçta oluşturulan aday çözümlerin içerisinde kullanılan öncül yapı taşlarını içermektedir. Bu kümelerin elemanları çözülecek olan probleme ve bulunmak istenen çözüm yöntemine özel olarak belirlenmektedir.

b) Çoklu İfade Programlama Sistemi:

   Seçim, çaprazlama ve mutasyon genetik operatörlerini kullanarak çözüm adaylarını belirlenmiş durma kriteri sağlanana kadar iyileştirmeye çalışan sistemdir.

c) Üretim Sistemi Simülasyonu Bileşeni:

   Çoklu ifade programlama sisteminin üzerinde çalıştığı çözüm adaylarının ele alınan problemi ne kadar iyi çözebildiğini gösteren uyumluluk değerlerini belirlemek için kullanılan deterministik simülasyon yapan bileşendir.

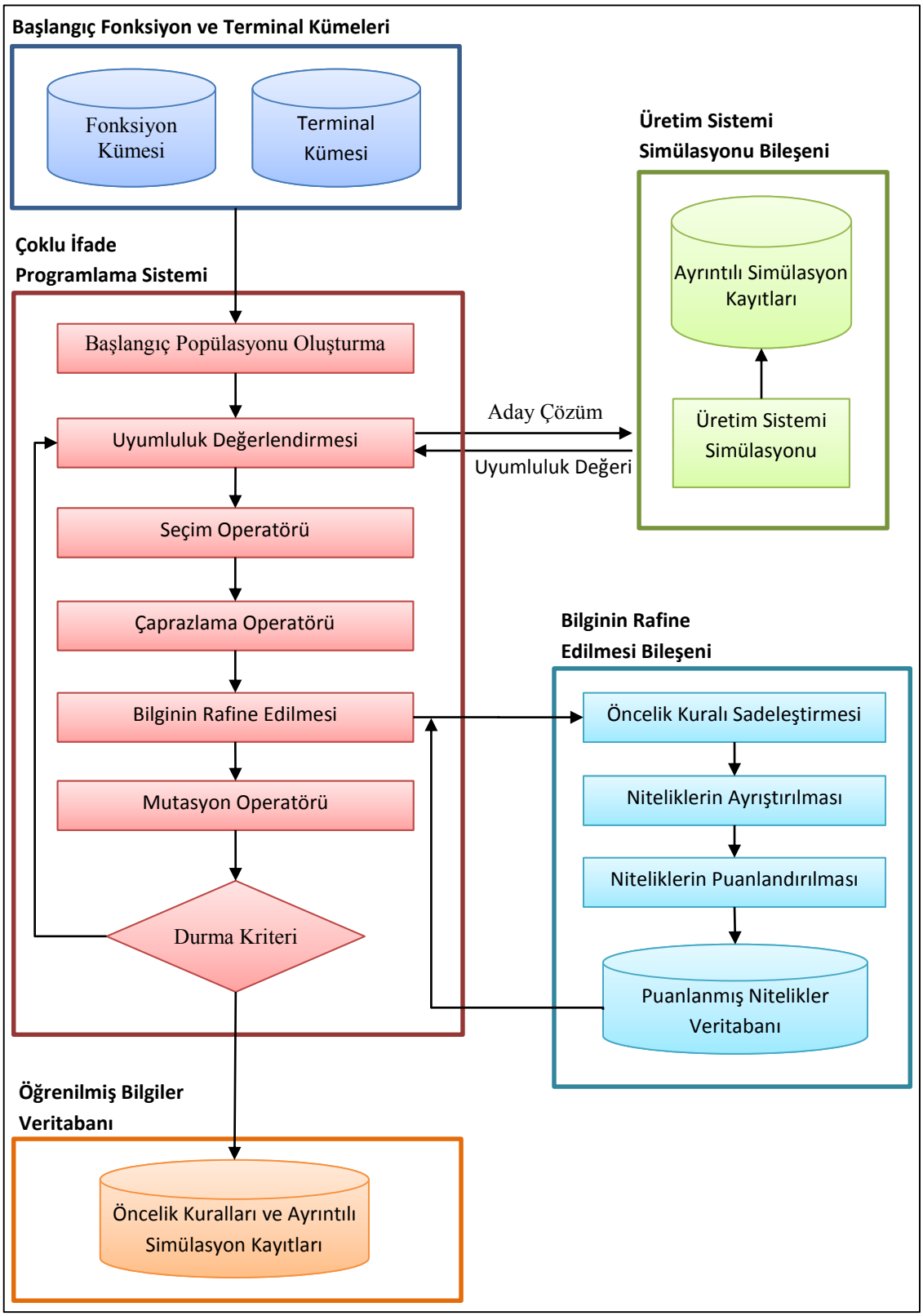
d) Bilginin Rafine Edilmesi Bileşeni:

   Çoklu ifade programlamanın üzerinde çalıştığı çözüm adaylarının barındırdığı ifadeleri (matematiksel denklem) sadeleştiren, bileşenlerine ayıran ve bileşenlerinin yararlılıklarına göre puanlanmasını (kredi) sağlayan bileşendir.

e) Öğrenilmiş Bilgiler Veritabanı:

   Çoklu ifade programlama tabanlı öğrenme sisteminin çalışması sırasında ve sonucunda oluşturulan çözümleri, çözümlerin bileşenleri ve puanlamaları ile çizelgeleme kararlarını barındıran veritabanıdır.

İlgili alt sistemler ve ilişkileri Şekil 4.1'de gösterilmiştir.



Şekil 4.1: MELES genel yapısı ve alt sistemleri

Bu çalışma kapsamında MELES ile üretim sistemlerinde çizelgeleme problemlerinin çözümünde kullanılacak olan öncelik kurallarının geliştirilmesi için yapılandırılması izleyen kısımda verilmiştir.

4.1.1 Başlangıç fonksiyon ve terminal kümeleri

Çoklu ifade programlama tabanlı öğrenme sistemimizin üreteceği öncelik kurallarının yapı taşları olarak kullanılacak olan terminal ve fonksiyon kümeleri Çizelge 4.1 ve Çizelge 4.2'de verilmiştir. İlgili terminal ve fonksiyon kümeleri, çözümlerinin geliştirileceği problemler ve daha önce geliştirilmiş olan öncelik kuralları incelenerek belirlenmiştir. Terminal ve fonksiyon kümelerinin elemanlarının geliştirilen öncelik kurallarında kullanılıp kullanılmaması kararı MELES tarafından ilgili elemanların katkılarına göre belirlenmekte ve böylece insan faktörü karar aşamasından çıkarılmaktadır. Ayrıca MELES belirlenmiş olan bu yapı taşlarını kullanarak kendi yapı taşlarını türetme ve kullanma yeteneğine sahiptir.

Çizelge 4.1: Terminal kümesi

Terminal Adı	Tanım	Formülasyon
TNOW	İlgili andaki zaman	
PT	İşin toplam işlem süresi	p_j
DD	Teslim zamanı	d_j
RT	İşin sisteme gönderildiği zaman	r_j
SL	Aylaklık	$\{d_j - p_j - TNOW\}$
w	Ağırlık	w_j
NinQ	Kuyruktaki iş sayısı	nq
SumPTinQ	Kuyruktaki işlerin işlem süreleri toplamı	$\sum_j p_j$
SumDDinQ	Kuyruktaki işlerin teslim zamanları toplamı	$\sum_j d_j$
AvePTinQ	Kuyruktaki işlerin işlem süreleri ortalaması	$\sum_j p_j / nq$
AveDDinQ	Kuyruktaki işlerin teslim zamanları ortalaması	$\sum_j d_j / nq$
WTinS	İşin sistemde bekleme süresi	$max\{r_j - TNOW, 0\}$
SumWTinS	İşlerin sistemde bekleme sürelerinin toplamı	$\sum_j max\{r_j - TNOW, 0\}$
AveWTinS	İşlerin sistemde bekleme sürelerinin ortalaması	$\sum_j max\{r_j - TNOW, 0\} / nq$
SumWeight	İşlerim ağırlıklarının toplamı	$\sum_j w_j$
AveWeight	İşlerin ağırlıklarının ortalaması	
VALUE	Rassal oluşturulmuş tamsayı değer	RND(minValue, maxValue)

Çizelge 4.2: Fonksiyon kümesi

Fonksiyon Adı	Tanım	Formülasyon
ADD	İkili toplama	a + b
SUB	İkili çıkarma	a – b
MUL	İkili çarpma	a * b
DIV	İkili bölme	a/b
POS	Pozitif durum	max {a, 0}
NEG	Negatif durum	– a
POW	Üs alma	$\|a\|^b$
EXP	Üstel	e^a
MIN	En küçük	min (a, b)
MAX	En büyük	max (a, b)
AVE	Ortalama	(a + b) / 2
SQRT	Karekök	$SQRT(a) = \begin{cases} 1 & , eğer\ a < 0 \\ \sqrt{a} & d.d. \end{cases}$
ABS	Mutlak Değer	$\|a\|$
SIGM	Sigmoid fonksiyonu	$\frac{1}{1+e^{-a}}$

4.1.2 Başlangıç popülasyonu

Başlangıç popülasyonu ele alınan problemdeki iş sayısına bağlı olarak 50 ile 200 kromozomdan oluşmakta ve MELES'in çalışmaya başladığı anda rastgele olarak doldurulmaktadır. Kromozomlardaki gen sayısı (derinlik) yine problemdeki iş sayısına bağlı olarak belirlenmekte ve 50 ile 150 arasında değişmektedir.

4.1.3 Uyumluluk fonksiyonu

Popülasyondaki kromozomların uyumluluk değerleri olarak ele alınan problemdeki işlerin toplam gecikme süresi kullanılmaktadır. Bir j işinin gecikme miktarı C tamamlanma zamanı ve d ilgili işin teslim zamanı olmak üzere $T_j = \max(0, C_j - d_j)$ olarak belirlenmektedir. Problemdeki bütün işlerin toplam gecikme miktarı ise $T_{toplam} = \sum_{j=1}^{n} T_j$ olarak belirlenmektedir. Eğer ilgili işin tamamlanma zamanı teslim zamanını geçmiş ise bu iş için bir gecikme oluşmuş bulunmaktadır (Li, et al., 2012).

4.1.4 Çoklu ifade programlama operatörleri

Bu bölümde MELES yapısı içerisinde kullanılan operatörler ve operatörlerin uygulama parametrelerine değinilmiştir.

4.1.4.1 Seçim operatörü

MELES'in bir alt bileşeni olan çoklu ifade programlama tekniğinin seçim operatörü olarak turnuva seçim tekniği uygulanmaktadır. Turnuva seçimi tekniğinde turnuva seçim büyüklüğü 2 olarak kullanılmıştır. Mevcut popülasyondaki en iyi uyumluluk değerine sahip kromozomun ilerleyen aşamalardaki popülasyona aktarılmasını garantilemek için ise Elitizm uygulanmıştır. Dolayısıyla her popülasyonun en iyi uyumluluk değerine sahip kromozomun barındırdığı çözümler kaybedilmemiş olmaktadır.

4.1.4.2 Çaprazlama operatörü

Çaprazlama operatörü olarak tek nokta basit çaprazlama operatörü kullanılmaktadır. Tek nokta çaprazlama operatörü ile iki farklı bireyde bir çaprazlama noktası belirlenmekte ve parçalar değiş tokuş edilerek iki adet çocuk birey oluşturulmaktadır. Çaprazlama operatörünün kromozomlar üzerindeki uygulanma olasılığı % 80 olarak belirlenmiştir.

4.1.4.3 Mutasyon operatörü

Mutasyon operatörü olarak tek nokta mutasyon operatörü kullanılmaktadır. Bu mutasyon operatörü ile kromozom üzerinde bir gen seçilmekte ve gen içeriğine bağlı olarak mutasyon uygulanmaktadır. Eğer seçilen gen bir fonksiyon barındırıyorsa Fonksiyon kümesi elemanlarından biri ile rastgele değiştirilmektedir. Burada fonksiyonların seçilme olasılıkları birbirine eşittir. Eğer seçilen gen bir nitelik barındırıyorsa MELES bilginin rafine edilmesi bileşeni tarafından oluşturulmuş olan özel bir veritabanı içerisindeki elemanlardan biri ile niteliklerin puanına bağlı olarak değiştirilmektedir. Puanlanmış Nitelik Veritabanı olarak isimlendirdiğimiz bu özel veritabanında yer alan bir niteliğin puanı ne kadar yüksek ise ilgili niteliğin seçilme olasılığı o kadar artmaktadır. Son olarak seçim bir genin içerisindeki referans bilgisi ise başka bir gene referans ile rastgele değiştirilmektedir. Mutasyon operatörünün kromozomlar üzerinde uygulanma olasılığı % 10 olarak belirlenmiştir.

4.1.5 Bilginin rafine edilmesi bileşeni

Bilginin rafine edilmesi bileşeni, kromozomlar içerisinde kodlanmış olan öncelik kurallarının yapı taşlarının (nitelikler) çözüme ne kadar katkısı olduğunu belirlemekte

ve ilgili nitelikleri bu katkıyı yansıtacak şekilde puanlamaktadır. Bu puanlama işlemi ile çözüm uzayının mutasyon yardımıyla daha bilinçli aranması sağlanmaktadır. Bir niteliğin puanı hesaplanırken ilgili niteliğin popülasyondaki bütün öncelik kurallarındaki tekrar sayısı ve bulunduğu öncelik kuralının uyumluluk değeri kullanılmaktadır.

Niteliklerin tekrar sayılarının belirlenmesi öncesinde öncelik kurallarını oluşturan matematiksel ifadeler matematiksel olarak sadeleştirilmektedir. Bu sadeleştirme işlemi için C# ve F# dilleri ile Maxima yazılımını entegre eden bir modül geliştirilmiştir. Çizelge 4.3'de bu işlemler kısaca gösterilmiştir.

Çizelge 4.3: Bilginin rafine edilmesi örneği

Öncelik kuralı	**Sadeleştirilmiş Öncelik Kuralı**	**Uyumluluk Değeri**	**Ayrıştırılmış Nitelikler**
2*PT + DD^2 - PT	PT + DD^2	50	PT, DD, DD^2, PT + DD^2
(RL^2 + RL*PT) / RL	RL + PT	70	RL, PT, RL + PT
(SL + DD) / DD	(SL + DD) / DD	20	SL, DD, SL + DD, SL / DD, (SL + DD) / DD

Çizelge 4.3'de bulunan öncelik kurallarının niteliklerinin ayrıştırılmasının ardından her bir nitelik için ayrı ayrı puan hesaplanır. Örneğin DD niteliğinin puanı bulunduğu birinci ve üçüncü öncelik kurallarının uyumluluk değerleri ile bir sabit sayı kullanılarak aşağıdaki gibi hesaplanmaktadır.

$$DD_{puan} = (70 - 50) + (70 - 20) + SabitPuan$$

Öncelik kurallarının uyumluluk değerlerinin hesaplanmasında işlerin toplam gecikmelerinin toplamı kullanıldığı için bu değerin direkt puan olarak kullanılması kötü

olan öncelik kuralında bulunan niteliğe yüksek puan vermek anlamına geleceğinden bütün uyumluluk değerleri düzeltmeye tabi olarak kullanılmaktadır. Düzeltme amacıyla uyumluluk değerleri popülasyondaki en kötü uyumluluk değerinden çıkartılarak kullanılmaktadır. Popülasyondaki en büyük uyumluluk değeri, her niteliğin puanına sabit olarak eklenen SabitPuan olarak kullanılmakta ve en büyük ile en küçük nitelik puanı arasındaki sayısal değer farkını azaltmak için kullanılmaktadır. SabitPuan ile her niteliğe mutasyon işlemi sırasında kayda değer bir seçilme şansı verilmekte ve çok yüksek puana sahip niteliklerin seçilerek çözümleri domine etmemeleri ve yerel eniyilere takılmama sağlanmaktadır.

Popülasyondaki öncelik kurallarında bulunmamakla birlikte başlangıç nitelik kümesinde bulunan nitelikler sadece SabitPuan atanarak Puanlanmış Nitelik Veritabanına eklenmektedir.

4.2 Oluşturulan Test Problemleri

Geliştirilmiş olan çoklu ifade programlama tabanlı öğrenme sistemimizin üzerinde çalışması için kullanılacak olan test problemleri, aynı alandaki diğer araştırmacıların çoğunluğunun kullandığı OR Library test problemi oluşturma yöntemi (Dimopoulos and Zalzala, 2001; Chou, 2009) ile oluşturulmuştur.

MELES ile öncelik kuralı keşfi aşamasında iki grup test problemi oluşturulmuştur. MELES tarafından öncelik kurallarının geliştirmesi için sadece birinci grup test problemleri kullanılmaktadır. Geliştirilen öncelik kurallarının literatürdeki diğer çalışmalarda kullanılan öncelik kuralları ile karşılaştırılması için ise ikinci grup test problemleri kullanılacaktır. Her iki gruptaki test problemindeki işlerin işlem süreleri tamsayı şeklindedir. Birinci grup test problemlerindeki işlem süreleri düzgün dağılım ile [1, 100] arasında türetilmiştir. İkinci test grubu ise işlerin işlem süreleri yine tamsayı olacak şekilde düzgün dağılım ile hem [1, 10] hem de [1, 100] arasında türetilen problemlerden oluşmaktadır. Bazı araştırmacılar işlem sürelerini sadece [1, 10] arasında belirlemesine rağmen [1, 100] arası belirlenen işlem süreli işlere sahip olan

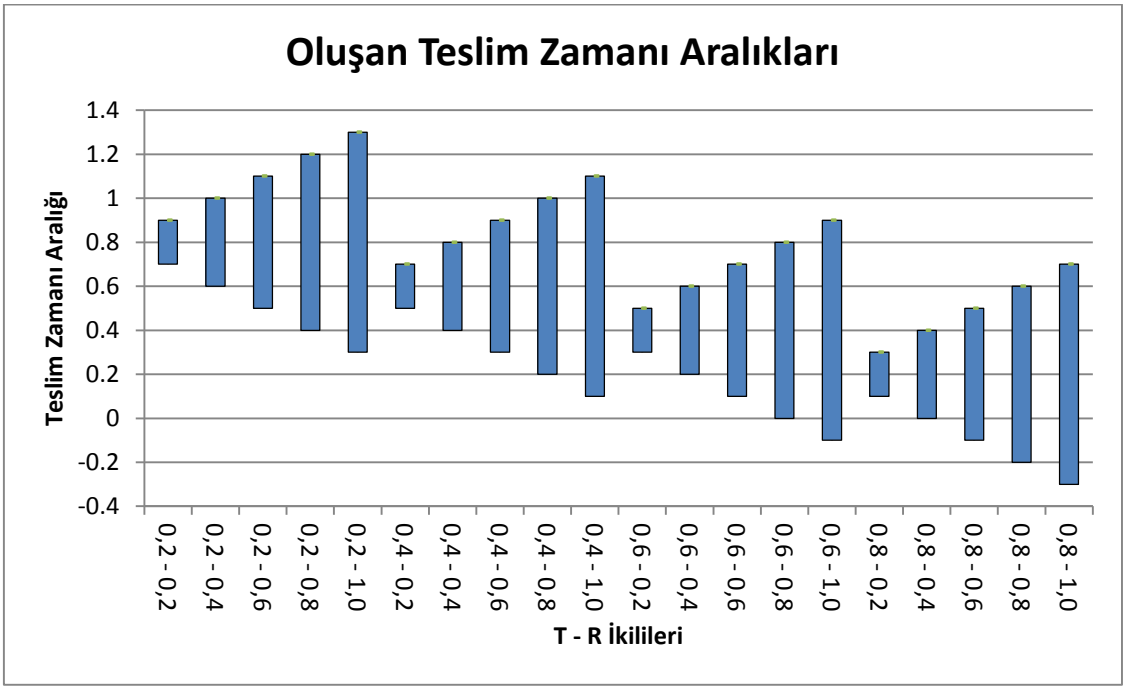
problemler daha zor problemler olarak kabul edilmektedir (Dimopoulos and Zalzala, 2001). Farklı işlem süresi dağılımına sahip test problemleri oluşturarak daha farklı karakteristiklere sahip problemler üzerinde MELES ile geliştirilen öncelik kurallarının performans değişimlerinin gözlenmesi amaçlanmıştır. Bütün test problemlerinde işler "0" anında sistemde bulunmakta ve her birinin ağırlığı 1 olacak şekilde birbirine eşit olarak işler ağırlıklandırılmaktadır. Bütün test problemlerinde işlerin teslim zamanları yine tamsayı olacak şekilde ve düzgün dağılım ile

$$[SP(1 - T - (R/2)), SP(1 - T + (R/2))]$$

aralığında olacak şekilde belirlenmiştir (Dimopoulos and Zalzala, 2001; Chou, 2009). Bu formülde SP problemdeki bütün işlerin işlem sürelerinin toplamı, T (TF – Tardiness Factor) gecikme faktörü ve R (RDD – Range of Due Date) teslim zamanı aralığıdır. Formüldeki T parametresi geciken işlerin beklenen yüzdesini göstermektedir. R parametresi ise teslim zamanı sıkılığını ya da oluşan teslim zamanı genişliğini göstermektedir. Problemlerin oluşturulması sırasında kullanılan teslim zamanı aralıkları, gecikme faktörleri ve bunlara göre oluşan teslim zamanı aralıkları Çizelge 4.4'de verilmiştir. Birinci test grubu için çizelgedeki her bir T ve R parametre çifti için 250 farklı problem oluşturulmuştur.

Çizelge 4.4: Gecikme zamanı sıklıkları (R), gecikme faktörleri (T) ve T ve R'ye göre oluşan teslim zamanı aralıları

T	R	Teslim Zamanı Alt Sınırı	Teslim Zamanı Üst Sınırı
0,2	0,2	0,7	0,9
0,2	0,4	0,6	1
0,2	0,6	0,5	1,1
0,2	0,8	0,4	1,2
0,2	1	0,3	1,3
0,4	0,2	0,5	0,7
0,4	0,4	0,4	0,8
0,4	0,6	0,3	0,9
0,4	0,8	0,2	1
0,4	1	0,1	1,1
0,6	0,2	0,3	0,5
0,6	0,4	0,2	0,6
0,6	0,6	0,1	0,7
0,6	0,8	0	0,8
0,6	1	-0,1	0,9
0,8	0,2	0,1	0,3
0,8	0,4	0	0,4
0,8	0,6	-0,1	0,5
0,8	0,8	-0,2	0,6
0,8	1	-0,3	0,7



Şekil 4.2: T – R ikililerine göre oluşan teslim zamanı aralıklarının grafiği

Birinci test grubu için oluşturulan problemlerdeki iş sayıları ise 3, 4, 5, 6, 7, 8, 9 ve 10'dur. Örneğin 6 işli ve Çizelge 4.4'de verilen değişik T ve R parametre çiftlerinin hepsi için oluşturulan toplam problem sayısı 5000'dir. Aynı şekilde diğer bütün iş sayıları için 5000'er problem olmak üzere birinci test grubu için toplamda işlem süreleri [1, 100] arası düzgün dağılan 40000 problem oluşturulmuştur.

İkinci test grubu için oluşturulan problemlerdeki iş sayıları ise 3, 4, 5, 6, 7, 8, 9, 10, 12, 15, 20, 30, 40, 50, 75, 100, 150, 200, 250, 500, 750 ve 1000'dir. Bu iş sayılarına sahip problemler işlem süreleri [1, 10] olmak üzere 4400 adet, işlem süresi [1, 100] arası düzgün dağılan şekilde olmak üzere 4400 adet ve toplamda 8800 adet oluşturulmuştur.

Birinci test grubunda bulunan 3, 4, 5, 6, 7, 8, 9 ve 10 işli problemlerden oluşan alt gruptaki 40.000 problemin her birinin Dal-Sınır algoritması ile işlerin gecikmelerinin toplamı bulunmuş ve MELES ile geliştirilen öncelik kurallarının en iyi çözüme yaklaşma dereceleri incelenmiştir. İkinci test grubundaki problemler ise genetik

algoritma ile iş sıralarının aranması yöntemi (gerçek sayı kodlamalı sabit uzunluklu kromozom, PMX çaprazlama ve klasik mutasyon) ile çözdürülerek (Sivanandam and Deepa, 2008) elde edilen toplam gecikme değerleri bulunmuştur. Bu toplam gecikme değerleri MELES ile geliştirilmiş öncelik kurallarının performans değerlendirilmelerinde kullanılmıştır. Çizelge 4.5'de problem gruplarının içerikleri ve kullanım amaçları özet şeklinde gösterilmektedir.

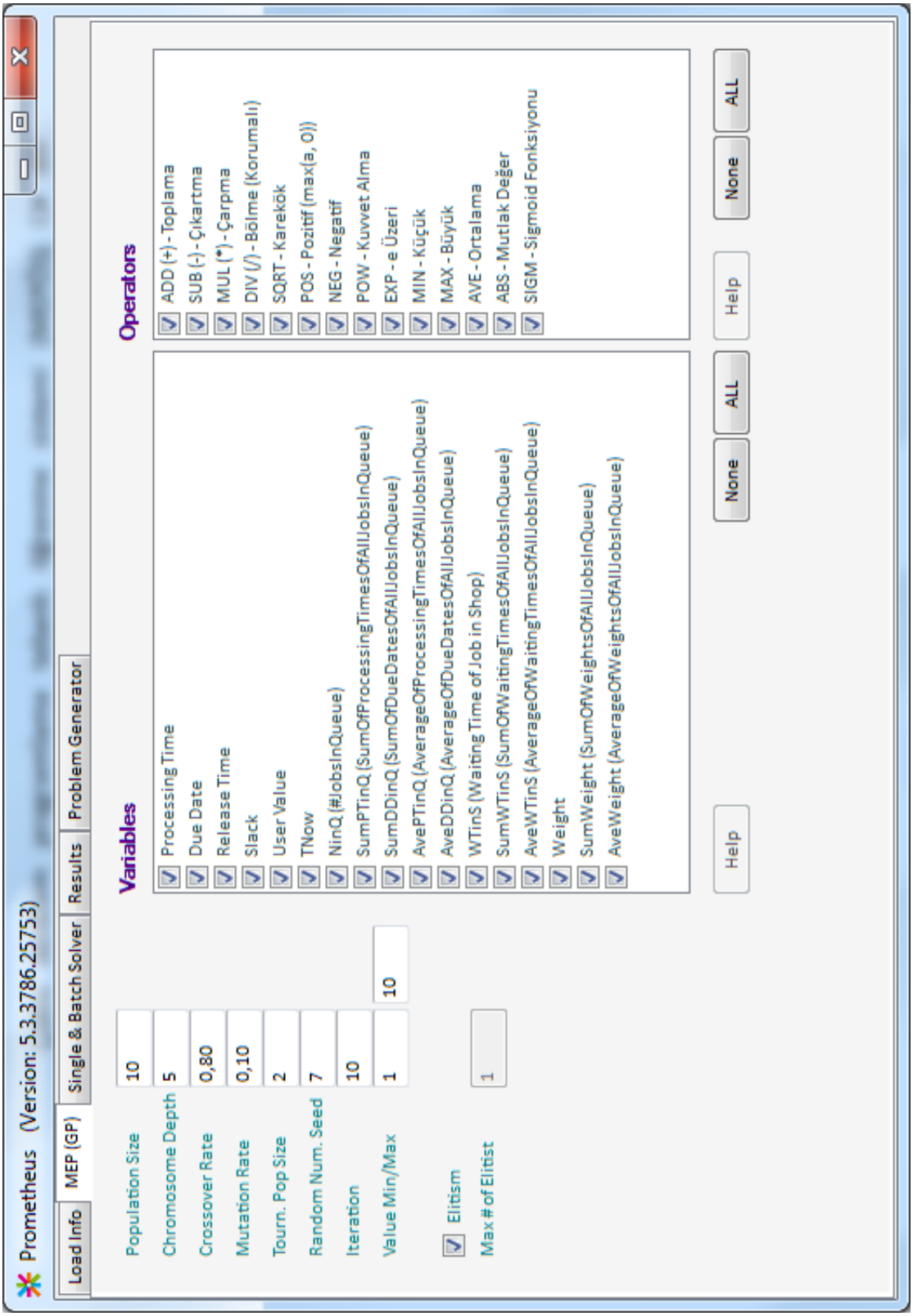
Çizelge 4.5: Problem gruplarının içerikleri ve kullanım amaçları

Grup Adı	Gruptaki problemlerin iş sayıları	İşlem süreleri dağılımı	Toplam Problem Sayısı	Kullanım Şekli
1. Grup	3-10	[1, 100]	40000	Öncelik kuralı geliştirme
2. Grup	3-1000	[1, 10]	4400	Performans ölçümü
	3-1000	[1, 100]	4400	Performans ölçümü

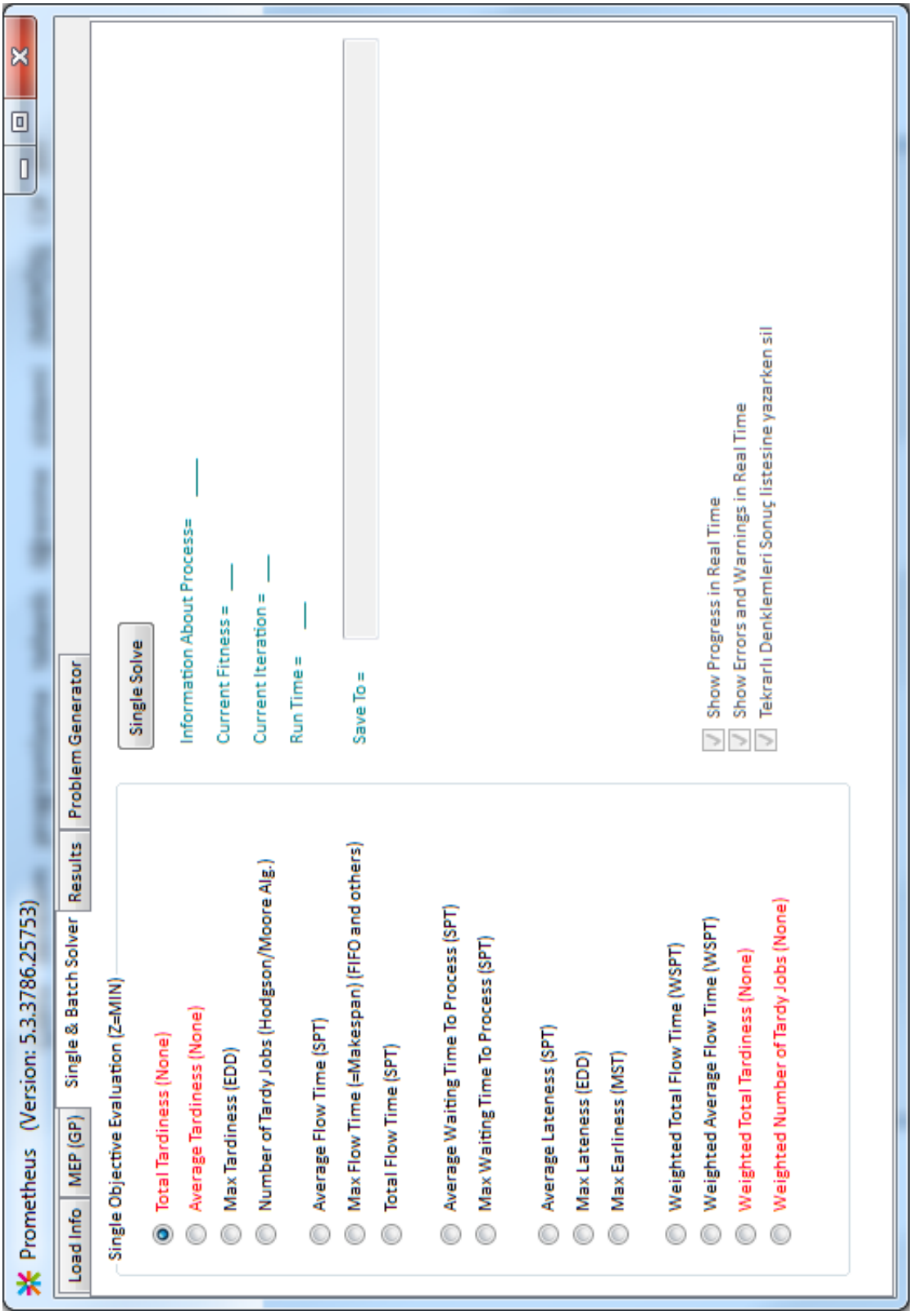
Problemlerin bu kadar sayıda çeşitli iş sayısı ve parametrelere göre oluşturulmasının sebebi her bir problem için geliştirilecek öncelik kuralının ilgili iş ve parametrelere göre nasıl değişeceğini ve davranışını gözlemlemektir.

4.3 Geliştirilen Program

Çoklu ifade programlama tabanlı öğrenme sistemimiz C# dili kullanılarak geliştirilmiştir. Test problemleri ise Core i7 920 (8way) işlemci, 4 GB RAM'e sahip bir bilgisayar ve Xeon E5-2650 2GHz (64way) işlemci, 196 GB RAM'e sahip bir sunucu üzerinde çözülmüştür. Geliştirilen programın iki arayüz görüntüsü Şekil 4.3 ve Şekil 4.4'de verilmiştir.



Şekil 4.3: Çoklu ifade programlama bölümü arayüzü



Şekil 4.4: Amaç fonksiyonu ve çözüm durumu bölümü arayüzü

Geliştirilen programda aşağıdaki 16 adet amaç fonksiyonu (performans ölçütü) kullanılabilmektedir. Amaç fonksiyonu listesinde parantez içlerinde ilgili amaç fonksiyonunu eniyileyen öncelik kuralları da verilmiştir.

1) Toplam gecikme (Bilinen bir öncelik kuralı yoktur),
2) Ortalama gecikme (Bilinen bir öncelik kuralı yoktur),
3) En büyük gecikme (EDD),
4) Geciken işlerin toplam sayısı (Hodgson/Moore algoritması),
5) Ortalama akış süresi (SPT),
6) En büyük akış süresi (Birden çok kural bulunmaktadır),
7) Toplam akış süresi (SPT),
8) Ortalama bekleme süresi (SPT),
9) En büyük bekleme süresi (SPT),
10) Teslim zamanından ortalama sapma (SPT),
11) Teslim zamanından en büyük sapma (EDD),
12) En büyük erken bitirme (MST),
13) Ağırlıklı toplam akış süresi (WSPT),
14) Ağırlıklı ortalama akış süresi (WSPT),
15) Ağırlıklı toplam gecikme (Bilinen bir öncelik kuralı yoktur),
16) Ağırlıklı geciken işlerin toplam sayısı (Bilinen bir öncelik kuralı yoktur).

4.4 Çoklu İfade Programlama Uygulama Parametreleri

Çoklu ifade programlama tabanlı öğrenme sisteminin (MELES) çalıştırılması esnasında kullanılan uygulama parametreleri özetle Çizelge 4.6'da verilmiştir. İlgili parametreler daha önceki yapılan çalışmalar ve üretilen test problemleri üzerinde yapılan denemeler sonucunda belirlenmiştir.

Çizelge 4.6: MELES parametreleri

Parametre	Değer
Popülasyon büyüklüğü	50 – 200
Kromozom derinliği	50 - 150
Çaprazlama oranı	% 80
Mutasyon oranı	% 10
Turnuva büyüklüğü	2
Iterasyon	100 – 2000
Elitist sayısı	1

4.5 Sonuçlar

Çoklu ifade programlama tabanlı öğrenme sisteminin (MELES) toplam gecikme ölçütü göz önüne alınarak geliştirilmiş olan öncelik kuralları ve bu öncelik kuralları üzerine yapılmış olan değerlendirmeler bu bölümde verilmiştir.

MELES'in öncelik kurallarını bulabilirliğinin test edilmesi amacıyla performans ölçütlerini eniyileyen öncelik kurallarının bilindiği durumlar için (En büyük gecikme, Ortalama akış süresi, En büyük akış süresi, Toplam akış süresi, Ortalama bekleme süresi, En büyük bekleme süresi, Teslim zamanından ortalama sapma, Teslim zamanından en büyük sapma ve En büyük erken bitirme) birinci grup test problemleri çözdürülmüştür. Ele alınan her bir problem ve performans ölçütü için performans ölçütünü eniyileyen ilgili öncelik kuralı MELES tarafından bulunmuştur. Böylece geliştirilmiş olan MELES'in ilgili öncelik kurallarını bulabilirliği test edilmiştir.

MELES'in doğrulamasının yapılmasının ardından toplam gecikme ölçütü için öncelik kurallarının geliştirilmesi aşamasına geçilmiştir. Dal-Sınır algoritması ile toplam gecikme değerleri belirlenen birinci gruptaki toplam 40.000 problem üzerinde toplam gecikmenin enküçüklenmesi amaç fonksiyonu göz önüne alınarak farklı rassal

sayı çekirdekleri (1, 2, 3 ve 4) kullanılarak çözdürülmüş ve Dal-Sınır algoritması ile bulunan en iyi çözümlerle karşılaştırılmıştır. Her bir çalıştırma sonucu oluşturulan öncelik kuralı ile ilgili problemin en iyi çözümü elde edilmiştir.

Bulunan sonuçlara örnek olarak 6 ve 8 işli problemlerin bir kısmı için sonuçlar izleyen çizelgelerde (Çizelge 4.7 ve Çizelge 4.8) verilmiştir. Birinci gruptaki problemlerin çözdürülmesi sonucu yaklaşık olarak 200.000.000 öncelik kuralı (40.000 problem x 4 rassal sayı çekirdeği x popülasyon başına ortalama 120 kromozom x popülasyon başına ortalama bulunan öncelik kuralı) elde edilmiş ve değerlendirilebilmesi ve tekrar eden kuralların elenebilmesi için sadeleştirilmiştir. Bu sadeleştirme işlemi C# ve F# dilleri geliştirilmiş fonksiyonlar ve Maxima programı ile otomatik olarak yapılmıştır. Bu aşamada MELES ile oluşturulan dosyaların toplam büyüklüğü sıkıştırılmamış halde 1.2 Terabyte'dır. Bu dosyalar sadeleştirilmemiş öncelik kurallarını ve problemlerin simülasyonları sırasında alınan kararlar, karar alternatifleri ve simülasyon ortamı anlık bilgilerini kapsamaktadır.

Çizelge 4.7: 6 işli problemlerin durum çizelgesi ve elde edilen denklemler

TF	RDD	Prob	Bulunan Denklemler	En İyi Çözüm	Bulunan Çözüm	Bilinen bir kural mı?
0,2	0,2	1	SL x PT Max(SL, PT) Sqrt(PT) + DD	29	29	
0,2	0,2	2	Max(SL, PT) Max(SL, PT)/(Sqrt(SL)+AveDDinQ)	29	29	
0,2	0,2	3	SL x PT PT + SL x PT Max(SL, PT) Max(PT x SL, PT) Max(PT x SL, SumDDinQ) Max(PT x SL, 0) Max(SL, PT x SL) PT x SL x Max(SL, PT x SL) (SL^2 + SL) x PT e^(Max(SL, PT)) (PT x SL) / WT Sqrt(PT) + (PT x SL) / WT	29	29	
0,2	0,2	4	Max(SL, PT) Max(RT^AveDDinQ, Max(SL, PT)) SL x PT Max(SL, (PT + 49)/2)	29	29	
0,2	0,2	5	2 x SL x Abs(DD) / (SL + 1)	61	61	
0,2	0,2	1	Min(Max(SL,0),Sqrt(DD)) Pos(Min(SL,Sqrt(DD))) Max(Min(SL,Sqrt(DD)),1)	61	61	
0,2	0,2	2	Min(SumDDinQ,Tnow + DD + DD)	61	61	
0,2	0,2	3	Ave(PT,Pos(SL)) Max(Tnow, Ave(DD, SumPTinQ)) PT + Pos(SL)	61	61	
0,2	0,2	4	DD Ave(DD, QC)	30	30	EDD
0,2	0,2	5	DD Ave(PT, SL)	30	30	EDD
0,2	0,2	1	DD Ave(PT, SL) Max(PT, DD) Max(AveWTinS, DD)	30	30	EDD
0,2	0,2	2	DD Div(2 x SL, Abs(SL) - 4 x SL^2) (Pos(SL) - AvePTinQ) / SL	30	30	EDD
0,2	0,2	3	PT RT	29	29	SPT FIFO
0,2	0,2	4	SL^2 Abs(SL) Sqrt(Abs(SL)) Abs(SL) + AveDDinQ^2 e^(Abs(SL)) PT x SL^2 Abs(SL) + AveDDinQ^2 + PT	85	85	
0,2	0,2	5	Max(PT, SL)	85	85	
0,2	0,4	1	DD	3	3	EDD

Çizelge 4.7 devamı

TF	RDD	Prob	Bulunan Denklemler	En İyi Çözüm	Bulunan Çözüm	Bilinen bir kural mı?
0,2	0,4	2	DD Sqrt(SL) Abs(SL)	122	122	EDD
0,2	0,4	3	DD	8	8	EDD
0,2	0,4	4	DD	45	45	EDD
0,2	0,4	5	DD SL	3	3	EDD MST
0,2	0,6	1	DD	95	95	EDD
0,2	0,6	2	DD SL	19	19	MST EDD
0,2	0,6	3	PT DD SL	0	0	SPT EDD MST
0,2	0,6	4	PT Neg(SL)	63	63	SPT
0,2	0,6	5	DD	0	0	EDD
0,2	0,8	1	DD	74	74	EDD
0,2	0,8	2	DD	65	65	EDD
0,2	0,8	3	DD	0	0	EDD
0,2	0,8	4	PT DD SL	0	0	SPT EDD MST
0,2	0,8	5	PT DD	18	18	SPT EDD
0,2	1,0	1	DD SL	0	0	EDD MST
0,2	1,0	2	DD SL	0	0	EDD MST
0,2	1,0	3	DD RT SL	0	0	EDD FIFO MST
0,2	1,0	4	DD RT SL	0	0	EDD FIFO MST
0,2	1,0	5	DD	23	23	EDD
0,4	0,2	1	DD + PT Abs(PT - DD) Max(2 x PT, DD) PT + Tnow PT^DD	72	72	
0,4	0,2	2	PT/Tnow Tnow x PT^2 PT/AveWT PT^Tnow	149	149	
0,4	0,2	3	Min(PT, SL) Max(PT, SL) Sqrt(PT, SL) Abs(PT x SL)	147	147	

Çizelge 4.7 devamı

TF	RDD	Prob	Bulunan Denklemler	En İyi Çözüm	Bulunan Çözüm	Bilinen bir kural mı?
0,4	0,2	4	PT Neg(DD) Neg(SL)	54	54	SPT
0,4	0,2	5	Max(DD - SL, DD)	73	73	
0,4	0,4	1	DD	166	166	EDD
0,4	0,4	2	Max(DD - SL, DD)	107	107	
0,4	0,4	3	Ave(PT, Sigm(SumDD) + DD) Ave(PT, Sigm(PT) + DD)	131	131	
0,4	0,4	4	RT	96	96	FIFO
0,4	0,4	5	DD	134	134	EDD
0,4	0,6	1	DD	49	49	EDD
0,4	0,6	2	Abs(SL) - Ave(QC, SL) Abs(SL)+PT	144	144	
0,4	0,6	3	DD	64	64	EDD
0,4	0,6	4	DD	79	79	EDD
0,4	0,6	5	DD	173	173	EDD
0,4	0,8	1	DD	90	90	EDD
0,4	0,8	2	Div(PT, DD)	50	50	
0,4	0,8	3	DD SL	118	118	EDD MST
0,4	0,8	4	DD	94	94	EDD
0,4	0,8	5	DD	38	38	EDD
0,4	1,0	1	DD	65	65	EDD
0,4	1,0	2	PT + Pos(SL)	115	115	
0,4	1,0	3	DD SL	0	0	EDD MST
0,4	1,0	4	Sigm(PT)+Pos(SL)	262	262	
0,4	1,0	5	PT x Max(QC, Sqrt(SL))	316	316	
0,6	0,2	1	Ave(Pos(SL),PT)	281	281	
0,6	0,2	2	PT	230	230	SPT
0,6	0,2	3	Abs(SL-Sqrt(PT))	469	469	
0,6	0,2	4	PT	217	217	SPT
0,6	0,2	5	Ave(DD, PT) Max(SL, PT) PT + DD PT x DD	198	198	
0,6	0,4	1	Max(PT/2, QC x SL)	211	211	
0,6	0,4	2	Max(PT, 2 x SL)	237	237	
0,6	0,4	3	DD	270	270	EDD
0,6	0,4	4	PT + Abs(SL)	336	336	
0,6	0,4	5	DD + PT Max(2 x SL, PT) PT + Pos(SL)	229	229	
0,6	0,6	1	Pow(SL, DD) Max(SL, PT) Max(SL, WTinS)	166	166	
0,6	0,6	2	DD	138	138	EDD

Çizelge 4.7 devamı

TF	RDD	Prob	Bulunan Denklemler	En İyi Çözüm	Bulunan Çözüm	Bilinen bir kural mı?
0,6	0,6	3	DD	101	101	EDD
0,6	0,6	4	Ave(DD, PT) Max(PT, SL)	144	144	
0,6	0,6	5	DD	137	137	EDD
0,6	0,8	1	Pow(PT, DD) Pow(SL, DD) Ave(PT, DD) DD + PT	204	204	
0,6	0,8	2	Min(DD, PT) Pow(DD, PT)	444	444	
0,6	0,8	3	Sigm(DD) Pow(PT, DD) e^(DD x DD)	302	302	
0,6	0,8	4	PT	90	90	SPT
0,6	0,8	5	Max(PT,Abs(SL)^SL) SL/(PT^PT)	397	397	
0,6	1,0	1	Max(2 x SL, PT) Ave(PT, Pos(2 x SL))	241	241	
0,6	1,0	2	Max(Abs(SL), PT)	98	98	
0,6	1,0	3	DD	162	162	EDD
0,6	1,0	4	Ave(Ave(SL,Abs(SL)),PT)	216	216	
0,6	1,0	5	Max(PT, SL) Abs(Max(PT, SL))	353	353	
0,8	0,2	1	Max(PT, SL)	467	467	
0,8	0,2	2	PT Neg(SL)	285	285	SPT
0,8	0,2	3	PT + Max(SL, 1)	626	626	
0,8	0,2	4	PT	345	345	SPT
0,8	0,2	5	Abs(SL) AveDD/SL SL x SL 1/SL	804	804	
0,8	0,4	1	PT	598	598	SPT
0,8	0,4	2	Ave(PT, Abs(SL))	503	503	
0,8	0,4	3	PT	532	532	SPT
0,8	0,4	4	PT + DD Ave(PT, DD)	316	316	
0,8	0,4	5	Ave(PT, Abs(SL))	337	337	
0,8	0,6	1	PT	279	279	SPT
0,8	0,6	2	PT x PT x Abs(SL)	586	586	
0,8	0,6	3	PT	647	647	SPT
0,8	0,6	4	PT	868	868	SPT
0,8	0,6	5	Ave(PT, DD)	444	444	
0,8	0,8	1	Max(PT, SL) Max(PT, Pos(SL))	611	611	
0,8	0,8	2	Min(PT, DD) Ave(WT + DD, Sqrt(DD))	636	636	

Çizelge 4.7 devamı

TF	RDD	Prob	Bulunan Denklemler	En İyi Çözüm	Bulunan Çözüm	Bilinen bir kural mı?
0,8	0,8	3	PT	1093	1093	SPT
0,8	0,8	4	Max(PT, SL) PT x Abs(SL) SumWT + Max(PT, SL)	214	214	
0,8	0,8	5	Abs(SL) SL x SL	177	177	
0,8	1,0	1	SL x DD Div(9, SL) Div(8, SL) Sigm(SL) x SL	1086	1086	
0,8	1,0	2	Abs(DD) DD x DD	681	681	
0,8	1,0	3	PT x Abs(SL)	596	596	
0,8	1,0	4	Max(PT, SL)	1037	1037	
0,8	1,0	5	PT^Sqrt(Abs(SL)) Ave(e^SL, PT)	1191	1191	

Çizelge 4.8: 8 işli problemlerin durum çizelgesi ve elde edilen denklemler

TF	RDD	Prob	Bulunan Denklemler	En İyi Çözüm	Bulunan Çözüm	Bilinen bir kural mı?
0,2	0,2	1	Sigm(Sqrt(SL^2)) PT + DD Max(DD, PT + DD) Sqrt(e^(SL x SL))	65	65	
0,2	0,2	2	DD	71	71	EDD
0,2	0,2	3	DD	57	57	EDD
0,2	0,2	4	DD PT	48	48	EDD SPT
0,2	0,2	5	DD	64	64	EDD
0,2	0,4	1	DD RT	43	43	EDD FIFO
0,2	0,4	2	DD	16	16	EDD
0,2	0,4	3	DD	2	2	EDD
0,2	0,4	4	DD	0	0	EDD
0,2	0,4	5	DD	64	64	EDD
0,2	0,6	1	DD SL	0	0	EDD MST
0,2	0,6	2	DD	8	8	EDD
0,2	0,6	3	DD SL	0	0	EDD MST
0,2	0,6	4	DD	0	0	EDD
0,2	0,6	5	DD	9	9	EDD
0,2	0,8	1	DD SL	0	0	EDD MST
0,2	0,8	2	DD SL	0	0	EDD MST
0,2	0,8	3	DD SL	0	0	EDD MST
0,2	0,8	4	DD	18	18	EDD
0,2	0,8	5	DD SL	0	0	EDD MST
0,2	1,0	1	PT DD SL	0	0	SPT EDD MST
0,2	1,0	2	DD	0	0	EDD
0,2	1,0	3	DD SL	0	0	EDD MST
0,2	1,0	4	DD SL	0	0	EDD MST
0,2	1,0	5	DD SL	0	0	EDD MST
0,4	0,2	1	DD	177	177	EDD
0,4	0,2	2	Max(PT, SL) Max(PT, 2 x SL)	321	321	
0,4	0,2	3	Abs(SL)^(Max(AvePT, PT))	370	370	
0,4	0,2	4	PT	191	191	SPT
0,4	0,2	5	Ave(PT, Pos(SL))	212	212	
0,4	0,4	1	DD	65	65	EDD

Çizelge 4.8 devamı

TF	RDD	Prob	Bulunan Denklemler	En İyi Çözüm	Bulunan Çözüm	Bilinen bir kural mı?
0,4	0,4	2	Ave(PT, DD)	181	181	
0,4	0,4	3	Max(PT, SL)	155	155	
0,4	0,4	4	Neg(Abs(2 x SL + PT)^(2 X SL +PT))	196	196	
0,4	0,4	5	DD + Ave(PT, 1)	184	184	
0,4	0,6	1	DD	38	38	EDD
0,4	0,6	2	Max(PT, SL)	192	192	
0,4	0,6	3	DD	59	59	EDD
0,4	0,6	4	DD^DD Max(DD, SumPT)	124	124	
0,4	0,6	5	DD + Abs(SL)	184	184	
0,4	0,8	1	DD SL	27	27	EDD MST
0,4	0,8	2	Ave(Pos(SL), PT)	315	315	
0,4	0,8	3	Min(SumPT, DD)	23	23	
0,4	0,8	4	DD SL	4	4	EDD MST
0,4	0,8	5	DD	38	38	EDD
0,4	1,0	1	DD SL	0	0	EDD MST
0,4	1,0	2	Max(PT, SL) PT + Pos(SL)	366	366	
0,4	1,0	3	DD	50	50	EDD
0,4	1,0	4	Ave(PT, DD) PT + DD	185	185	
0,4	1,0	5	Pos(2 x SL) + Ave(SumWT, PT)	485	485	
0,6	0,2	1	Max(PT, SL)	442	442	
0,6	0,2	2	Max(PT, DD)	454	454	
0,6	0,2	3	Ave(PT, Pos(SL)) PT + Pos(SL)	587	587	
0,6	0,2	4	Ave(PT, Pos(SL)) PT + Pos(SL)	667	667	
0,6	0,2	5	Abs(Ave(SL - PT, QC))^(PT-Sqrt(SL))	470	470	
0,6	0,4	1	Max(PT, SL + PT)	277	277	
0,6	0,4	2	Ave(PT, SL) Abs(SL) x PT	609	609	
0,6	0,4	3	DD	357	357	EDD
0,6	0,4	4	Max(DD - SL, DD)	630	630	
0,6	0,4	5	Max(Ave(SumPT,e^DD),Pow(SL,PT))	270	270	
0,6	0,6	1	Ave(PT, Pos(SL))	344	344	
0,6	0,6	2	Max(PT, SL)	291	291	
0,6	0,6	3	Max(10, Div(PT/5))	235	235	
0,6	0,6	4	Max(PT, SL)	232	232	
0,6	0,6	5	DD	254	254	EDD
0,6	0,8	1	2 x DD + PT/2	253	253	
0,6	0,8	2	DD	324	324	EDD
0,6	0,8	3	Max(PT, SL)	560	560	
0,6	0,8	4	Max(PT, SL)	129	129	
0,6	0,8	5	DD	212	212	EDD

Çizelge 4.8 devamı

TF	RDD	Prob	Bulunan Denklemler	En İyi Çözüm	Bulunan Çözüm	Bilinen bir kural mı?
0,6	1,0	1	Max(DD x SL, 1)	444	444	
0,6	1,0	2	Max(SL^2, AveDD) Abs(SL)^2 x PT	464	464	
0,6	1,0	3	PT + Abs(DD)	470	470	
0,6	1,0	4	Ave(PT, SL)	665	665	
0,6	1,0	5	Ave(Min(AveWT, PT), DD)	181	181	
0,8	0,2	1	PT	687	687	SPT
0,8	0,2	2	Max(PT, SL) PT + Pos(SL)	866	866	
0,8	0,2	3	Abs(SL) SL X SL	764	764	
0,8	0,2	4	PT + Pos(SL)	974	974	
0,8	0,2	5	Max(PT, SL)	468	468	
0,8	0,4	1	Ave(PT, SL) PT + Pos(SL)	1075	1075	
0,8	0,4	2	DD-Min(SL, Sigm(SumWT))	619	619	
0,8	0,4	3	SL^(Sigm(SL)) + SL	686	686	
0,8	0,4	4	Max(PT, SL)	829	829	
0,8	0,4	5	Max(PT, SL)	802	802	
0,8	0,6	1	Max(PT, SL)	1318	1318	
0,8	0,6	2	Max(PT, SL) PT + Pos(SL)	1235	1235	
0,8	0,6	3	Max(PT, 2 x SL)	892	892	
0,8	0,6	4	PT	1720	1720	SPT
0,8	0,6	5	Max(PT, SL)	1215	1215	
0,8	0,8	1	Max(PT, 2 x SL)	1411	1411	
0,8	0,8	2	Max(PT, AvePT)	965	965	
0,8	0,8	3	Max(PT, 2 x SL)	888	888	
0,8	0,8	4	Max(PT, 2 x SL)	842	842	
0,8	0,8	5	PT + Pos(SL)	698	698	
0,8	1,0	1	PT + Pos(SL)	1364	1364	
0,8	1,0	2	Max(PT, SL)	1059	1059	
0,8	1,0	3	PT + Pos(SL)	331	331	
0,8	1,0	4	Max(PT, SL)	547	547	
0,8	1,0	5	PT + Pos(SL)	515	515	

Bu aşamaya kadar birinci grup test problemleri tek tek ele alınarak MELES ile öncelik kuralı geliştirilmiştir. Böylece ikinci test problemleri grubu üzerinde performansları denenecek olan MELES ile geliştirilmiş öncelik kuralları için bir çeşitlilik sağlanmış ve test problemleri tek tek ele alındığında en iyi çözüme öncelik kuralı geliştirilerek ulaşılabileceği görülmüştür. Çalışmanın izleyen aşamasında birinci grup test problemleri hem rastgele hem de sıralı şekilde (aynı TF ya da aynı RDD değerlerine sahip olacak biçimde) gruplandırılarak, gruplandırılmış test problemi alt

kümeleri için öncelik kuralları türetilecek ve gruplandırılmış problemlerin ortak karakteristiklerini göz önüne alan öncelik kuralları türetilecektir. Bu gruplamalarda alt kümelerdeki problem sayıları 2, 4, 5, 10 ve 20 olacak şekilde belirlenmiştir. Bu şekilde test problemleri rastgele seçilerek 44.000, sıralı şekilde seçilerek 44.000 olmak üzere 88.000 alt küme oluşturulmuş ve MELES yardımıyla bu alt kümeler için öncelik kuralları türetilmiştir.

Alt kümeler için MELES yardımıyla geliştirilmiş olan öncelik kuralları bir önceki aşamada problemlerin tek tek çözülerek elde edilmiş olan öncelik kuralları ile sadeleştirilerek birleştirilmiştir. Birleştirme işleminin ardından tekrar eden kuralların elenmesi sonucu toplamda 5.574.279 adet farklı (birebir aynı olmayan) öncelik kuralı elde edilmiştir. Bu kurallar dal-sınır algoritması yardımıyla en iyi toplam gecikme değerleri elde edilmiş olan birinci test grubundaki problemler üzerinde tek tek denenerek her bir öncelik kuralının en iyi değerinden sapma miktarları elde edilmiştir. Ardından birinci gruptaki toplam 40.000 problemden en az 400 tanesini çözebilen 8142 adet öncelik kuralı seçilerek bir aday öncelik kuralları kümesi oluşturulmuştur.

Aday öncelik kuralları kümesindeki 8142 adet öncelik kuralının performanslarının daha fazla karakteristik çeşitliliğe sahip bir ortamda değerlendirilmesi için ikinci gruptaki test problemleri kullanılmıştır. İkinci test grubundaki 8800 adet problemin her biri 8142 adet öncelik kuralı ve klasik öncelik kuralları ile çözdürülerek karşılaştırmaları yapılmıştır. Elde edilen sonuçlar incelendiğinde geliştirilen öncelik kurallarından sadece biri kullanılarak bütün problemler çözülememesine rağmen bazı öncelik kurallarının klasik kurallardan çok daha iyi sonuç verdiği görülmüştür. Bu öncelik kuralları aşağıda verilmiştir.

Kural 1: Max (SL, PT) + PT + Max (SL, 0)

Kural 2: Max (SL, PT)

Kural 3: PT + Max (SL, 0)

MELES ile geliştirilmiş kurallar arasından seçilen bu üç kuralın literatürde karşılaştırma amacıyla kullanılan klasik kurallarla performans karşılaştırmaları dal-sınır algoritması ve genetik algoritma ile ulaşılabilmiş en iyi çözümü bulma sayıları olarak Çizelge 4.9 ve Çizelge 4.10'da, dal-sınır algoritması ve genetik algoritma ile ulaşılabilmiş en iyi çözümden sapma yüzdesi ortalamaları Çizelge 4.11.a ve Çizelge 4.12.a ve en büyük sapma olarak ise Çizelge 4.11.b ve Çizelge 4.12.b'de verilmiştir.

Çizelge 4.9: [1, 10] işlem süreli ikinci grup test problemleri için MELES öncelik kuralları ile klasik öncelik kurallarının eniyilediği problem sayısı karşılaştırması

	MELES Kuralları			**Klasik Kurallar**					
İş Sayısı	**Kural 1**	**Kural 2**	**Kural 3**	**SPT**	**EDD**	**MST**	**FIFO**	**MON**	**CR**
3	190(*)	158	197	122	117	52	50	53	45
4	183	140	194	86	97	39	16	23	17
5	171	119	195	55	85	24	22	8	9
6	169	115	188	39	79	33	5	4	4
7	165	91	177	18	69	27	2	1	0
8	166	99	179	12	66	29	2	0	0
9	169	93	174	9	65	30	1	3	3
10	165	82	184	7	67	33	0	1	1
12	155	76	171	1	57	25	0	0	0
15	157	78	169	4	48	33	0	0	0
20	157	66	172	1	51	35	0	0	0
30	142	63	144	1	51	36	0	0	0
40	143	70	133	0	46	36	0	0	0
50	132	61	137	0	48	40	0	0	0
75	137	57	136	0	50	41	0	0	0
100	140	63	103	1	48	40	0	0	0
150	137	74	105	0	50	45	0	0	0
200	126	70	110	0	52	42	0	0	0
250	130	69	110	0	53	40	0	0	0
500	149	74	87	0	52	44	0	0	0
750	140	78	92	0	53	41	0	0	0
1000	137	81	86	0	49	43	0	0	0
Toplam	3360(**)	1877	3243	356	1353	808	98	93	79

(*) Kural 1, iş sayısı 3 olan toplam 200 problemden 190 tanesini optimum çözmüştür.
(**) Kural 1, 3-1000 işli toplam 4400 problemden 3360 tanesini optimum çözmüştür.

Çizelge 4.10: [1, 100] işlem süreli ikinci grup test problemleri için MELES öncelik kuralları ile klasik öncelik kurallarının eniyilediği problem sayısı karşılaştırması

	MELES Kuralları			**Klasik Kurallar**					
İş Sayısı	**Kural 1**	**Kural 2**	**Kural 3**	**SPT**	**EDD**	**MST**	**FIFO**	**MON**	**CR**
3	172(*)	138	199	111	110	38	54	52	33
4	162	122	192	64	87	22	25	20	12
5	169	114	188	44	76	26	11	8	7
6	155	101	184	30	65	18	8	5	2
7	148	95	187	18	70	18	2	2	2
8	162	96	178	13	58	17	0	1	1
9	154	96	171	8	65	31	1	2	2
10	146	81	172	4	53	23	0	1	1
12	135	76	179	4	53	26	0	0	0
15	133	62	152	0	41	24	0	0	0
20	126	61	157	2	48	28	0	0	0
30	123	56	146	3	48	35	0	0	0
40	108	58	146	5	53	38	0	0	0
50	125	59	135	2	46	36	0	0	0
75	127	63	116	1	46	40	0	0	0
100	131	56	120	0	50	42	0	0	0
150	125	56	114	1	48	40	0	0	0
200	123	68	112	0	51	39	0	0	0
250	128	73	94	0	49	42	0	0	0
500	130	69	95	0	48	41	0	0	0
750	137	78	86	0	53	42	0	0	0
1000	135	79	91	0	52	45	0	0	0
Toplam	3054(**)	1757	3214	310	1270	711	101	91	60

(*) Kural 1, iş sayısı 3 olan toplam 200 problemden 172 tanesini optimum çözmüştür.
(**) Kural 1, 3-1000 işli toplam 4400 problemden 3054 tanesini optimum çözmüştür.

Çizelge 4.11.a: [1, 10] işlem süreli ikinci grup test problemleri için MELES öncelik kuralları ile klasik öncelik kurallarının optimumdan ortalama sapma karşılaştırması

	MELES Kuralları			**Klasik Kurallar**					
İş Sayısı	**Kural 1**	**Kural 2**	**Kural 3**	**SPT**	**EDD**	**MST**	**FIFO**	**MON**	**CR**
3	1,9(*)	15,4	0,3	33,7	15,6	55,7	64,1	65,1	61,5
4	1,0	16,5	0,4	67,4	16,9	63,1	116,8	109,9	107,2
5	2,0	14,2	0,5	85,3	20,9	72,7	126,2	153,6	143,2
6	2,1	12,7	0,6	88,2	25,8	73,3	167,2	180,4	178,9
7	1,8	23,5	1,6	153,2	25,1	85,7	267,9	270,6	261,5
8	1,2	11,2	0,8	134,0	28,1	90,2	225,6	277,9	279,7
9	1,8	12,0	0,9	137,2	33,9	92,3	225,5	334,7	339,1
10	1,5	10,8	0,6	142,7	33,5	82,6	285,5	307,7	312,8
12	1,3	11,5	1,2	146,2	33,8	79,5	232,0	323,3	323,1
15	2,2	10,9	1,2	320,4	41,4	80,2	500,4	546,9	543,1
20	0,6	8,1	0,7	390,2	43,2	85,7	666,3	990,9	989,9
30	0,6	5,3	0,7	577,1	42,6	75,4	1117,3	1528,6	1523,7
40	0,7	3,0	0,9	854,7	46,3	74,4	1653,6	1996,3	1993,4
50	0,4	8,6	0,6	2031,1	46,4	74,6	3533,2	4639,0	4635,0
75	0,4	7,5	0,4	1841,4	49,6	68,7	3375,7	6020,0	6016,4
100	0,5	7,7	0,5	4731,7	52,4	67,5	7656,4	10166,5	10163,5
150	0,2	2,2	0,4	11849,8	54,7	64,4	19695,1	28244,7	28242,2
200	0,3	3,2	0,2	15682,5	52,6	69,5	26652,4	43970,9	43967,3
250	1,1	8,5	0,2	25922,2	53,9	68,1	47406,7	64645,1	64642,6
500	0,2	2,5	0,1	103089,4	55,8	67,4	173984,9	270209,7	270207,2
750	0,4	2,0	0,1	228632,0	54,9	65,9	360403,5	578119,8	578116,3
1000	0,2	0,9	0,1	257938,4	57,0	64,7	438314,6	602525,2	602522,3
Genel Ort.	1,0(**)	9,0	0,6	29765,8	40,2	73,7	49394,1	73437,6	73435,0

(*) Kural 1, iş sayısı 3 olan toplam 200 problemde optimumdan ortalama %1,9 sapmıştır.
(**) Kural 1, 3-1000 işli toplam 4400 problemde optimumdan ortalama %1,0 sapmıştır.

Çizelge 4.11.b: [1, 10] işlem süreli ikinci grup test problemleri için MELES öncelik kuralları ile klasik öncelik kurallarının optimumdan en büyük sapma karşılaştırması

	MELES Kuralları			Klasik Kurallar					
İş Sayısı	**Kural 1**	**Kural 2**	**Kural 3**	**SPT**	**EDD**	**MST**	**FIFO**	**MON**	**CR**
3	100,0(*)	300,0	37,5	600,0	118,2	600,0	700,0	1000,0	700,0
4	33,3	300,0	28,6	1300,0	80,0	500,0	1500,0	1200,0	900,0
5	50,0	700,0	50,0	1500,0	105,0	700,0	1350,0	1900,0	1900,0
6	66,7	200,0	25,0	1700,0	133,3	500,0	2800,0	2200,0	2200,0
7	100,0	366,7	55,6	3300,0	100,0	600,0	4900,0	3200,0	3200,0
8	20,0	400,0	18,2	3000,0	106,7	1100,0	3900,0	7200,0	7200,0
9	100,0	250,0	18,8	2900,0	125,0	512,5	2800,0	3900,0	3900,0
10	100,0	160,0	21,9	3400,0	112,5	375,0	8900,0	7200,0	7200,0
12	30,0	169,2	26,1	2050,0	121,7	275,0	2700,0	3950,0	3950,0
15	100,0	200,0	47,1	17000,0	130,8	255,6	19300,0	11900,0	11900,0
20	21,4	200,0	15,3	10400,0	128,6	500,0	18800,0	21800,0	21800,0
30	33,3	93,5	10,0	15700,0	112,2	800,0	60000,0	83200,0	83200,0
40	25,0	80,0	21,1	18700,0	122,2	400,0	46400,0	44900,0	44900,0
50	12,5	400,0	12,7	86400,0	85,3	500,0	133200,0	152600,0	152600,0
75	25,0	400,0	6,7	54600,0	89,4	233,3	84200,0	236900,0	236900,0
100	25,0	333,3	10,0	383100,0	92,9	375,0	550500,0	508800,0	508800,0
150	9,2	100,0	15,4	807500,0	99,2	300,0	1200800,0	1358700,0	1358700,0
200	25,0	200,0	8,3	471200,0	85,7	450,0	818100,0	1266700,0	1266700,0
250	100,0	600,0	2,8	814500,0	82,2	375,0	1724200,0	2039400,0	2039400,0
500	20,0	200,0	1,5	4777100,0	100,0	350,0	7648600,0	7978350,0	7978350,0
750	45,5	100,0	3,4	9255350,0	81,4	460,0	15732850,0	16998700,0	16998700,0
1000	12,5	36,6	1,1	16086050,0	84,1	400,0	28802150,0	31160650,0	31160650,0
Genel Ort.	47,9(**)	263,2	19,9	1491697,7	104,4	480,1	2584938,6	2813379,5	2813352,3

(*) Kural 1, iş sayısı 3 olan 200 problemde optimumdan en büyük sapma ortalaması %100 olmuştur.
(**) Kural 1, 3-1000 işli 4400 problemde optimumdan en büyük sapma ortalaması %47,9 olmuştur.

Çizelge 4.12.a: [1, 100] işlem süreli ikinci grup test problemleri için MELES öncelik kuralları ile klasik öncelik kurallarının optimumdan ortalama sapma karşılaştırması

	MELES Kuralları			**Klasik Kurallar**					
İş Sayısı	**Kural 1**	**Kural 2**	**Kural 3**	**SPT**	**EDD**	**MST**	**FIFO**	**MON**	**CR**
3	2,1(*)	17,9	0,0	81,7	15,4	107,0	138,5	126,1	136,2
4	2,4	16,1	0,3	71,5	25,9	107,3	128,1	148,4	170,7
5	1,4	30,2	0,6	131,4	23,0	123,0	208,9	234,3	244,8
6	2,0	25,9	0,5	141,6	30,4	126,3	318,7	383,8	391,7
7	2,4	23,2	0,4	153,1	29,7	117,9	271,1	330,7	326,8
8	1,4	20,1	1,4	206,5	35,6	124,7	356,5	378,7	383,1
9	1,6	16,9	1,3	242,5	34,7	139,1	453,6	558,2	565,2
10	2,1	23,9	0,9	390,6	37,9	140,9	548,7	721,9	727,0
12	0,9	8,2	0,5	357,0	40,6	118,3	482,0	721,2	719,7
15	2,0	14,9	1,2	298,8	42,9	100,2	506,7	719,4	721,0
20	1,3	24,8	0,9	1477,8	46,2	132,3	2168,0	3235,7	3235,9
30	0,9	12,1	0,6	591,2	51,1	89,7	1207,6	1850,9	1849,7
40	1,1	8,9	0,8	1072,6	49,4	83,8	1878,2	3251,3	3250,4
50	0,8	4,3	0,7	4617,8	54,1	100,1	9032,3	10104,4	10103,3
75	0,3	5,6	0,6	5591,6	55,1	81,2	7723,6	15878,9	15876,2
100	7,8	10,9	0,4	31873,5	55,5	104,0	52186,5	87500,4	87497,4
150	0,5	42,2	0,4	20569,9	57,9	91,9	35616,0	72154,7	72151,1
200	0,6	6,4	0,2	21311,6	60,0	89,0	40932,9	67555,4	67553,3
250	0,3	6,7	0,3	50525,9	59,4	83,6	91033,4	181384,8	181381,6
500	0,6	9,7	0,4	127693,9	62,3	84,0	241814,8	460010,1	460006,9
750	0,7	7,9	0,1	775133,3	61,3	92,3	1406685,7	2151052,2	2151049,3
1000	0,3	9,8	0,1	3172862,1	60,9	88,7	5628839,9	8470996,1	8470992,9
Genel Ort.	1,5(**)	15,8	0,6	191608,9	45,0	105,7	341933,3	524059,0	524060,6

(*) Kural 1, iş sayısı 3 olan toplam 200 problemde optimumdan ortalama %2,1 sapmıştır.
(**) Kural 1, 3-1000 işli toplam 4400 problemde optimumdan ortalama %1,5 sapmıştır.

Çizelge 4.12.b: [1, 100] işlem süreli ikinci grup test problemleri için MELES öncelik kuralları ile klasik öncelik kurallarının optimumdan en büyük sapma karşılaştırması

	MELES Kuralları			Klasik Kurallar					
İş Sayısı	Kural 1	Kural 2	Kural 3	SPT	EDD	MST	FIFO	MON	CR
3	63,6(*)	414,3	2,1	5300,0	140,0	2800,0	4100,0	4100,0	4100,0
4	63,6	640,0	14,1	5300,0	163,4	2800,0	4100,0	4920,0	4920,0
5	63,6	1200,0	39,1	5300,0	163,4	2800,0	4100,0	4920,0	4920,0
6	63,6	1200,0	39,1	5300,0	207,5	2800,0	11600,0	11400,0	11400,0
7	100,0	1200,0	39,1	5966,7	207,5	2800,0	11600,0	11400,0	11400,0
8	100,0	1200,0	39,1	10350,0	207,5	2800,0	11600,0	11400,0	11400,0
9	100,0	1200,0	39,1	10350,0	207,5	2800,0	11600,0	11400,0	11400,0
10	100,0	1200,0	39,1	16300,0	207,5	2800,0	15250,0	18775,0	18775,0
12	35,5	150,0	18,4	13940,0	190,9	2650,0	13480,0	19800,0	19800,0
15	69,4	733,3	26,2	13500,0	160,2	552,6	24000,0	32366,7	32366,7
20	100,0	900,0	12,2	110500,0	154,9	3500,0	137500,0	127300,0	127300,0
30	30,0	508,3	8,3	16561,1	127,5	428,6	24300,0	45200,0	45200,0
40	51,3	216,7	24,0	49800,0	114,1	400,0	52575,0	131775,0	131775,0
50	26,4	150,0	7,9	258800,0	133,6	833,3	533533,3	491866,7	491866,7
75	10,4	520,0	9,2	611200,0	109,7	900,0	668500,0	1546200,0	1546200,0
100	800,0	800,0	9,0	3229300,0	109,0	3300,0	4789800,0	6053100,0	6053100,0
150	33,3	6100,0	5,1	2262200,0	96,1	2500,0	3661100,0	7536300,0	7536300,0
200	35,1	214,3	3,6	539400,0	93,8	1200,0	1137880,0	2613300,0	2613300,0
250	14,9	433,3	5,5	1839800,0	95,7	1466,7	3407766,7	6584400,0	6584400,0
500	69,4	933,3	22,7	5783733,3	107,9	933,3	9616133,3	22837400,0	22837400,0
750	57,1	840,0	1,8	49308633,3	92,2	2133,3	90297633,3	127884766,7	127884766,7
1000	22,2	600,0	0,9	299299000,0	93,9	2600,0	520695400,0	627657200,0	627657200,0
Genel Ort.	91,4(**)	970,6	18,4	16518206,1	144,7	2081,7	28869706,9	36529058,6	36529058,6

(*) Kural 1, iş sayısı 3 olan 200 problemde optimumdan en büyük sapma ortalaması %63,6 olmuştur.
(**) Kural 1, 3-1000 işli 4400 problemde optimumdan en büyük sapma ortalaması %91,4 olmuştur.

Bu üç kuralın performansları hem birinci hem de ikinci grup test problemlerinin çözdürülmesi ile elde edilen sonuçlar incelendiğinde, kuralların birbirini tamamlayan şekilde çalıştığı görülmüştür. Yani bu üç kuraldan en az biri dal-sınır algoritması ile bulunmuş eniyi değerleri problemlerin büyük çoğunluğunda bulabilmekte ve dal-sınır algoritması uygulanmamış problemlerin hepsinde klasik öncelik kurallarına eşit ya da daha iyi sonuçlar elde etmektedir. Bu üç kuralın grup olarak performansı eniyi çözümü bulma sayıları olarak Çizelge 4.13 ve Çizelge 4.14'de, bilinen eniyi çözümden ortalama sapma yüzde olarak Çizelge 4.15.a ve Çizelge 4.16.a'da, en büyük sapma yüzde olarak Çizelge 4.15.b ve Çizelge 4.16.b'de verilmiştir. Karşılaştırmalarda klasik öncelik kuralları da bir grup olarak ele alınmıştır.

Çizelge 4.13: [1, 10] işlem süreli ikinci grup test problemleri için MELES öncelik kuralları ile klasik öncelik kurallarının küme olarak eniyilediği problem sayısı karşılaştırması

İş Sayısı	{K1, K3}	{K1, K2, K3}	Klasik Kurallar
3	200(*)	200	184
4	198	198	150
5	200	200	120
6	197	200	108
7	192	200	80
8	197	200	74
9	192	200	71
10	198	200	70
12	192	199	58
15	196	199	51
20	194	199	52
30	187	199	52
40	189	200	46
50	187	200	48
75	191	200	50
100	182	199	49
150	178	200	50
200	177	200	52
250	179	200	53
500	177	200	52
750	174	200	53
1000	169	200	49
Toplam	4146(**)	4393	1572

(*) Kural 1 ve Kural 3'den oluşan küme, iş sayısı 3 olan 200 problemin 200'ünü optimum çözmüştür.
(**) Kural 1 ve Kural 3'den oluşan küme, 3-1000 işli 4400 problemin 4146'sını optimum çözmüştür.

Çizelge 4.14: [1, 100] işlem süreli ikinci grup test problemleri için MELES öncelik kuralları ile klasik öncelik kurallarının küme olarak eniyilediği problem sayısı karşılaştırması

İş Sayısı	**{K1, K3}**	**{K1, K2, K3}**	**Klasik Kurallar**
3	200(*)	200	182
4	197	197	138
5	198	199	108
6	193	194	91
7	198	200	81
8	194	196	66
9	196	197	69
10	197	200	53
12	197	198	56
15	192	200	41
20	193	199	49
30	190	197	51
40	183	195	58
50	187	198	48
75	184	199	47
100	193	200	50
150	187	199	49
200	180	200	51
250	171	200	49
500	174	200	48
750	171	200	53
1000	171	200	52
Toplam	4146(**)	4368	1490

(*) Kural 1 ve Kural 3'den oluşan küme, iş sayısı 3 olan 200 problemin 200'ünü optimum çözmüştür.
(**) Kural 1 ve Kural 3'den oluşan küme, 3-1000 işli 4400 problemin 4146'sını optimum çözmüştür.

Çizelge 4.15.a: [1, 10] işlem süreli ikinci grup test problemleri için MELES öncelik kuralları ile klasik öncelik kurallarının küme olarak optimumdan ortalama sapma karşılaştırması

İş Sayısı	{K1, K3}	{K1, K2, K3}	Klasik Kurallar
3	0,00(*)	0,00	1,66
4	0,07	0,07	3,27
5	0,00	0,00	7,48
6	0,18	0,18	8,05
7	0,36	0,34	11,38
8	0,06	0,04	12,66
9	0,23	0,13	15,25
10	0,04	0,02	15,79
12	0,24	0,01	17,36
15	0,41	0,05	22,26
20	0,10	0,00	25,03
30	0,06	0,01	24,97
40	0,15	0,00	29,92
50	0,08	0,00	29,57
75	0,05	0,00	31,65
100	0,08	0,02	34,03
150	0,04	0,00	35,21
200	0,05	0,00	33,85
250	0,04	0,00	35,61
500	0,02	0,00	37,19
750	0,02	0,00	36,70
1000	0,03	0,00	38,01
Ortalama	0,10(**)	0,04	23,04

(*) Kural 1 ve Kural 3'den oluşan küme, iş sayısı 3 olan toplam 200 problemde optimumdan ortalama %0,00 sapmıştır.
(**) Kural 1 ve Kural 3'den oluşan küme, 3-1000 işli toplam 4400 problemde optimumdan ortalama %0,10 sapmıştır.

Çizelge 4.15.b: [1, 10] işlem süreli ikinci grup test problemleri için MELES öncelik kuralları ile klasik öncelik kurallarının küme olarak optimumdan en büyük sapma karşılaştırması

İş Sayısı	{K1, K3}	{K1, K2, K3}	Klasik Kurallar
3	0,00(*)	0,00	50,00
4	8,33	8,33	42,86
5	0,00	0,00	70,00
6	16,67	16,67	50,00
7	22,22	22,22	90,00
8	3,33	3,33	56,52
9	10,71	10,71	111,11
10	3,66	3,66	85,00
12	12,50	2,25	78,85
15	47,06	8,40	110,00
20	4,50	0,46	92,86
30	3,85	0,82	94,44
40	12,70	0,00	122,22
50	6,08	0,00	83,89
75	1,81	0,00	83,59
100	3,88	3,88	92,86
150	1,10	0,00	99,25
200	1,53	0,00	85,72
250	2,80	0,00	82,24
500	0,47	0,00	100,00
750	1,00	0,00	81,44
1000	1,12	0,00	84,13
Ortalama	7,52(**)	3,67	83,95

(*) Kural 1 ve Kural 3'den oluşan kümenin, iş sayısı 3 olan toplam 200 problemde optimumdan en büyük sapma ortalaması %0,00 olmuştur.
(**) Kural 1 ve Kural 3'den oluşan kümenin, 3-1000 işli toplam 4400 problemde optimumdan en büyük sapma ortalaması %7,52 olmuştur.

Çizelge 4.16.a: [1, 100] işlem süreli ikinci grup test problemleri için MELES öncelik kuralları ile klasik öncelik kurallarının küme olarak optimumdan ortalama sapma karşılaştırması

İş Sayısı	{K1, K3}	{K1, K2, K3}	Klasik Kurallar
3	0,00(*)	0,00	0,76
4	0,05	0,05	4,00
5	0,00	0,00	6,49
6	0,11	0,08	9,07
7	0,03	0,00	11,08
8	0,19	0,15	13,41
9	0,32	0,21	13,72
10	0,03	0,00	16,11
12	1,38	0,00	18,12
15	3,10	0,00	22,50
20	2,11	0,00	25,18
30	1,39	0,06	29,56
40	1,82	0,03	28,71
50	1,37	0,03	30,55
75	0,85	0,00	33,91
100	8,10	0,00	35,07
150	0,84	0,02	36,72
200	0,70	0,00	38,15
250	0,55	0,00	37,62
500	0,93	0,00	40,72
750	0,73	0,00	39,43
1000	0,33	0,00	39,38
Ortalama	1,13(**)	0,03	24,10

(*) Kural 1 ve Kural 3'den oluşan küme, iş sayısı 3 olan toplam 200 problemde optimumdan ortalama %0,00 sapmıştır.
(**) Kural 1 ve Kural 3'den oluşan küme, 3-1000 işli toplam 4400 problemde optimumdan ortalama %1,13 sapmıştır.

Çizelge 4.16.b: [1, 100] işlem süreli ikinci grup test problemleri için MELES öncelik kuralları ile klasik öncelik kurallarının küme olarak optimumdan en büyük sapma karşılaştırması

İş Sayısı	{K1, K3}	{K1, K2, K3}	Klasik Kurallar
3	0,00(*)	0,00	23,23
4	5,70	5,70	64,58
5	0,47	0,23	64,58
6	8,00	8,00	82,93
7	3,26	0,00	114,67
8	9,80	9,80	114,67
9	23,33	23,33	114,67
10	3,31	0,00	114,67
12	35,48	0,37	98,61
15	69,44	0,00	93,09
20	100,00	0,21	99,28
30	30,00	3,55	109,48
40	51,28	2,78	103,46
50	26,42	3,66	114,87
75	10,39	0,54	95,53
100	800,00	0,00	99,25
150	33,33	2,64	96,13
200	35,14	0,00	93,83
250	14,89	0,00	95,67
500	69,44	0,00	107,89
750	57,14	0,00	92,20
1000	22,22	0,00	93,92
Ortalama	64,05(**)	2,76	94,87

(*) Kural 1 ve Kural 3'den oluşan kümenin, iş sayısı 3 olan toplam 200 problemde optimumdan en büyük sapma ortalaması %0,00 olmuştur.
(**) Kural 1 ve Kural 3'den oluşan kümenin, 3-1000 işli toplam 4400 problemde optimumdan en büyük sapma ortalaması %64,05 olmuştur.

Elde edilen sonuçlar incelendiğinde klasik öncelik kurallarının en iyi çözümü bazı durumlarda bulabildiği görülmüştür. En iyi çözümü bulan kurallar çoğunlukla EDD, SPT ve MST kurallarıdır. FIFO kuralı ender zamanlarda en iyi çözümü bulmuştur, fakat her durumda başka bir klasik öncelik kuralı da en iyi çözüme erişmiştir.

EDD klasik öncelik kuralının başarısı genel olarak RDD parametresinin azalması ile doğru orantılıdır. RDD parametresinin artması durumunda ise TF parametresinin orta değerleri için (0,4 – 0,6) en iyi çözümler bulunabilmekte, uç değerler için çözüme erişememektedir. Ayrıca EDD kuralı RDD parametresinin görece geniş olduğu (0,8) durumlarda en iyi çözüme ulaşamamaktadır. EDD kuralı genel olarak [1, 10] işlem süreli test problemlerinin % 30,75'inde (4400'de 1353), [1, 100] işlem süreli test problemlerinin ise % 28,86'sında (4400'de 1270) optimum çözüme ulaşmıştır. EDD kuralı optimum çözümü elde etme bakımından en iyi klasik öncelik kuralıdır.

MST klasik öncelik kuralının başarısı, RDD parametresinin azalırken aynı zamanda teslim zamanı aralığının arttığı durumlarda artmaktadır. Dolayısıyla MST kuralı teslim zamanı genişliğinin azaldığı ve beklenen geciken iş sayısının arttığı durumlarda başarılı olmaktadır. Buna karşın RDD parametresinin 0,6 ve 0,8 olduğu durumlarda başarı gösterememektedir. MST kuralı genel olarak [1, 10] işlem süreli test problemlerinin % 18,36'sında (4400'de 808), [1, 100] işlem süreli test problemlerinin ise % 16,16'sında (4400'de 711) optimum çözüme ulaşmıştır. Optimum çözüme ulaşma bakımından MST kuralı EDD kuralından sonra ikinci en iyi klasik öncelik kuralıdır.

SPT klasik öncelik kuralı ise genel olarak beklenen geciken iş sayısının az olduğu durumlarda başarılı olabilmektedir. SPT kuralı genel olarak [1, 10] işlem süreli test problemlerinin % 8,09'unda (4400'de 356), [1, 100] işlem süreli test problemlerinin ise % 7,04'ünda (4400'de 310) optimum çözüme ulaşmıştır.

Bütün klasik öncelik kuralları hem tek tek hem de küme olarak ele alındığında iş sayısı artarken test problemlerini optimum şekilde çözme sayısı oldukça azalırken, bu değerden sapma miktarları da artmaktadır.

Ele alınan bütün test problemleri için, MELES tarafından üretilen öncelik kuralları birinci test grubundaki ilgili test problemlerini optimum olarak çözmektedir. İlgili kümedeki her bir test problemi için MELES ile öncelik kuralı türetilmesi ile birinci test grubundaki her bir problem için optimum çözüm veren bir ya da daha fazla öncelik kuralı bulunmuştur ve %100 (40.000/40.000) çözüm başarısı sağlanmıştır. İlgili 40000 test probleminin en az bir tanesini optimum çözebilen 5.574.279 adet öncelik kuralı MELES tarafından geliştirilmiştir. Bu öncelik kurallarından 8142 adeti en az 400 adet test problemini çözebilmektedir ve aday öncelik kuralı olarak seçilmiştir. Aday öncelik kuralları kümesindeki kurallar daha fazla problem çeşitliliğine sahip ikinci test grubu problemleri üzerinde denenmiştir. Bu işlem sonucunda 3 adet kuralın (Kural 1: Max (SL, PT) + PT + Max (SL, 0); Kural 2: Max (SL, PT); Kural 3: PT + Max (SL, 0)) çok başarılı sonuçlar verdiği görülmüştür. [1, 10] dağılımlı işlem süresine sahip ikinci grup test problemlerinde Kural 1 %84 (3360/4400), Kural 2 %42,65 (1877/4400) ve Kural 3 %73,70 (3243/4400) başarı sağlamıştır. Bu grupta literatürde karşılaştırma için kullanılan öncelik kurallarından en iyi üçü EDD %30,75 (1353/4400), MST %18,36 (808/4400) ve SPT %8,09 (356/4400) başarı sağlayabilmiştir. Daha zor olan [1, 100] dağılımlı işlem süresine sahip ikinci grup test problemlerinde ise Kural 1 %69,41 (3054/4400), Kural 2 %39,93 (1757/4400) ve Kural 3 %73,04 (3214/4400) başarı sağlamıştır. İlgili grupta literatürde karşılaştırma için kullanılan öncelik kurallarından en iyi üçü EDD %28,86 (1270/4400), MST %16,16 (711/4400) ve SPT %7,04 (310/4400) başarı sağlayabilmiştir. [1, 10] dağılımlı işlem süresine sahip ikinci grup test problemlerinde bilinen en iyi değerden ortalama sapmalara bakıldığında, Kural 1 %1,0, Kural 2 %9,0 ve Kural 3 %0,6 ortalama sapma değerine sahipken EDD %40,2, MST %73,7 ve SPT %29.765,8 ortalama sapmaya sahiptir. Daha zor olan [1, 100] dağılımlı işlem süresine sahip ikinci grup test problemlerinde bilinen en iyi değerden ortalama sapmalara bakıldığında, Kural 1 %47,9, Kural 2 %263,2 ve Kural 3 %19,9 ortalama sapma değerine sahipken EDD %104,4, MST %480,1 ve SPT %1.491.697,7 ortalama sapmaya sahiptir. Dolayısıyla MELES tarafından geliştirilen aday öncelik kuralları arasından seçilen üç adet öncelik kuralı, literatürde karşılaştırma amacıyla kullanılan öncelik kurallarından problemlerin bilinen en iyi çözümlerine ulaşma sayısı bağlamında çok daha iyi sonuç vermektedir. MELES tarafından geliştirilen üç kural

problemdeki çizelgelenecek olan iş sayısının artışı durumundan klasik öncelik kurallarının tersine çok daha az etkilenmektedir.

Yapılan incelemeler sonucunda toplam gecikme performans ölçütü göz önüne alındığında MELES tarafından geliştirilen bu üç kuralın birbirini tamamlayan bir yapısı olduğu görülmüştür. Yani bu kurallar ortak çalışmaları sonucunda çok daha fazla problemi optimum şekilde çözebilmektedirler. [1, 10] dağılımlı işlem süresine sahip ikinci grup test problemlerinde {Kural 1, Kural 2, Kural 3} kümesi %99,84 (4393/4400) başarı sağlamış, yani problemlerin bilinen eniyi çözümlerine ulaşmıştır. İlgili test problemleri grubunda klasik öncelik kurallarının hepsinin oluşturduğu küme {SPT, EDD, MST, FIFO, MON, CR} ise %35,73 (1572/4400) başarı sağlayabilmektedir. Daha zor olan [1, 100] dağılımlı işlem süresine sahip ikinci grup test problemlerinde ise {K1, K2, K3} kümesi %99,27 (4368/4400) başarı sağlamış, klasik öncelik kuralları kümesi ise %33,86 (1490/4400) başarı yakalayabilmiştir. Bilinen en iyi çözümden ortalama sapmalar bazında ise [1, 10] dağılımlı işlem süresine sahip ikinci grup test problemlerinde {K1, K2, K3} kümesi sadece %0,04 ortalama sapmaya sahipken klasik öncelik kuralları kümesi %23,04'lük bir ortalama sapmaya sahiptir. Daha zor olan [1, 100] dağılımlı işlem süresine sahip ikinci grup test problemlerinde {K1, K2, K3} kümesi %0,03 ortalama sapmaya sahiptir. Buna karşın klasik öncelik kuralları kümesi %24,10'lık bir ortalama sapma ile çalışmışlardır. Problemlerdeki iş sayılarının artması nedeniyle {K1, K2, K3} kümesinin performansında herhangi bir değişiklik olmazken, klasik öncelik kuralları kümesinin performansı hem [1, 10] hem de [1, 100] dağılımlı problemlerde ortalama %72 kötüleşmektedir.

BÖLÜM 5

ÇİZELGELEME ALGORİTMALARININ GENETİK PROGRAMLAMA İLE KEŞFİ

Bu bölümde üretim sistemlerinde çizelgeleme problemlerinin çözümünde yararlanılacak çizelgeleme algoritmalarının geliştirilmesi için kullanılacak olan Çoklu İfade Programlama Algoritma Öğrenme Sistemi (MEPAL) olarak adlandırdığımız yaklaşımımız, yapılandırılması, uygulamaları ve sonuçları ile ilgili bilgi verilecektir. MEPAL bütün üretim sistemlerine uygulanabilecek bir teknik olmasına rağmen uygulamalarımız literatürde kullanılan OR Library test problemleri üzerinde yapılmıştır.

***Algoritmalar ve Algoritmik Problem Çözme**:*

Algoritma, bir işi yapmak için tanımlanan, bir başlangıç durumundan başladığında, açıkça belirlenmiş bir son durumunda sonlanan ve bir çıktı üreten, sonlu işlemler kümesidir. Algoritma kelimesi bu konuda ilk çalışmaları yapan El-Harezmi'nin isminden gelmektedir. El-Harezmi (d:780, ö:850), algoritmayı, doğrusal ve ikili denklemlerin çözülebilmesi için izlenmesi gereken aritmetik işlemler olarak tarif etmiştir. Günümüzdeki şekli ve tanımına ise David Hilbert'in 1928 yılındaki "Karar Problemi" (Entscheidungsproblem) probleminin çözümü için uygulanması yoluyla gelmiştir.

Algoritma genel olarak bir problemin çözümü için uygulanması gereken işlem adımlarını tarif eden yapıdır. Aynı zamanda, bir algoritma tekrarlanması halinde tasarlandığı amacı yerine getiren bir veya bir çok adımın oluşturduğu bir ifade biçimidir. Adım, kendi içinde anlamlı ve bağımsız işlemlerin oluşturduğu ve işlev

bütünlüğü taşıyan en küçük algoritma birimidir. İterasyon algoritmanın tekrarlanması olup, iterasyon sayısı bir algoritmanın ana adımlarının problemin çözümü için kaç kez tekrarlandığının bir sayacıdır. Bir algoritma Church-Turing tezine göre dört ana bileşen tipinde oluşur. Bunlar koşullu ilerle (conditional Goto), koşulsuz ilerle (unconditional Goto), atama (assignment) ve dur (halt)'dur. Her algoritma bu bileşenler ya da bu bileşenlerin temsil ettiği operatörler vasıtasıyla oluşturulabilir.

Algoritmalar çeşitli şekillerde sınıflandırılabilir. Bu sınıflandırmalar genelde uygulama şekline göre (deterministik, paralel, dağıtık, yaklaşık sonuç veren, kuantum vb.), metodolojisine göre (sayımlama yapan, böl-yönet teknikli, indirgeme yapan vb.), kullanım alanına göre (çizelgeleme, arama, sıralama, şifreleme, veri sıkıştırma vb.) ve karmaşıklığına göre (doğrusal, üssel vb.) yapılmaktadır. Bunlara ek olarak farklı sınıflandırma şekilleri de kullanılabilmektedir (Burgin, 2004; Turing, 1939).

***Çizelgeleme Algoritmaları**:*

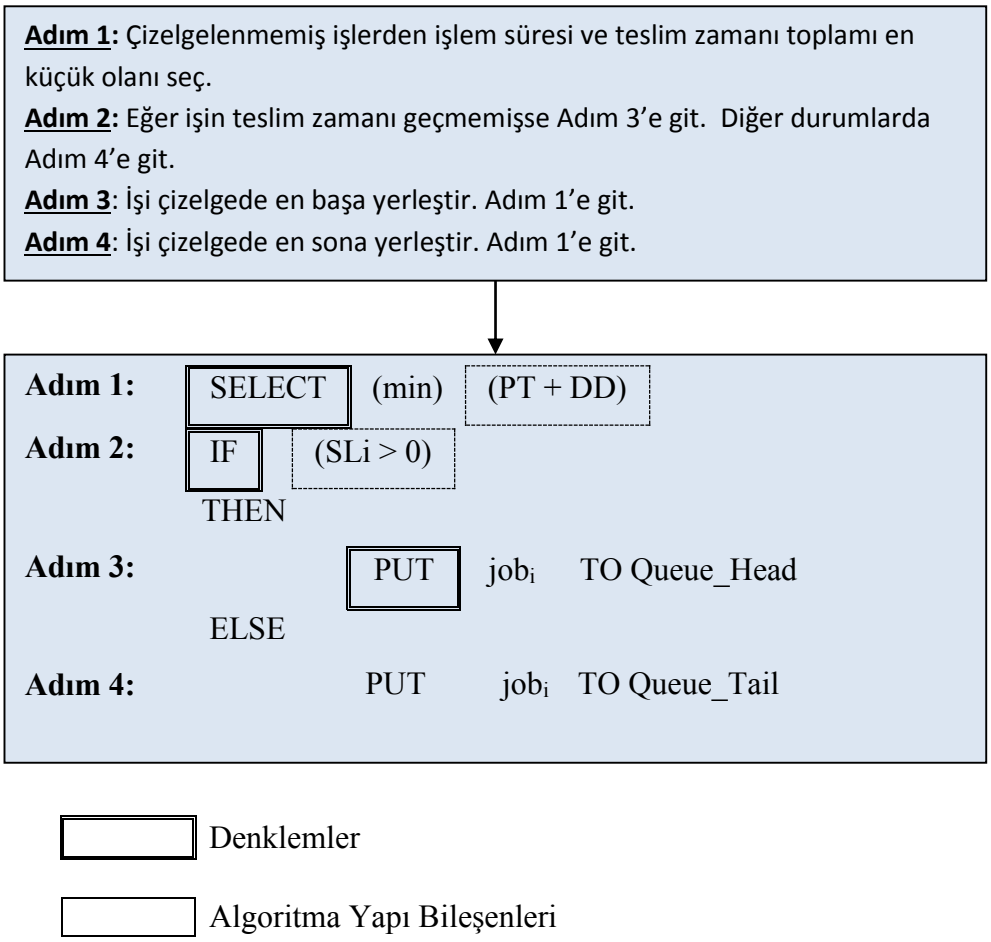
Çizelgeleme işleminde işlerin tezgahlardaki işlem sıraları bulunmaya çalışıldığından, bir çizelgeleme algoritması, performans ölçütünü göz önüne alarak işlerin tezgahlardaki işlem sıralarını belirlemek için gerçekleştirilecek bir dizi işlem olarak tanımlanabilir. MEPAL ile tezgah kuyruğundaki işlerin sırasını belirleyen bir algoritma tasarlanmaktadır.

Çizelgeleme algoritmaları verilen bir iş listesinin belirlenmiş bir değerlendirme ölçütüne uygun olarak sıralanması problemini çözmeye yönelik algoritmalardır. Bu bağlamda, çizelgeleme kuralları da aynı işlevi yerine getiren özel basit ve genellikle tek adımlık algoritmik yapılar olarak tanımlanabilir. Çizelgeleme algoritmalarının avantajlarından biri, daha önceki bölümlerde üzerinde tartışılmış ve geliştirme işlemleri yapılmış öncelik kurallarını da içerisinde barındırabilmesidir. Başka bir ifade ile her öncelik kuralı tek adımlık bir çizelgeleme algoritması olarak düşünülebilir. Çizelgeleme algoritmalarının bir diğer avantajı öncelik kurallarının sahip olduğu dezavantajları büyük oranda tersine çevirebilmesidir. Birçok durumda bir öncelik kuralı aynı problem parametrelerine sahip problemlerde bile farklı iyilikte sonuçlar

verebilmektedir. Sonuçların farklı iyiliklerde olması bazı zamanlarda bir kural-kümesi kullanılarak azaltılabilmekte ya da yok edilebilmektedir. Bu duruma en iyi çözüm ise hangi zamanlarda hangi işlemlerin yapılması gerektiğinin bulunabilmesidir.

Bugüne kadar oluşturulmuş olan çizelgeleme algoritmaları (Moore algoritması, Johnsson algoritması vb.) araştırmacıların akademik çabaları sonucu belirlenmiştir. Yine birçok algoritma, problemlerin özel durumları için akademik çalışmaları sonucu geliştirilmiştir. Algoritma oluşturmanın otomatikleştirilmesi bu tür özel durumların bulunabilmesini, yapılan işlemlerin nedenlerinin belirlenebilmesini, yeterince araştırılmamış birçok problem türünün incelenebilmesini ve bütün bu durumlar için algoritma geliştirilebilmesini ya da geliştirme sürecinin hızlanmasını sağlayacaktır.

Çalışma kapsamında MEPAL ile geliştirilecek olan algoritmalara bir örnek ve algoritmanın bileşenleri aşağıda verilmiştir (Şekil 5.1).



Şekil 5.1: Örnek bir algoritma ve bileşenleri

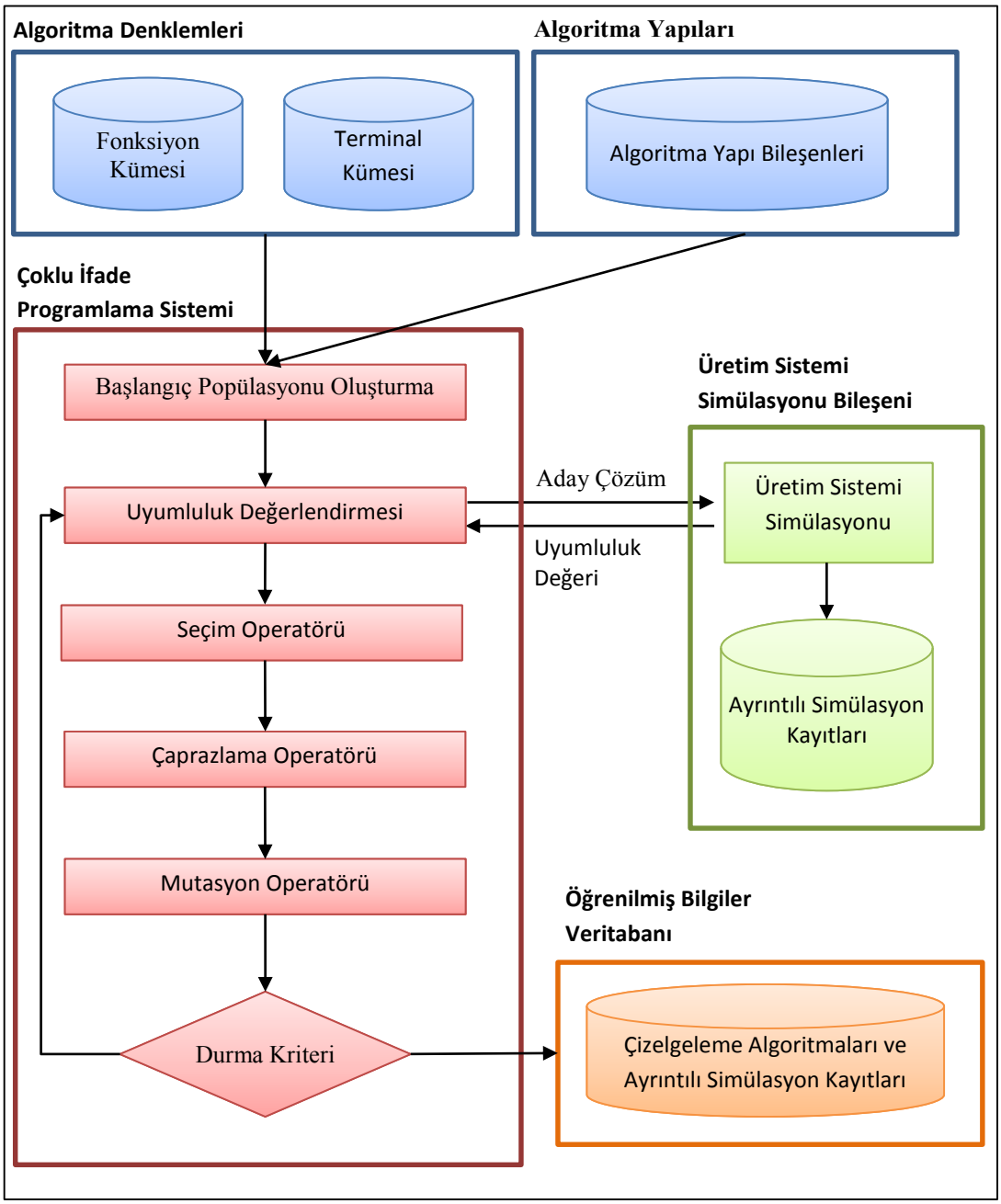
5.1 Çoklu İfade Programlama Algoritma Öğrenme Sistemi (MEPAL)

Çoklu ifade programlama tabanlı algoritma öğrenme sistemi (MEPAL) üretim sistemlerinde çizelgeleme problemlerinin çözümünde kullanılacak olan çizelgeleme algoritmalarını sadece standart bilgi ve bilgi yapılarını kullanarak oluşturan bir öğrenme sistemidir. Çizelgeleme algoritmalarının oluşturulması sırasında bir ya da daha fazla amacı göz önüne alarak yeni bilgi yapı ve akışlarını geliştirir ve bu işlem sırasında ve sonucunda aldığı ara karar ve bilgileri saklayarak ileri aşamalarda tekrar kullanabilir.

MEPAL amaçladığı bilgileri, bu bölüm kapsamında çizelgeleme algoritmalarını, bulabilmek amacıyla çeşitli alt sistemlerden oluşmaktadır. Bu alt sistemler aşağıdaki gibidir:

a) Algoritma Denklemleri Bileşeni:
   Çoklu ifade programlama bileşeni tarafından geliştirilen çizelgeleme algoritmaları içerisinde kullanılan denklemlerin yapı taşlarını barındırmaktadır. Bu denklemler algoritmanın işleyişi sırasında kullanılmaktadır.
b) Algoritma Yapıları Bileşeni:
   Çoklu ifade programlama bileşeni tarafından geliştirilen çizelgeleme algoritmaları içerisinde kullanılan SELECT, PUT, IF THEN ELSE gibi yapıları barındırmaktadır. Bu yapılar algoritmanın işleyişini sağlamaktadır.
c) Çoklu İfade Programlama Bileşeni:
   Seçim, çaprazlama ve mutasyon genetik operatörlerini kullanarak çözüm adaylarını belirlenmiş durma kriteri sağlanana kadar iyileştirmeye çalışan sistemdir.
d) Üretim Sistemi Simülasyonu Bileşeni:
   Çoklu ifade programlama sisteminin üzerinde çalıştığı çözüm adaylarının ele alınan problemi ne kadar iyi çözebildiğini gösteren uyumluluk değerlerini belirlemek için kullanılan deterministik simülasyon yapan bileşendir.
e) Öğrenilmiş Bilgiler Veritabanı:
   Çoklu ifade programlama tabanlı algoritma öğrenme sisteminin çalışması sırasında ve sonucunda oluşturulan algoritmaları, algoritmaların bileşenlerini ve simülasyon içindeki çizelgeleme kararlarını barındıran veritabanıdır.

İlgili alt sistemler ve ilişkileri Şekil 5.2'de gösterilmiştir.



Şekil 5.2: MEPAL genel yapısı ve alt sistemleri

Bu çalışma kapsamında MEPAL ile üretim sistemlerinde çizelgeleme problemlerinin çözümünde kullanılacak olan çizelgeleme algoritmalarının geliştirilmesi için yapılandırılması izleyen kısımda verilmiştir.

5.1.1 Çizelgeleme algoritması bileşenleri

Çizelgeleme algoritmaları, algoritma yapı bileşenleri ve denklemlerden oluşmaktadır. Algoritma yapı bileşenleri algoritmanın akışını sağlarken, denklemler durum/akış kontrolünde kullanılarak yapı bileşenlerinin çalışma biçimini belirlemektedir. Algoritma denklemlerinin oluşturulması MELES ile öncelik kuralı oluşturma işlemine benzer şekilde yapılmaktadır. MEPAL içinde algoritmaların oluşturulması için daha önce kullanılan çoklu ifade genetik programlama tekniği kullanılmaktadır. Çizelgeleme algoritmalarını oluşturan algoritma yapı bileşenleri ve denklemlerin yapı taşları kullanıcılar tarafından problemin yapısına ve amaçlarımıza bağlı olarak belirlenir.

5.1.1.1 Algoritma denklemleri

Algoritma denklemlerinin oluşturulması sırasında MELES'e benzer biçimde terminal ve fonksiyon kümeleri kullanılmaktadır. Terminal kümesinde bulunan terminal öğeleri, problemin tanımlanmasında ve çözülmesinde doğrudan kullanılacak olan değişken, parametre ve sabitlerdir. Terminal kümesi öğeleri işlerin hem tek tek hem de grup halinde bazı bilgilerini yansıtabilecek şekilde belirlenmiştir. Fonksiyon kümesindeki öğeler ise terminal öğelerini belli bir işlem mantığına göre birleştirmeye yararlar ve en basit halleri aritmetik operatörlerdir. Bu çalışmada kullanılacak olan fonksiyon ve terminal kümelerinin öğeleri mevcut algoritmalar ve problem yapısı incelenerek belirlenmiş ve izleyen Çizelge 5.1, Çizelge 5.2.a ve Çizelge 5.2.b'de verilmiştir.

Çizelge 5.1: Fonksiyon kümesi

Fonksiyon Adı	Tanım	Formülasyon
ADD	İkili toplama	a + b
SUB	İkili çıkarma	a – b
MUL	İkili çarpma	a * b
DIV	İkili bölme	a/b
POS	Pozitif durum	max {a, 0}
NEG	Negatif durum	– a
POW	Üs alma	$\|a\|^b$
EXP	Üstel	e^a
MIN	En küçük	min (a, b)
MAX	En büyük	max (a, b)
AVE	Ortalama	(a + b) / 2
SQRT	Karekök	$SQRT(a) = \begin{cases} 1 & , eğer\ a < 0 \\ \sqrt{a} & d.d. \end{cases}$
ABS	Mutlak Değer	$\|a\|$

Çizelge 5.2.a: Terminal kümesi (Tek Tezgah)

Terminal Adları	Tanım
PT	İşlem süresi
DD	Teslim zamanı
RT	İşin sisteme geliş zamanı
SL	İşin bolluğu
CT	İşin tamamlanma zamanı
Value	[1, 10] arasında rassal bir sayı
Nj(Q)	Q kuyruğundaki işlerin sayısı
SumPT(Q), AvePT(Q), MinPT(Q), MaxPT(Q)	Q kuyruğundaki işlerin toplam, ortalama, en küçük ve en büyük işlem süresi
SumDD(Q), AveDD(Q), MinDD(Q), MaxDD(Q)	Q kuyruğundaki işlerin toplam, ortalama, en küçük ve en büyük teslim zamanı
SumSL(Q), AveSL(Q), MinSL(Q), MaxSL(Q)	Q kuyruğundaki işlerin toplam, ortalama, en küçük ve en büyük bollukları
SumCT(Q), AveCT(Q), MinCT(Q), MaxCT(Q)	Q kuyruğundaki işlerin toplam, ortalama, en küçük ve en büyük tamamlanma zamanları
SumRT(Q), AveRT(Q), MinRT(Q), MaxRT(Q)	Q kuyruğundaki işlerin toplam, ortalama, en küçük ve en büyük sisteme geliş zamanları
Index_SelectedJob	Seçilen işin kuyruktaki sırası
IndexG_SelectedJob	Seçilen işin indisi

Çizelge 5.2.b: Terminal kümesi (İki tezgah)

Terminal Adları	Tanım
PT(j)	İşin j tezgahındaki işlem süresi
DD	Teslim zamanı
RT	İşin sisteme geliş zamanı
SL(j)	İşin j tezgahındaki bolluğu
CT	İşin tamamlanma zamanı
Value	[1, 10] arasında rassal bir sayı
Nj(Q)	Q kuyruğundaki işlerin toplam sayısı
SumPT(Q), AvePT(Q), MinPT(Q), MaxPT(Q)	Q kuyruğundaki işlerin toplam, ortalama, en küçük ve en büyük işlem süresi
SumDD(Q), AveDD(Q), MinDD(Q), MaxDD(Q)	Q kuyruğundaki işlerin toplam, ortalama, en küçük ve en büyük teslim zamanı
SumSL(Q), AveSL(Q), MinSL(Q), MaxSL(Q)	Q kuyruğundaki işlerin toplam, ortalama, en küçük ve en büyük bollukları
SumCT(Q), AveCT(Q), MinCT(Q), MaxCT(Q)	Q kuyruğundaki işlerin toplam, ortalama, en küçük ve en büyük tamamlanma zamanları
SumRT(Q), AveRT(Q), MinRT(Q), MaxRT(Q)	Q kuyruğundaki işlerin toplam, ortalama, en küçük ve en büyük sisteme geliş zamanları
Sum(PT), Ave(PT), Min(PT), Max(PT)	İlgili işin toplam, ortalama, en küçük ve en büyük işlem süresi
Index_MinPT, Index_MaxPT	İlgili işin en küçük ve en büyük işlem süresinin olduğu tezgah indisi
Index_SelectedJob	Seçilen işin kuyruktaki sırası
IndexG_SelectedJob	Seçilen işin indisi

Terminal kümesi öğelerinin bazılarında indis olarak Q ifadesi bulunmaktadır. Q indisi hem tezgahın fiziksel kuyruğunu hem de algoritma tarafından oluşturulabilen sanal kuyrukları ifade etmektedir. Bu sanal kuyruklar algoritma tarafından işlerin belirli mantıklarda gruplanabilmesi için kullanılmaktadır.

5.1.1.2 Algoritma yapı bileşenleri

Algoritma yapı bileşenleri bir çizelgeleme algoritmasının oluşturulmasında ve akışının sağlanmasında kullanılan yapıtaşlarıdır. Bu yapıtaşları mevcut algoritmaların ve problem yapılarının incelenmesi sonucu belirlenmiştir.

SELECT Operatörü:

SELECT operatörü bir kuyrukta bulunan işlerden ölçüt denklemini kullanarak bir iş seçimi yapmaktadır. İş kümesi tezgahın fiziksel kuyruğu olabileceği gibi algoritma tarafından oluşturulan sanal bir kuyrukta olabilir. Algoritma içerisinde yeni bir SELECT operatörü çalışana kadar, seçilmiş olan iş kuyruk içerisinde konum değiştirse ya da yeni bir kuyruğa aktarılsa bile hala seçilmiş durumda kalmaktadır. Genel yapısı aşağıdaki gibidir.

SELECT	Min/Max	Ölçüt Denklemi	Kuyruk_ID

SELECT operatörü Kuyruk_ID ile belirtilen kuyruktaki işlere Ölçüt Denklemi ile değer atamaktadır. Değer atanan işler arasından Min/Max'a bağlı olarak en küçük ya da en büyük değerli işi seçmektedir.

Örnek:

SELECT	Min	PT	Kuyruk_1

Yukarıda verilen SELECT operatörü; Kuyruk_1'dan en küçük (Min) işlem süreli (PT) değerli işi seçmektedir.

PUT Operatörü:

PUT operatörü SELECT operatörü vasıtasıyla daha önceki adımlarda seçilmiş olan işi belirtilen bir kuyruktaki belirli bir pozisyona atamaktadır. Genel yapıları aşağıdaki gibidir.

PUT	**Kuyruk_ID**	**Kuyruk Pozisyonu (First/Last)**

PUT operatörü daha önce SELECT ile seçilmiş olan bir işi, Kuyruk_ID ile belirtilen kuyruğun başına ya da sonuna (Kuyruk Pozisyonu) eklemektedir.

PUT	**Job**	**Kuyruk_ID**	**Kuyruk Pozisyonu (First/Last)**

PUT operatörü ayrıca daha önce seçilmiş ve Job değişkenine atanmış olan bir işi Kuyruk_ID ile belirtilen kuyruğun başına ya da sonuna eklenmektedir.

Örnek:

PUT	Kuyruk_1	Last

Yukarıda belirtilen PUT operatörü, daha önce SELECT ile seçilmiş olan işi Kuyruk_1'deki işlerin en sonuna yerleştirmektedir.

SORT Operatörü:

SORT operatörü bir kuyruktaki işleri ölçüt denklemine göre küçükten büyüğe (AZ) ya da büyükten küçüğe (ZA) sıralama işlemi yapmaktadır. Genel yapısı aşağıdaki gibidir.

SORT	**AZ / ZA**	**Ölçüt Denklemi**	**Kuyruk_ID**

SORT operatörü ölçüt denklemini kullanarak kuyrukta (Kuyruk_ID) bulunan bütün işlere bir değer atamaktadır. Daha sonra işlerin atanmış değerlerine göre büyükten küçüğe (ZA) ya da küçükten büyüğe (AZ) işleri sıralamaktadır.

Örnek:

SORT	AZ	DD	Kuyruk_2

Yukarıda belirtilen SORT operatörü ile Kuyruk_2'deki bütün işler teslim zamanlarına (DD) göre küçükten büyüğe (AZ) sıralanmaktadır.

IF – THEN – ELSE Operatörü:

IF-THEN-ELSE operatörü içerisindeki ölçütler yardımıyla karşılaştırma yaparak karşılaştırma sonucuna göre algoritmayı yönlendirmektedir. Karşılaştırmanın doğruluğunun sağlandığı durumda THEN altındaki algoritma yapı bileşenleri, sağlanmadığı durumda ELSE altındaki algoritma yapı bileşenleri işletilir. Genel yapısı aşağıdaki gibidir.

IF	**Ölçüt Denklemi 1**	**Operatör** {<, >, ≤, ≥, ==, !=}	**Ölçüt Denklemi 2**	**THEN**
				ELSE

IF-THEN-ELSE operatörü, ölçüt denklemi 1 ve ölçüt denklemi 2 sonuçlarını ilgili operatörler (<, >, ≤, ≥, ==, !=) yolu ile karşılaştırmaktadır. Karşılaştırma sonucunun doğru olması durumunda THEN, karşılaştırma sonucunun yanlış olması durumunda ELSE altında bulunan algoritma adımları izlenmektedir. Karşılaştırma operatörlerinin anlamları Çizelge 5.3'de verilmiştir.

Çizelge 5.3: Karşılaştırma operatörleri ve anlamları

Karşılaştırma Operatörü	Anlamı
<	Küçük
>	Büyük
≤	Küçük eşit
≥	Büyük eşit
==	Eşit
!=	Eşit değil

Örnek:

IF	DD	<	AvePT(2)

Yukarıda verilen IF-THEN-ELSE operatörü, seçilmiş olan işin teslim zamanı ile 2. Kuyruktaki işlerin ortalama işlem süresini karşılaştırmaktadır. Eğer seçilmiş olan işin teslim zamanı 2. kuyruktaki işlerin ortalama işlem sürelerinden küçükse THEN , küçük değilse ELSE altındaki algoritma adımları izlenecektir.

APPEND Operatörü:

APPEND operatörü sistemdeki fiziksel ve sanal kuyrukları (iş kümelerini) birleştirmek için kullanılmaktadır. Genel yapısı aşağıdaki gibidir.

APPEND	**Kuyruk_ID_1**	**Kuyruk_ID_2**

APPEND	**ALL**

APPEND operatörü, iki kuyruk parametresine sahip olduğu durumda Kuyruk_ID_1 ve Kuyruk_ID_2 kuyruklarını belirtilen sıra ile birleştirmektedir. Diğer durumda sistemde yer alan bütün kuyrukları (fiziksel ve sanal) kuyruklarının indis numaraları küçükten büyüğe olacak biçimde birleştirmektedir.

Örnek:

APPEND	Kuyruk_3	Kuyruk_1

Yukarıda verilen APPEND operatörü Kuyruk_3 ve Kuyruk_1'deki bütün işleri Kuyruk_3 içerisinde toplamaktadır. Birleştirme işlemi öncesinde Kuyruk_3 içerisinde bulunan işler birleştirme işlemi sonrasında ön sırada bulunmaktadır.

ASSUME Operatörü:

ASSUME operatörü seçilen bir işin herhangi bir kuyruğun (fiziksel ya da sanal) başında ya da sonunda olduğu varsayımına göre seçilen işin tamamlanma zamanı, bolluk gibi çeşitli bilgilerinin hesaplanmasına yardımcı olmaktadır. Genel yapısı aşağıdaki gibidir.

ASSUME	**SELECT (…)**	**Kuyruk_ID**	**First / Last**

ASSUME operatörü, kendi içerisindeki SELECT operatörü ile seçilen işin Kuyruk_ID ile belirtilen kuyruğun başında ya da sonunda (First / Last) olduğu varsayımına göre ilgili iş için hesaplanabilir bütün bilgilerin hesaplanmasında kullanılır. İçerisindeki SELECT operatörü daha önce açıklanmış olan SELECT operatörü ile aynı parametrelere sahiptir ve aynı şekilde iş seçme işlemi yapmaktadır.

OBJ_VALUE Operatörü:

OBJ_VALUE operatörü seçilen bir kuyrukta bulunan işler için çizelgeleme probleminin çözümünde kullanılan performans ölçütü (toplam gecikme, geciken işlerin toplam sayısı vb.) değerini hesaplamaktadır. Genel yapısı aşağıdaki gibidir.

OBJ_VALUE	Kuyruk_ID

OBJ_VALUE operatörü Kuyruk_ID ile seçilmiş olan kuyruktaki tüm işlerin aynı sıra ile üretim sisteminde (tek tezgah, iki tezgah vb.) işlenmesi durumunda elde edilecek olan performans ölçütü değerini hesaplamaktadır. Bu hesaplama sırasında eğer varsa diğer kuyruklardaki işler göz ardı edilir. OBJ_VALUE ile hesaplama işlemi yapılan kuyruktaki işler hesaplama ardından çizelgelenmiş ya da bir daha işlem görmeyecek işler sınıfına girmezler ve işlem sonunda kuyruktaki işler içerisinde bir azalma olmaz.

FOREACH_JOB Operatörü:

FOREACH_JOB operatörü bir kuyruktaki tüm işler için belirtilen işlemlerin yapılmasını sağlamaktadır. Genel yapısı aşağıdaki gibidir.

FOREACH_JOB	Kuyruk_ID

FOREACH_JOB operatörü Kuyruk_ID ile belirtilen kuyruk içerisindeki bütün işler için ya da kendi içerisindeki işlemlerin yapısına bağlı olarak kuyruk içerisindeki bütün işlerin sayısı kadar belirli işlemlerin yapılmasını sağlar. Kuyruk_ID ile belirtilen kuyruk fiziksel ya da sanal bir kuyruk olabilir.

FOREACH_QUEUE Operatörü:

FOREACH_QUEUE operatörü verilen tüm kuyruklar için belirtilen işlemlerin yapılmasını sağlamaktadır. Genel yapısı aşağıdaki gibidir.

FOREACH_QUEUE	Kuyruk Listesi

FOREACH_QUEUE operatöründe bulunan kuyruk listesinde sadece bir kuyruk (ör: Kuyruk_2) bulunabileceği gibi sıralı olan (ör: Kuyruk_1, Kuyruk_2, Kuyruk_3) ya da olmayan (ör: Kuyruk_1, Kuyruk_4) bir kuyruk listesi ya da sistemdeki bütün fiziksel ve sanal kuyruklar (ALL) bulunabilir. Bu operatör belirtilen kuyruklar için ya da kendi içerisindeki işlemlerin yapısına bağlı olarak kuyrukların sayısı kadar belirli işlemlerin yapılmasını sağlar. Kuyruk listesindeki kuyruklar fiziksel ya da sanal kuyruk olabilir.

DIFF Operatörü:

DIFF operatörü belirtilen iş ya da kuyrukların karşılaştırılmasında ve verilen parametreye göre hangisinin iyi olduğunun bulunmasında kullanılır. Genel yapıları aşağıdaki gibidir.

DIFF	İş listesi	Karşılaştırma Parametresi	Min / Max

DIFF operatörü ile işler üzerinde karşılaştırma yapıldığı durumlarda kullanılan iş listesi; sıralı olan (ör: Job_1, Job_2, Job_3) ve olmayan (ör: Job_1, Job_4) işler, belirli kuyruklardaki bazı (ör: 3. ve 4. Kuyruktaki işlem süresi 10 ve üzeri olan işler) ya da bütün işler (ör: 4. Kuyruktaki bütün işler) ya da sistemdeki bazı (sistemde teslim zamanı 100 ve üzerindeki işler) ya da bütün işler şeklinde oluşturulabilir. Karşılaştırma parametresi ise herhangi bir iş parametresi (işlem süresi, teslim zamanı vb.) ya da bir denklem (PT + DD) olabilir. DIFF operatörü karşılaştırma parametresine bağlı olarak iş listesinde belirtilen işler arasında en iyi iş ya da işleri seçmektedir. Min parametresi ile kullanıldığında en iyi iş en küçük değerli, Max parametresi ile kullanıldığında ise en iyi iş en büyük değerli olan iştir. DIFF operatörünün çalışması sonucunda hem işin kendisi (SELECT benzeri) hem de seçilen işin iş listesi içeresindeki sıra numarası bilgisi elde edilir. Eğer birden fazla iş karşılaştırma parametresine göre aynı değere sahipse (ör: aynı işlem süresine sahip iki iş) her iki iş seçilir ve iş sıra numarası olarak “0” değeri elde edilir.

DIFF	Min / Max	Job_ID_1	Ölçüt Denklemi 1	Job_ID_2	Ölçüt Denklemi 2	…

DIFF operatörü ayrıca işler üzerinde her bir iş için farklı bir ölçüt denklemi kullanarak da işlem yapabilmektedir. Bu durumda kullanılacak işlerin sayısında bir kısıtlama bulunmamaktadır. En basit halinde Job_ID_1 bilgilerinden ölçüt denklemi 1 kullanılarak elde edilen değer ile Job_ID_2 bilgilerinden ölçüt denklemi 2 kullanılarak elde edilen değer karşılaştırılır ve en küçük (Min) değerli ya da en büyük (Max) değerli iş ve iş sırası seçilir. Örneğin indisi 2 olan işin işlem süresi ile indisi 5 olan işin bolluğu karşılaştırılıp en küçük değerli olan işin seçimi için kullanılabilir.

DIFF	Kuyruk listesi	Karşılaştırma Parametresi	Min / Max

Kuyruklar üzerinde karşılaştırma işlemi yapıldığı durumlarda ise kullanılan kuyruk listesi, sıralı olan veya olmayan ve bütün kuyruklar şeklinde oluşturulabilir. Karşılaştırma parametresi ise bir denklem (PT + DD) ya da kuyruğun performans ölçütü değeri (OBJ_VALUE) olabilmektedir. Bir denklem olduğu durumda kuyruktaki bütün işler için hesaplanan denklem değerlerinin toplamı kullanılır. İlgili değere göre en düşük (Min) ya da en büyük (Max) değerli kuyruk seçilir.

SUB_SET Operatörü:

SUB_SET operatörü belirtilen kritere göre bir grup işin seçilmesi için kullanılmaktadır. Genel yapıları aşağıdaki gibidir.

SUB_SET	Kuyruk_ID	Index_First	Index_Last

SUB_SET operatörü ile iş seçimi için başlangıç ve son indis şeklinde iş aralığı kullanabilmektedir. Bu durumda Index_First ile başlayan ve Index_Last ile biten bütün işler seçilmektedir. Örneğin iş indisi 2 ile 5 arasındaki 4 iş bu şekilde seçilebilir (Job_2, Job_3, Job_4, Job_5).

SUB_SET	Kuyruk_ID	Index_First	Step

SUB_SET operatörü ile iş seçimi için başlangıç indisi ile belirtilen işten itibaren belirli sayıda iş seçilebilmektedir. Bu durumda Index_First ile başlayan ve Step ile belirtilen sayıda iş seçilebilmektedir. Örneğin iş indisi 2 ve Step 2 olduğu durumda (Job_2, Job_3, Job_4) seçilir.

SUB_SET	Kuyruk_ID	Ölçüt Denklemi	Operatör {<, >, ≤, ≥, ==, !=}	Job_ID

SUB_SET operatörü ile belirtilen kuyruktaki ölçüt denklemini sağlayan bütün işlerde seçilebilmektedir. Ölçüt denkleminin karşılaştırılmasında kullanılan operatörler {<, >, ≤, ≥, ==, !=} şeklindedir ve Çizelge 5.3'de bu operatörler açıklanmıştır. Bu şekilde Kuyruk_ID ile belirtilen kuyruk içerisindeki bütün işler ölçüt denklemi ve operatör kullanılarak Job_ID ile belirtilen iş ile karşılaştırılır. Bu karşılaştırma sonucu işler seçilir. Örneğin "3 | PT | > | Job_1" ile 3. kuyruktaki işlem süreleri 1. işten büyük olan bütün işler seçilmektedir.

SORT_BASED_BEST Operatörü:

SORT_BASED_BEST operatörü ile ilgili kuyruktaki bütün işler, liste şeklinde belirtilen ölçüt denklemlerinden kullandığımız performans ölçütü değerini en iyileyen ölçüt denklemine göre sıralanırlar. Genel yapısı aşağıdaki gibidir.

SORT_BASED_BEST	Kuyruk_ID	Ölçüt Denklemleri Listesi	Min / Max

SORT_BASED_BEST operatörü içerisindeki ölçüt denklemleri listesinde bir ya da daha fazla ölçüt denklemi bulunabilmektedir. Bu operatör ile Kuyruk_ID ile belirtilen kuyruktaki işler performans ölçütünü enküçükleyecek (Min) ya da enbüyükleyecek (Max) şekilde sıralanırlar.

MOVE Operatörü:

MOVE operatörü ile bir kuyruktaki belirtilen kritere uyan bir iş diğer bir kuyruğa aktarılır. Genel yapısı aşağıdaki gibidir.

MOVE	**Kuyruk_ID_1**	**Ölçüt denklemi**	**Min / Max**	**Kuyruk_ID_2**	**First / Last**

MOVE operatörü ile Kuyruk_ID_1 ile belirtilen kuyruktan ölçüt denklemine göre en küçük (Min) ya da en büyük (Max) değere sahip iş, Kuyruk_ID_2 ile belirtilen kuyruğun başına (First) ya da sonuna (Last) aktarılmaktadır. Örneğin MOVE komutu ile 2. kuyruktaki en küçük işlem süresine sahip iş 4. kuyruğun başına aktarılabilir.

SWITCH_CASE Operatörü:

SWITCH_CASE operatörü algoritma içerisinde birden fazla dallandırma yapmak için kullanılır. Bu dallandırma N ile belirtilen değişken ya da ölçüt denkleminin değerine bağlı olarak yapılır. Genel yapısı aşağıdaki gibidir.

SWITCH_CASE (N)
varsayılan: N=0 olması durumunda yapılacak işlemler
1: N=1 olması durumunda yapılacak işlemler
…
m: N=m olması durumunda yapılacak işlemler

SWITCH_CASE operatöründe N'in bütün değerleri için yapılacak olan işlemlerin belirtilmesi zorunlu değildir. Sadece varsayılan durum başlangıçta zorunlu olarak bulunmaktadır. Çizelgeleme algoritmasının MEPAL ile geliştirilmesi ve iyileştirilmesi sırasında diğer durumlar SWITCH_CASE operatörü içerisine gerekliyse eklenebilmektedir. Eğer N'in değeri için SWITCH_CASE içerisinde bir ayrıt bulunmuyorsa varsayılan ile belirtilen işlemler yürütülür. Örneğin varsayılan, N=1, N=4 eşitlikleri için yapılacak işlemlerin bulunduğu bir SWITCH_CASE durumunda N değerinin 3 olması durumunda yapılacak olan işlemler varsayılan durumunda yapılacak olan işlemlerdir. Burada N işlem süresi, kuyruktaki iş sayısı, teslim zamanı ya da bir ölçüt denklemi olabilir.

JOB_POINTER Operatörü:

JOB_POINTER operatörü bir işin algoritma boyunca bellekte tutulmasını sağlar. Bir atama operatörüdür. Yani SELECT gibi iş seçme becerisi olan bir operatör ile seçilen işi algoritma boyunca bellekte tutar ve ilgili iş üzerinde veya ilgili iş kullanılarak işlem yapılabilmesini sağlar. SELECT operatörü gibi yapılar ile seçilen bir iş sonraki algoritma adımlarında tekrar SELECT operatörü çalışması durumunda hafızadan atılırken, JOB_POINTER operatörü ataması ile seçilen bir iş tekrar atama yapılana kadar hafızada kalacaktır. Kullanımına bir örnek aşağıdaki gibidir.

JOB_POINTER_1 = SELECT (Min) (PT) (Kuyruk_1)

Bu algoritma adımının çizelgeleme algoritması içerisinde kullanılması durumunda JOB_POINTER_1 ile belirtilen iş JOB_POINTER_1'e yeniden atama yapılmadığı sürece aynı kalacaktır. Bu işin başka bir operatör ile seçilmesi, bu kuyruktan başka bir kuyruğa aktarılmış olması vb. durumlar JOB_POINTER_1 içerisinde seçili olan işi etkilemeyecektir.

QUEUE_POINTER Operatörü:

QUEUE_POINTER operatörü bir kuyruğun algoritma boyunca bellekte tutulabilmesini ve üzerinde işlem yapılabilmesine olanak verir. Bir atama operatörüdür ve kullanımı aşağıdaki gibidir.

QUEUE_POINTER_1 = Kuyruk_ID_3

Yukarıda indisi 3 olan kuyruk bütün algoritma boyunca QUEUE_POINTER_1 ile ulaşılabilir şekilde atanmaktadır. Bu atama işlemi POINTER_QUEUE_1'e yeniden atama yapılana kadar geçerlidir. İlgili kuyrukta iş bulunup bulunmaması ya da APPEND operatörü ile ilgili kuyruğun başka bir kuyrukla işlerinin birleştirilmesi durumu değiştirmemektedir.

EQ_LIST Operatörü:

Çizelgeleme algoritmasının çalışması süresince kullanılabilecek olan bir ölçüt denklemi kümesine sahip bir operatördür. Bir atama operatörüdür. Kullanımı aşağıdaki gibidir.

EQ_LIST_1 = {Ölçüt denklemi 1, Ölçüt denklemi 2, …}

Çizelgeleme algoritması içerisinde EQ_LIST_1 şeklinde kullanılarak içerisindeki bütün ölçüt denklemlerine ulaşılabileceği gibi "EQ_LIST_1[indis]" şeklinde içerisindeki belirtilen indiste (sıradaki) bulunan ölçüt denklemine ulaşılabilir.

CLONE(JOB) Operatörü:

CLONE(JOB) operatörü belirtilen bir işin sanal bir imajını oluşturmak için kullanılır. Oluşturulan sanal işlerin üzerinde diğer gerçek işler gibi bütün işlemler yapılabilir. Fakat sanal işin kopyalandığı gerçek iş üzerinde oluşan değişiklikler (bolluk vb.) sanal işi etkilemeyecektir. Bir işin belirli bir andaki kopyasının oluşturularak çizelgeleme algoritmasının çalışması süresince bilgilerinin değişmeyecek şekilde saklanması şeklinde çalışır. Çizelgeleme algoritmasının çalışmasının sonlandığı anda oluşturulan bütün sanal işler eğer sonuç çizelgesinde yer almıyorsa sistemden silinmektedir. Bir işin birden fazla kuyrukta aynı anda incelenmesi ya da bir işin zaman içinde bilgilerindeki değişimin izlenmesi için kullanılmaktadır.

CLONE(QUEUE) Operatörü:

CLONE(QUEUE) operatörü ile bir kuyruğun kopyasının oluşturulması işlemi yapılmaktadır. Kopya oluşturma sırasında ilgili kuyrukta bulunan bütün işlerin de bir sanal kopyası oluşturulur. Kaynak kuyruk üzerinde yapılan işlemler ve kaynak kuyruktaki işlerin değişimi kopya kuyruğu etkilememektedir. Aynı şekilde kopya kuyruk üzerindeki değişiklikler kaynak kuyruğu etkilememektedir. Aynı kuyruk üzerinde farklı değişiklikler yapabilmek ve değişimi izlemeyebilmek amacıyla kullanılmaktadır.

5.1.2 Çoklu ifade programlama bileşeni

Çoklu ifade programlama bileşeni, MEPAL içerisindeki aday algoritma çözümlerini seçim, çaprazlama ve mutasyon genetik operatörleri ile iyileştiren bileşendir. Çoklu ifade programlama bileşeni içerisinde aday çözümleri barındıran örnek bir kromozom sadeleştirilmiş biçimde Şekil 5.3'de verilmiştir.

	Kromozom		
Gen 1	PUT_1		
Gen 2	PUT_2		
Gen 3	SELECT_1	1	
Gen 4	IF_1	1	2
Gen 5	IF_2	3	
Gen 6	SELECT_2	4	

Şekil 5.3: Örnek MEPAL kromozomu

Örnek MEPAL kromozomunda her bir gen bir algoritma yapı bileşeni barındırmakta ve eğer varsa izleyen algoritma adımını diğer genlere referans yoluyla belirtmektedir. İlgili MEPAL kromozomları baştan sona doğru okunan bir yapıya sahiptir. Şekil 5.3'de verilen örnek kromozomun içerdiği algoritmalar Şekil 5.4'de verilmiştir.

Gen 1	PUT_1
Gen 2	PUT_2
Gen 3	SELECT_1 PUT_1
Gen 4	IF_1 THEN PUT_1 ELSE PUT_2
Gen 5	IF_2 THEN SELECT_1 PUT_1
Gen 6	SELECT_2 IF_1 THEN PUT_1 ELSE PUT_2

Şekil 5.4: Örnek MEPAL kromozomunun içerdiği algoritmalar

Her bir MEPAL kromozomunda gen sayısı kadar algoritma kodlanmış durumdadır. Burada kodlanmış olan algoritmaların bazılarında (1, 2 ve 4) algoritmanın işleyişi sırasında problem çıkacaktır. Çünkü bu algoritmalarda bir iş seçilmeden iş üzerinde işlem yapılmaya çalışılmaktadır (PUT operatörü öncesinde SELECT operatörü olmaması). MEPAL bu kromozomla yeni algoritmalar geliştirebilecek olmasına rağmen işlevsel algoritma sayısı çoğu durumda az olacaktır. Bu tür eksikliklerin giderilmesi ve işlevsel algoritmaların sayısının arttırılması amacıyla her bir gen içerisindeki algoritmanın başına bir "*Varsayılan SELECT*" ve her bir gen içerisindeki algoritmanın sonuna "*Varsayılan APPEND*" operatörü sanal olarak eklenmektedir. Varsayılan SELECT ve varsayılan APPEND yapıları her kromozom için ayrı ayrı belirlenmekte ve çoklu ifade genetik programlama ile aranmasına olanak

sağlanmaktadır. İçerisinde bir SELECT operatörü olmayan algoritmalar böylece işler hale getirilmektedir. Eğer algoritma içerisinde bir SELECT operatörü varsa bu operatör ile tekrar bir seçim işlemi yapılacağından varsayılan SELECT operatörü işlevsiz kalmaktadır. Dolayısıyla algoritma üzerinde herhangi bir yıkıcı işleme sebep olunmamaktadır. Algoritma sonuna eklenen varsayılan APPEND operatörüyle ise algoritmanın çalışması sırasında oluşmuş olan iş kuyrukları (fiziksel ve sanal) birleştirilerek tezgah üzerinde işlenecek olan işlerin sıraları belirlenmiş olmaktadır. Yine varsayılan APPEND operatörü algoritma içerisinde bir APPEND operatörü bulunması halinde işlevsiz kalmaktadır. Varsayılan SELECT ve varsayılan APPEND operatörlerinin kullanımı ile Şekil 5.3'de verilen örnek kromozomun içerdiği algoritmalar Şekil 5.5'de verilmiştir.

Gen 1	*Default SELECT* PUT_1 *Default APPEND*
Gen 2	*Default SELECT* PUT_2 *Default APPEND*
Gen 3	SELECT_1 PUT_1 *Default APPEND*
Gen 4	*Default SELECT* IF_1 THEN PUT_1 ELSE PUT_2 *Default APPEND*
Gen 5	IF_2 THEN SELECT_1 PUT_1 *Default APPEND*
Gen 6	SELECT_2 IF_1 THEN PUT_1 ELSE PUT_2 *Default APPEND*

Şekil 5.5: Örnek MEPAL kromozomunun içerdiği algoritmalar

Şekil 5.5 içerisinde verilen örnek MEPAL kromozomundaki 4. genin içerdiği algoritma aşağıdaki gibi adımlaştırılabilir.

Adım 1: Eğer çizelgelenmemiş iş varsa Default SELECT ile belirtilen şekilde iş seçim işlemi yapılır ve Adım 2'ye geçilir. Bütün işler çizelgelenmişse Adım 5'e geçilir.

Adım 2: IF_1 ile yapılan karşılaştırma sonucu doğru ise Adım 3'e, değilse Adım 4'e geçilir.

Adım 3: Adım 1'de seçilen iş PUT_1 ile belirtilen kuyrukta belirtilen pozisyona yerleştirilir ve Adım 1'e dönülür.

Adım 4: Adım 1'de seçilen iş PUT_2 ile belirtilen kuyrukta belirtilen pozisyona yerleştirilir ve Adım 1'e dönülür.

Adım 5: PUT_1 ve PUT_2'de kullanılan kuyruklardaki işler bir tek kuyrukta birleştirilir.

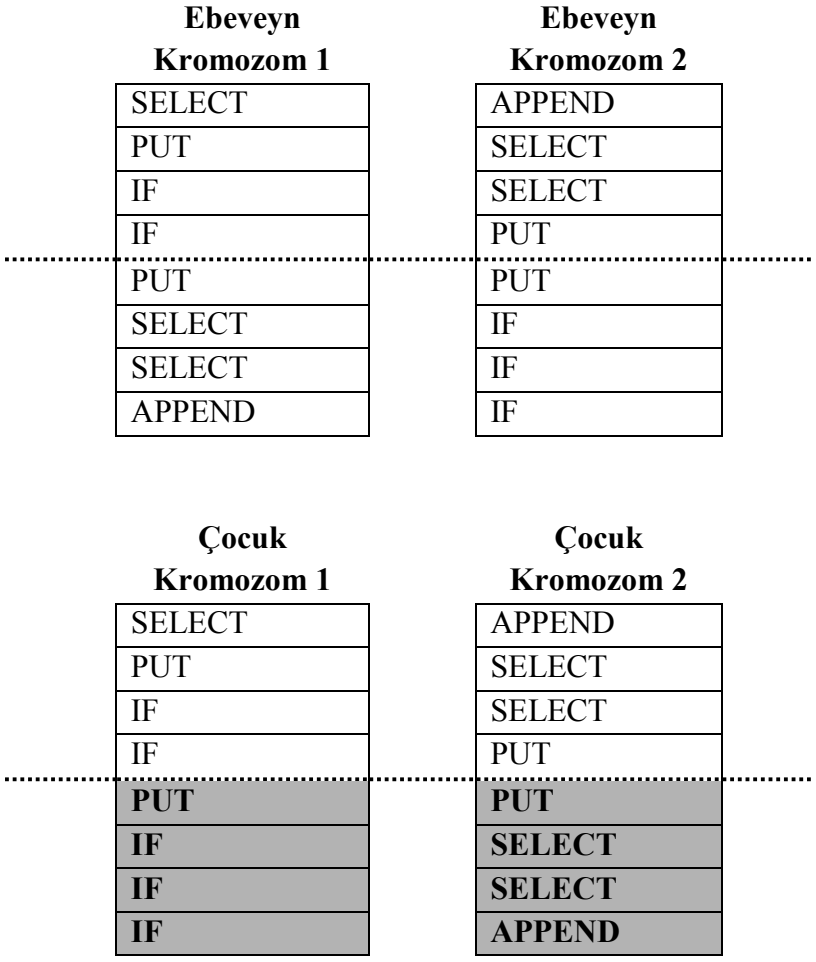
Çoklu ifade programlama tekniğinin aday algoritma çözümleri üzerinde gerçekleştireceği operatörler izleyen kısımda açıklanmıştır.

5.1.2.1 Seçim (Selection) operatörü

Bir sonraki nesile aktarılacak olan kromozomların seçimini yapan operatördür. Seçilen bu kromozomlar üzerinde bir sonraki nesile aktarılmadan önce çaprazlama ve mutasyon operatörleri de olasılıklarına göre yapılmaktadır. MEPAL'in bir alt bileşeni olan çoklu ifade programlama bileşeni içerisinde turnuva seçim tekniği uygulanmaktadır. Turnuva seçim büyüklüğü ise 2 olarak belirlenmiştir. Mevcut popülasyondaki en iyi uyumluluk değerine sahip kromozomun izleyen popülasyona aktarılmasını garantilemek amacıyla ise Elitizm uygulanmaktadır. Dolayısıyla her popülasyondaki en iyi uyumluluk değerine sahip olan kromozomun barındırdığı algoritmalar kaybedilmemiş olmaktadır.

5.1.2.2 Çaprazlama (Crossover) operatörü

Çaprazlama operatörü kromozomlar arasında bilgi alışverişine olanak veren ve bu yolla yeni çözümlerin oluşturulmasını sağlayan operatördür. Geliştirilmiş olan kromozom yapısı için tek nokta ya da çok nokta çaprazlama operatörleri kullanılabilir. MEPAL ile algoritma geliştirme çalışması için kromozomlar üzerinde tek nokta çaprazlama operatörü kullanılmaktadır. Tek nokta çaprazlama işleminde ebeveyn kromozomlarda her birinde aynı pozisyonda bir çaprazlama noktası seçilmekte ve çaprazlama noktası sonrasındaki parçalar değiş tokuş edilerek çocuk kromozomlar oluşturulmaktadır. Tek nokta çaprazlama işlemine bir örnek çaprazlama noktası 5. gen olmak üzere Şekil 5.6'da verilmiştir.



Şekil 5.6: MEPAL içerisinde tek nokta çaprazlama örnek işlemi

5.1.2.3 Mutasyon (Mutation) operatörü

Mutasyon operatörü çoklu ifade programlama bileşenin son safhasını oluşturur ve çözüm arama işleminin incelenmemiş bölgelere yayılmasını sağlayan bir operatördür. Ayrıca çözüm sürecinde çoklu ifade programlama bileşenin yerel en iyiye takılmasını önleme görevini de bu operatör üstlenmiştir. MEPAL içerisinde mutasyon operatörü olarak tek nokta mutasyon operatörü kullanılmaktadır. Basit mutasyon operatöründe kromozom üzerinde seçilen bir noktadaki bilgi başka bir bilgi ile değiştirilir. Yani seçilen noktada bir algoritma yapı bileşeni varsa bu bilgi başka bir algoritma yapı bileşeni ile bir denklem varsa başka bir denklem ile ve bir başka gene bir gen referansı varsa başka bir gene referans ile değiştirilmektedir. Basit mutasyon işlemine bir örnek mutasyon noktası 5. gendeki algoritma yapı bileşeni olmak üzere Şekil 5.7'de verilmiştir. Mutasyon işlemi sonucunda gen içerisindeki SELECT algoritma yapı bileşeni PUT ile değiştirilmiştir.

Mutasyon öncesi birey		**Mutasyon sonrası birey**
SELECT		SELECT
PUT		PUT
IF		IF
IF		IF
SELECT	→	**PUT**
APPEND		APPEND
APPEND		APPEND
PUT		PUT

Şekil 5.7: MEPAL içerisinde basit mutasyonun gen içeriği örnek işlemi

5.1.3 Üretim simülasyonu bileşeni

Üretim simülasyonu bileşeni çoklu ifade programlama bileşeni içerisindeki kromozomların barındırdığı çizelgeleme algoritmalarının ele alınan problemi ne kadar iyi çözebildiğini belirlemek için kullanılır. Kromozom içerisinde kodlanmış halde bulunan algoritmalar üretim simülasyonu bileşenine gönderilerek deterministik simülasyon ortamında çalıştırılır ve kullanılan performans ölçütüne bağlı olarak hesaplanan uyumluluk değeri çoklu ifade programlama bileşenine geri gönderilir. Böylece çoklu ifade programlama bileşeni hangi kromozomların daha etkili algoritmalara sahip olduğunu belirleyebilecektir.

5.2 Oluşturulan Test Problemleri

Çoklu İfade Programlama Algoritma Öğrenme Sistemi (MEPAL) ile algoritma keşfi, çalışmalarımızı yaptığımız bilgisayar sistemlerinden daha yüksek sistem gereksinimleri ve çok fazla zaman gerektirdiğinden, MELES için oluşturulan test problemlerine göre daha az sayıda test problemi kullanılmıştır. MEPAL ile çizelgeleme algoritması keşfi için kullanılacak olan test problemleri OR Library test problemi oluşturma yöntemi (Dimopoulos and Zalzala, 2001; Chou, 2009) ile türetilmiştir.

MEPAL ile çizelgeleme algoritmalarının keşfi aşamasında iki grup test problemi türetilmiştir. MEPAL'in mevcut çizelgeleme algoritmalarını keşfedebilirliğini test edebilmek amacıyla birinci ve ikinci test problemleri kümesi, MEPAL'in yeni çizelgeleme algoritmalarını keşif çalışmaları için üçüncü test problemleri kümesi kullanılacaktır.

MEPAL'in mevcut çizelgeleme algoritmalarını keşfedebilirliğini test etmek amacıyla literatürdeki Moore ve Johnson algoritmaları seçilmiştir. Moore algoritması tek tezgahlı çizelgeleme problemlerinde geç kalan işlerin toplam sayısını enküçükleyen bir algoritmadır. Johnson algoritması ise iki tezgahlı akış tipi üretim sistemlerindeki

çizelgeleme problemlerinde en büyük tamamlanma zamanını (makespan) enküçükleyen bir algoritmadır. Tek tezgahlı çizelgeleme problemlerinde geç kalan işlerin toplam sayısını enküçükleyen algoritma bulma çalışmalarında birinci, iki tezgahlı akış tipi üretim sistemlerindeki çizelgeleme problemlerinde en büyük tamamlanma zamanını (makespan) enküçükleyen algoritma bulma çalışmalarında ise ikinci test problemleri kümesi kullanılacaktır.

Her üç test problemleri kümesindeki işlerin işlem süreleri tamsayı şeklindedir ve işlem süreleri [1, 100] arası düzgün dağılıma uygun olarak türetilmiştir. Bütün test problemlerinde işler "0" anında sistemde bulunmakta ve her birinin ağırlığı "1" olacak şekilde birbirine eşit olarak işler ağırlıklandırılmaktadır. Bütün test problemlerinde işlerin teslim zamanları tamsayı olacak şekilde ve düzgün dağılım ile

$$[SP(1 - T - (R/2)), SP(1 - T + (R/2))]$$

aralığında olacak şekilde belirlenmiştir (Dimopoulos and Zalzala, 2001; Chou, 2009). Bu formülde SP problemdeki bütün işlerin işlem sürelerinin toplamı, T (TF – Tardiness Factor) gecikme faktörü ve R (RDD – Range of Due Date) teslim zamanı aralığıdır. Formüldeki T parametresi geciken işlerin beklenen yüzdesini göstermektedir. R parametresi ise teslim zamanı sıkılığını ya da oluşan teslim zamanı genişliğini göstermektedir. Problemlerin oluşturulması sırasında kullanılan teslim zamanı aralıkları, gecikme faktörleri ve bunlara göre oluşan teslim zamanı aralıkları Çizelge 5.4'de verilmiştir. Birinci test grubu için çizelgedeki her bir T ve R parametre çifti için 5 farklı problem oluşturulmuştur.

Çizelge 5.4: Gecikme zamanı sıklıkları (R), gecikme faktörleri (T) ve T ve R'ye göre oluşan teslim zamanı aralıları

T	R	Teslim Zamanı Alt Sınırı	Teslim Zamanı Üst Sınırı
0,2	0,2	0,7	0,9
0,2	0,4	0,6	1
0,2	0,6	0,5	1,1
0,2	0,8	0,4	1,2
0,2	1	0,3	1,3
0,4	0,2	0,5	0,7
0,4	0,4	0,4	0,8
0,4	0,6	0,3	0,9
0,4	0,8	0,2	1
0,4	1	0,1	1,1
0,6	0,2	0,3	0,5
0,6	0,4	0,2	0,6
0,6	0,6	0,1	0,7
0,6	0,8	0	0,8
0,6	1	-0,1	0,9
0,8	0,2	0,1	0,3
0,8	0,4	0	0,4
0,8	0,6	-0,1	0,5
0,8	0,8	-0,2	0,6
0,8	1	-0,3	0,7

Her test problemi kümesi için oluşturulan problemlerdeki iş sayıları ise 5, 6, 7, 8, 9 ve 10'dur. Örneğin 6 işli ve Çizelge 5.4'de verilen değişik T ve R parametre çiftlerinin hepsi için oluşturulan toplam problem sayısı 100'dür. Aynı şekilde diğer bütün iş sayıları için 100'er problem olmak üzere her test problemleri kümesi için toplamda işlem süreleri [1, 100] arası düzgün dağılan 600 problem oluşturulmuştur. Birinci ve üçüncü test problemleri kümelerinde problemler tez tezgahlı iken ikinci test problemleri kümesinde problemler iki tezgahlı problemlerdir. İkinci test problemleri

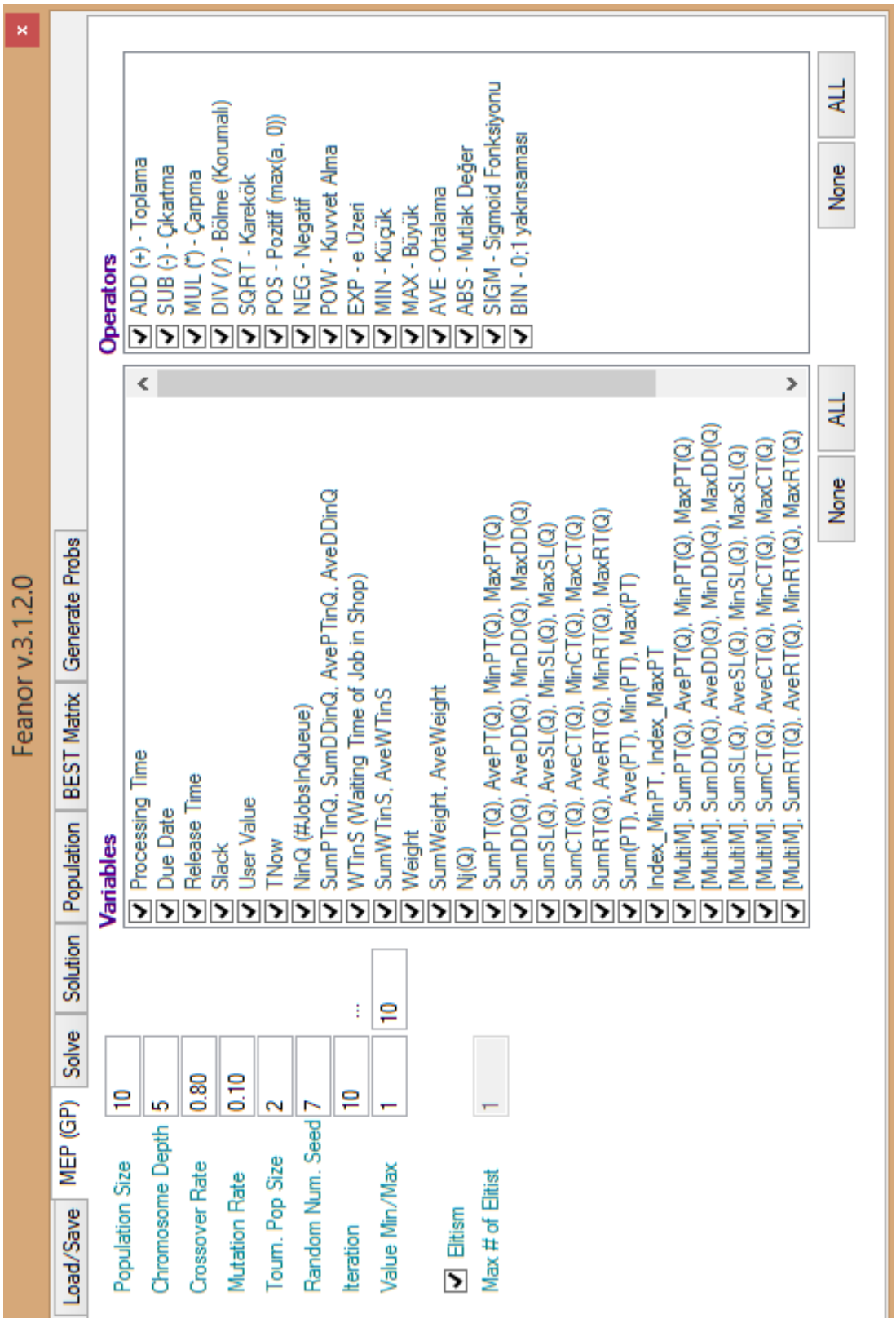
kümesinde bütün işler aynı sıra ile önce birinci ardından ikinci tezgah üzerinde işleneceklerdir. Birinci test problemleri kümesindeki problemler geç kalan işlerin toplam sayısı, ikinci test problemleri kümesindeki problemler en büyük tamamlanma zamanı ve üçüncü test problemleri kümesindeki problemler toplam gecikme performans ölçütleri göz önüne alınarak dal-sınır algoritması ile çözdürülmüş ve optimum değerleri elde edilmiştir. Çizelge 5.5'de problem kümelerinin içerikleri ve kullanım amaçları özet şeklinde gösterilmektedir.

Çizelge 5.5: Problem gruplarının içerikleri ve kullanım amaçları

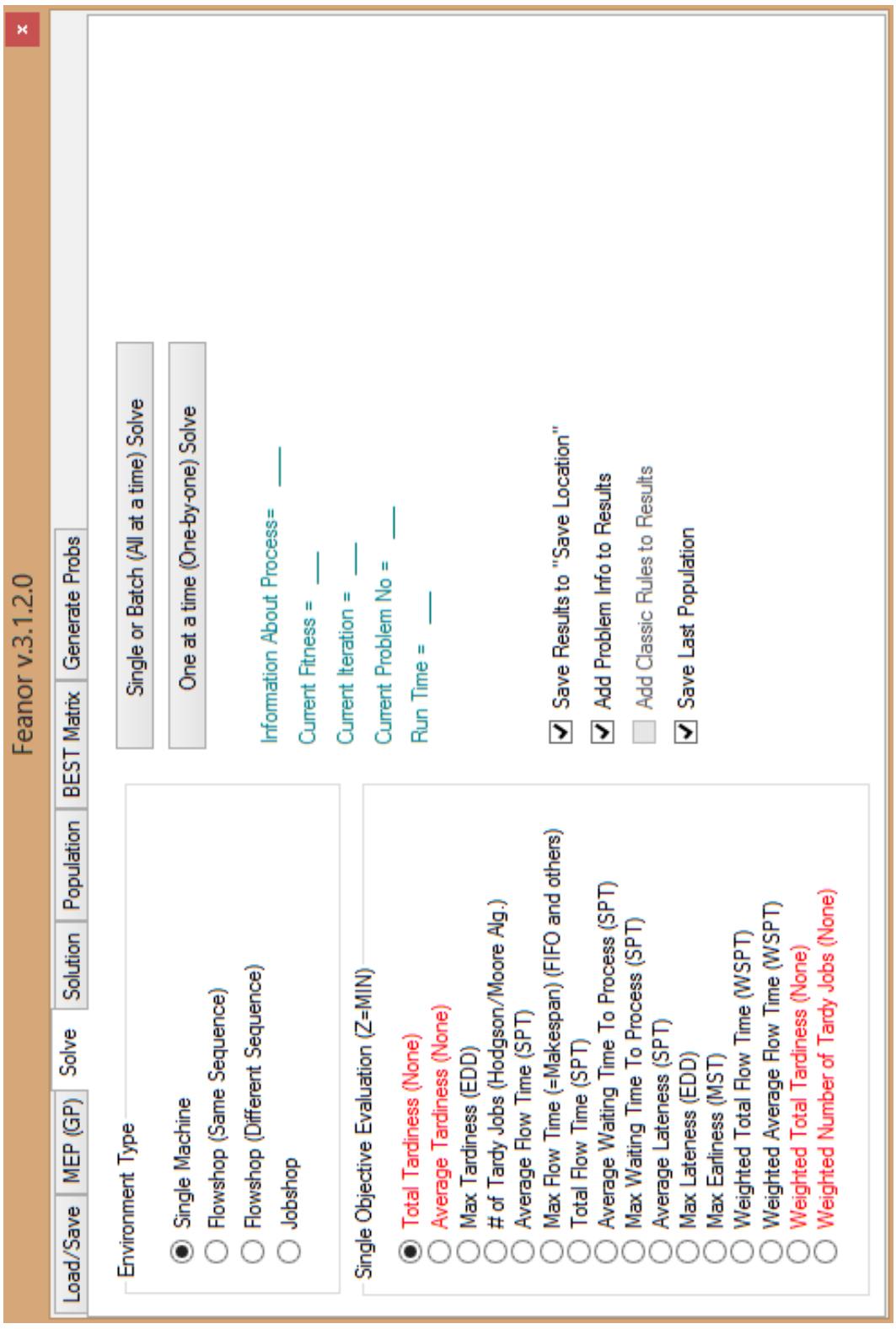
Grup Adı	İş Sayıları	Tezgah Sayıları	İşlem süreleri dağılımı	Toplam Problem Sayısı	Kullanım Şekli
1. Grup	5-10	1	[1, 100]	600	Tek tezgah – Geciken işlerin toplam sayısı ölçütü için çizelgeleme algoritmasının bulunması
2. Grup	5-10	2	[1, 100]	600	İki tezgah – En büyük tamamlanma zamanı ölçütü için çizelgeleme algoritmasının bulunması
3. Grup	5-10	1	[1, 100]	600	Tek tezgâh – Toplam gecikme ölçütü için çizelgeleme algoritması keşfi

5.3 Geliştirilen Program

Çoklu ifade programlama tabanlı algoritma öğrenme sistemi (MEPAL) C# dili kullanılarak geliştirilmiştir. Test problemleri ise Core i7 920 (8way) işlemci, 4 GB RAM'e sahip bir bilgisayar ve Xeon E5-2650 2GHz (64way) işlemci, 196 GB RAM'e sahip bir sunucu üzerinde çözülmüştür. Geliştirilen programın iki arayüz görüntüsü Şekil 5.8 ve Şekil 5.9'da verilmiştir.



Şekil 5.8: MEPAL terminal ve fonksiyon öğeleri arayüzü



Şekil 5.9: MEPAL üretim sistemi ortamı ve performans ölçütü arayüzü

MEPAL tez tezgahlı, akış tipi (tezgahlarda aynı iş sıralı ve farklı iş sıralı) ve çok tezgahlı çizelgeleme problemleri üzerinde çalışabilmektedir. Geliştirilen programda 16 adet amaç fonksiyonu (performans ölçütü) kullanılabilmektedir. Kullanılabilir performans ölçütü listesi aşağıda verilmiştir.

1) Toplam gecikme,
2) Ortalama gecikme,
3) En büyük gecikme,
4) Geciken işlerin toplam sayısı,
5) Ortalama akış süresi,
6) En büyük akış süresi,
7) Toplam akış süresi,
8) Ortalama bekleme süresi,
9) En büyük bekleme süresi,
10) Teslim zamanından ortalama sapma,
11) Teslim zamanından en büyük sapma,
12) En büyük erken bitirme,
13) Ağırlıklı toplam akış süresi,
14) Ağırlıklı ortalama akış süresi,
15) Ağırlıklı toplam gecikme,
16) Ağırlıklı geciken işlerin toplam sayısı.

5.4 Çoklu İfade Programlama Uygulama Parametreleri

Çoklu ifade programlama tabanlı algoritma öğrenme sisteminin (MEPAL) çalıştırılması esnasında kullanılan uygulama parametreleri özetle Çizelge 5.6'da verilmiştir. İlgili parametreler daha önce yapılan çalışmalar ve türetilen test problemleri üzerinde yapılan denemeler sonucunda belirlenmiştir.

Çizelge 5.6: MEPAL parametreleri

Parametre	Değer
Popülasyon büyüklüğü	50 – 200
Kromozom derinliği	200 - 250
Çaprazlama oranı	% 80
Mutasyon oranı	% 10
Turnuva büyüklüğü	2
Iterasyon	500 – 5000
Elitist sayısı	1

5.5 Sonuçlar

Çoklu ifade programlama tabanlı algoritma öğrenme sistemi (MEPAL) ile tek tezgahlı çizelgeleme problemlerinde geç kalan işlerin toplam sayısını enküçükleyen algoritma bulma çalışmaları, iki tezgahlı akış tipi üretim sistemlerindeki çizelgeleme problemlerinde en büyük tamamlanma zamanını enküçükleyen algoritma bulma çalışmaları ve tek tezgahlı çizelgeleme problemlerinde toplam gecikmeyi enküçükleyen algoritma bulma çalışmaları izleyen bölümlerde verilmiştir.

5.5.1 Tek tezgah – Geciken işlerin toplam sayısı ölçütü için çizelgeleme algoritmaları keşfi sonuçları

Tek tezgahlı çizelgeleme problemleri için geciken işlerin toplam sayısı ölçütü göz önüne alınarak MEPAL ile çizelgeleme algoritması keşfi çalışma sonuçları bu bölümde incelenecektir. Literatürde tek tezgahlı çizelgeleme problemleri için geciken işlerin toplam sayısı ölçütünü eniyilemek amacıyla Moore algoritması kullanılmaktadır. Moore algoritmasının işlerin sıfır anında sistemde olduğu durumlar için tek tezgahlı

çizelgeleme problemlerinde geciken işlerin toplam sayısını enküçüklediği bilinmektedir (French, 1982). Moore algoritması aşağıda verilmiştir.

Adım 1: Bütün işler EDD öncelik kuralına göre sıralanır. Elde edilen sıralama (J1, J2, …, Jn) olsun.
Adım 2: Sıralama içerisindeki ilk geciken iş bulunur (J4 kabul edelim). Eğer geciken iş yoksa Adım 4'e gidilir.
Adım 3: Sıralama içerisinde ilk geciken işe kadar olan işler içerisinde (J1, J2, J3, J4) en büyük işlem zamanına sahip iş sıralamadan çıkartılarak Reddedilen İşler kümesine alınır.
Adım 4: Tezgahta öncelikle sıralamada kalan işler mevcut sıraları ile ardından reddedilen işler herhangi bir sıra ile işlenirler. Reddedilen işler her zaman geciken işler olacağından işlem görme sıraları geciken işlerin toplam sayısını değiştirmeyecektir.

Tek tezgahlı çizelgeleme problemleri için geciken işlerin toplam sayısı ölçütü göz önüne alınarak MEPAL ile çizelgeleme algoritması keşfi çalışmaları için birinci grup test problemleri kullanılmıştır. Birinci test problemleri grubu, 5, 6, 7, 8, 9 ve 10 işli 100'er test probleminden oluşmakta ve toplamda 600 test problemi barındırmaktadır. İlk aşamada birinci test problemleri grubundaki her bir problem tek tek farklı rassal sayı çekirdekleri (1 ve 2) kullanılarak çözdürülmüş ve Dal-Sınır algoritması ile bulunan en iyi çözümlerle karşılaştırılmıştır. Her bir çalıştırma sonucu oluşturulan çizelgeleme algoritmaları ile ilgili problemin en iyi çözümü elde edilmiştir. Bulunan sonuçlara örnek olarak 7 işli problemlerin bir kısmı için sonuçlar izleyen Çizelge 5.7'de verilmiştir.

Çizelge 5.7: 7 işli problemlerin durum çizelgesi ve elde edilen çizelgeleme algoritmaları

TF	RDD	Prob	Bulunan Çizelgeleme Algoritması	En İyi Çözüm	Bulunan Çözüm	Bilinen bir çözüm mü?
0,2	0,2	1	SORT (AZ) (PT) (Q1)	0	0	SPT
0,2	0,2	3	SELECT (Min) (DD) (Q1) PUT (Q1) (Last)	1	1	EDD
0,2	0,4	1	SORT (AZ) (Max(SL, PT)) (Q1)	2	2	
0,2	0,4	4	SORT (AZ) (PT+Max(SL,0)) (Q1)	1	1	
0,4	0,4	2	SORT (AZ) (DD) (Q1)	1	1	EDD

0,4	0,4	3	SELECT (Min) (SL)
			IF (SL < 0) THEN
			{PUT (Q2) (First)}
			ELSE
			{PUT (Q3) (Last)}
			APPEND (Q3, Q2)
				1	1	

0,4	0,6	1	SELECT (Min) (DD/SL)
			IF (SL > 0) THEN
			{PUT (Q2) (Last)}
			ELSE
			{PUT (Q3) (First)}
			APPEND (Q2, Q3)
				2	2	

0,6	0,2	2	SORT (AZ) ((PT+DD)/SL) (Q1)	1	1	
0,6	0,4	2	SELECT (Min) (Max(SL,PT)) PUT (Q1) (Last)	2	2	
0,8	0,2	1	SELECT (Min) (DD) (Q1) PUT (Q2) (Last) IF (CT – DD > 0) THEN {SELECT (Max) (PT) (Q2) PUT (Q3) (Last) APPEND (Q1, Q3)}	4	4	Moore Algoritması
0,8	0,4	2	SELECT (Min) (DD) (Q1) PUT (Q2) (Last) IF (CT – DD > 0) THEN {SELECT (Max) (PT) (Q2) PUT (Q3) (Last) APPEND (Q1, Q3)}	3	3	Moore Algoritması
0,8	0,8	1	SORT (AZ) (PT+Max(SL,0)) (Q1)	2	2	

Birinci test problemleri grubundaki bütün problemler MEPAL ile keşfedilen çizelgeleme algoritmaları vasıtasıyla geciken işlerin toplam sayısı performans ölçütüne bağlı olarak optimum çözülmüştür.

Birinci test problemleri grubundaki test problemlerinin tek tek çözdürülmesi ile elde edilen çizelgeleme algoritmaları incelendiğinde birçok problemin Moore algoritmasına alternatif bir çizelgeleme algoritması ile çözüldüğü görülmüştür. Bunun sebebi problemler tek tek çözdürüldüğünde MEPAL'in optimum çözümü veren ve daha karmaşık olan Moore algoritmasını bulmadan önce eniyi çözümü veren daha basit bir çizelgeleme algoritması keşfetmesidir. Problemlerin tek tek ele alındığı bu aşamada MEPAL toplam 600 test probleminden 16 tanesi için Moore algoritmasını bulmuştur ve bu problemlerin hepsi gecikme faktörü (TF) 0,8 olan problemlerdir. Gecikme faktörü bir problemdeki gecikmesi beklenen işlerin yüzdesini vermektedir.

Problemlerin ortak karakteristiklerini MEPAL ile keşfedilecek çizelgeleme algoritmasına yansıtmak ve daha genel bir çizelgeleme algoritması bulmak amacıyla birinci test problemleri grubundaki test problemleri 50 problemden oluşan alt kümelere ayrılmış ve 12 alt problem kümesi oluşturulmuştur. Bu alt problem kümelerine MEPAL ile çizelgeleme algoritması keşfetme çalışmaları için elde ettiğimiz sonuçlar Çizelge 5.8'de verilmiştir.

Çizelge 5.8: Problem alt kümeleri için keşfedilen çizelgeleme algoritmaları

Alt küme	İş Sayısı	Problem Sayısı	Bulunan Algoritma
1	5	50	Moore Alg. (*)
2	5	50	Moore Alg. (*)
3	6	50	Moore Alg. (*)
4	6	50	Moore Alg. (*)
5	7	50	Moore Alg. (*)
6	7	50	Moore Alg. (*)
7	8	50	Moore Alg. (*)
8	8	50	Moore Alg. (*)
9	9	50	Moore Alg. (*)
10	9	50	Moore Alg. (*)
11	10	50	Moore Alg. (*)
12	10	50	Moore Alg. (*)

(*)

```
SELECT (Min) (DD) (Q1)
PUT (Q2) (Last)
IF (CT – DD > 0) THEN
  {SELECT (Max) (PT) (Q2)
   PUT (Q3) (Last)
   APPEND (Q1, Q3)}
```

Birinci test problemleri grubundaki problemlerden oluşturulan 50 test problemlik alt kümelere MEPAL ile çizelgeleme algoritması oluşturulması sonucu her bir alt küme için Moore algoritması keşfedilmiştir. Dolayısıyla çoklu ifade programlama tabanlı algoritma öğrenme sistemimiz geciken işlerin toplam sayısı performans ölçütü göz önüne alınarak yapılan çizelgeleme algoritması keşfinde problemlerin karakteristiklerini ve değişimlerini öğrenebilmiş ve buna uygun olan çizelgeleme algoritmasını keşfedebilmiştir.

5.5.2 İki tezgah – En büyük tamamlanma zamanı ölçütü için çizelgeleme algoritmaları keşfi sonuçları

İki tezgahlı çizelgeleme problemleri için en büyük tamamlanma zamanı ölçütü göz önüne alınarak MEPAL ile çizelgeleme algoritması keşfi çalışma sonuçları bu bölümde incelenecektir. Literatürde iki tezgahlı çizelgeleme problemleri için en büyük tamamlanma zamanı ölçütünü eniyilemek amacıyla Johnson algoritması kullanılmaktadır. Johnson algoritmasının işlerin sıfır anında sistemde olduğu, her iki tezgah içinde aynı iş sırası kullanıldığı ve her bir işin önce birinci ardından ikinci tezgahta işlem gördüğü durum için iki tezgahlı çizelgeleme problemlerinde en büyük tamamlanma zamanını enküçüklediği bilinmektedir (French, 1982). Johnson algoritması aşağıda verilmiştir.

***n** adet iş için*

Adım 1: k = 1, m = n atanır.

Adım 2: Çizelgelenmemiş işler kümesi A = {J1, j2, …, Jn} olarak, çizelgelenmiş işler B = {} olarak atanır.

Adım 3: A kümesindeki işlerden herhangi bir tezgahtaki en küçük işlem süreli iş seçilir. Aynı en küçük işlem süresine sahip birden fazla iş varsa iş rastgele seçilir.

Adım 4: Eğer işin en küçük işlem süresi birinci tezgaha ait ise aşağıdaki alt işlemler yapılır. Değilse Adım 5'e geçilir.

a) İlgili iş B kümesinde k. pozisyona yerleştirilir.
b) İlgili iş A kümesinden çıkartılır.
c) k bir arttırılır. (k = k + 1)
d) Adım 6'ya geçilir.

Adım 5: (İşin en küçük işlem süresi ikinci tezgaha ait ise aşağıdaki alt işlemler yapılır.)

a) İlgili iş B kümesinde m. pozisyona atanır.
b) İlgili iş A kümesinden çıkartılır.
c) m bir azaltılır. (m = m – 1)
d) Adım 6'ya geçilir.

Adım 6: Çizelgelenmemiş işler kümesi A'da hala iş varsa Adım 2'ye geçilir.

İki tezgahlı çizelgeleme problemleri için en büyük tamamlanma zamanı ölçütü göz önüne alınarak MEPAL ile çizelgeleme algoritması keşfi çalışmaları için ikinci grup test problemleri kullanılmıştır. İkinci test problemleri grubu, 5, 6, 7, 8, 9 ve 10 işli 100'er test probleminden oluşmakta ve toplamda 600 test problemi barındırmaktadır. İkinci test problemleri grubundaki her bir problem tek tek farklı rassal sayı çekirdekleri (1 ve 2) kullanılarak çözdürülmüş ve Dal-Sınır algoritması ile bulunan en iyi çözümlerle karşılaştırılmıştır. Her bir çalıştırma sonucu oluşturulan çizelgeleme algoritmaları ile ilgili problemin en iyi çözümü elde edilmiştir. Bulunan sonuçlara örnek olarak 7 işli problemlerin bir kısmı için sonuçlar izleyen Çizelge 5.9'da verilmiştir.

Çizelge 5.9: 7 işli problemlerin durum çizelgesi ve elde edilen çizelgeleme algoritmaları

TF	RDD	Prob	Bulunan Çizelgeleme Algoritması	En İyi Çözüm	Bulunan Çözüm	Bilinen bir çözüm mü?
0,2	0,2	1	SORT (AZ) (PT) (Q1)	324	324	SPT
0,2	0,6	3	Algoritma 1	376	376	Johnson
0,2	1,0	2	Algoritma 1	337	337	Johnson
0,4	0,2	4	Algoritma 2	298	298	Johnson
0,4	0,4	3	Algoritma 1	378	378	Johnson
0,4	0,8	1	Algoritma 2	301	301	Johnson
0,6	0,2	1	SELECT (Min) (PT) (Q1) PUT (Q1) (Last)	362	362	SPT
0,6	0,4	3	Algoritma 2	364	364	Johnson
0,6	0,4	3	Algoritma 1	418	418	Johnson
0,8	0,2	2	SORT (AZ) (PT) (Q1)	275	275	Johnson
0,8	0,6	4	Algoritma 2	400	400	Johnson
0,8	1,0	4	Algoritma 2	310	310	Johnson

Algoritma 1:

JOB1 = SELECT (Min) (PT(1)) (Q0)

JOB2 = SELECT (Min) (PT(2)) (Q0)

SWITCH_CASE (DIFF((JOB1, JOB2), (Min)))

Varsayılan: PUT (JOB1) (Q1) (Last)

0: PUT (JOB1) (Q1) (Last)

1: PUT (JOB1) (Q1) (Last)

2: PUT (JOB2) (Q2) (First)

Algoritma 2:

IF (MIN(PT(1), Q0) < MIN(PT(2), Q0))

THEN {MOVE (Q0) (PT(1)) (Min) (Q1) (Last)}

ELSE {MOVE (Q0) (PT(2)) (Min) (Q2) (First)}

İkinci test problemleri grubundaki bütün problemler MEPAL ile iki tezgahlı çizelgeleme problemleri için keşfedilen çizelgeleme algoritmaları vasıtasıyla en büyük tamamlanma zamanını enküçüklemek performans ölçütüne bağlı olarak optimum çözülmüştür.

İkinci test problemleri grubundaki test problemlerinin tek tek çözdürülmesi ile elde edilen çizelgeleme algoritmaları incelendiğinde, birçok problemin Johnson algoritması keşfedilerek çözüldüğü görülmüştür. Johnson algoritması iki farklı kodlama şeklinde keşfedilmiştir (Çizelge 5.9'de Algoritma 1 ve Algoritma 2). Problemlerin tek tek ele alındığı bu aşamada MEPAL toplam 600 test probleminden 523 tanesi için Johnson algoritmasını bulmuştur. Dolayısıyla çoklu ifade programlama tabanlı algoritma öğrenme sistemimiz en büyük tamamlanma zamanı performans ölçütü göz önüne alınarak yapılan çizelgeleme algoritması keşfinde iki tezgahlı çizelgeleme problemleri için uygun çizelgeleme algoritmaları keşfedebilmiştir.

5.5.3 Tek tezgah – Toplam gecikme ölçütü için çizelgeleme algoritmaları keşfi sonuçları

Tek tezgahlı çizelgeleme problemleri için toplam gecikme performans ölçütü göz önüne alınarak MEPAL ile çizelgeleme algoritması keşfi çalışma sonuçları bu bölümde incelenecektir. Literatürde tek tezgahlı çizelgeleme problemleri için toplam gecikme ölçütünü eniyilemek amacıyla kullanılan bir çizelgeleme algoritması bulunmamaktadır.

Tek tezgahlı çizelgeleme problemleri için toplam gecikme performans ölçütü göz önüne alınarak MEPAL ile çizelgeleme algoritması keşfi çalışmaları için üçüncü grup test problemleri kullanılmıştır. Üçüncü test problemleri grubu, 5, 6, 7, 8, 9 ve 10 işli 100'er test probleminden oluşmakta ve toplamda 600 test problemi barındırmaktadır. İlk aşamada üçüncü test problemleri grubundaki her bir problem tek tek farklı rassal sayı çekirdekleri (1, 2, 3 ve 4) kullanılarak çözdürülmüş ve Dal-Sınır algoritması ile bulunan en iyi çözümlerle karşılaştırılmıştır. Her bir çalıştırma sonucu oluşturulan çizelgeleme algoritmaları ile ilgili 600 test probleminin en iyi çözümü elde edilmiştir. Birinci aşamada elde edilen çizelgeleme algoritmaları büyük bir çeşitliliğe sahiptir ve genelde aşağıdaki gibi karmaşık olmayan yapılara sahiptir.

Örnek Algoritma 1 (MEPAL Çıktısı):

SELECT [Min] [PT] [Q1]

PUT [Q2] [Last]

Örnek Algoritma 1 (Adımlaştırılmış):

Aşağıdaki adımlar iş sayısı kadar uygulanmaktadır.

Adım 1: Q1 kuyruğundan en küçük işlem süreli iş seçilir.

Adım 2: Seçilen iş Q2 kuyruğundaki işlerin sonuna eklenir.

Örnek Algoritma 2 (MEPAL Çıktısı):

SORT [AZ] [PT] [Q1]

Örnek Algoritma 2 (Adımlaştırılmış):

Adım 1: Q1 kuyruğundaki işler işlem sürelerine göre küçükten büyüğe sıralanır.

Örnek Algoritma 3 (MEPAL Çıktısı):

SORT_BASED_BEST [Q1] [(PT); (PT + DD)] [Min]

Örnek Algoritma 3 (Adımlaştırılmış):

Adım 1: Q1 kuyruğundaki işler işlem sürelerine göre küçükten büyüğe sıralanır ve kullanılan performans ölçütüne göre performans ölçütü değeri belirlenir.

Adım 2: Q1 kuyruğundaki işler işlem süreleri ve teslim zamanları toplamına (PT+DD) göre küçükten büyüğe sıralanır ve kullanılan performans ölçütüne göre performans ölçütü değeri belirlenir.

Adım 3: En küçük performans ölçütü değeri belirlenir ve bu performans ölçütü değerini veren sıralama en son sıralama olarak kabul edilir.

Üçüncü test problemleri grubundaki bütün problemler MEPAL ile keşfedilen çizelgeleme algoritmaları vasıtasıyla geciken işlerin toplam sayısı performans ölçütüne bağlı olarak optimum çözülmüştür.

Üçüncü test problemleri grubundaki test problemlerinin tek tek çözdürülmesi ile elde edilen sonuçlar incelendiğinde keşfedilen çizelgeleme algoritmalarından baskın bir ya da daha fazla çizelgeleme algoritması bulunmamaktadır. Problemlerin ortak karakteristiklerini MEPAL ile keşfedilecek çizelgeleme algoritmasına yansıtmak ve daha genel ve baskın bir çizelgeleme algoritması bulmak amacıyla üçüncü test problemleri grubundaki test problemleri 50 sıralı problemden ve 50 rastgele seçilmiş problemden oluşan alt kümelere ayrılmış ve 24 alt problem kümesi oluşturulmuştur. Bu alt problem kümelerine MEPAL ile çizelgeleme algoritması keşfetme çalışmaları için elde ettiğimiz sonuç çizelgeleme algoritmaları aşağıda verilmiştir.

Algoritma 1:

```
SELECT [Min][DD / PT][Q1]

PUT [Q2][First]

FOREACH_JOB [Q1]

{

IF (OBJ_VALUE(Q2, ASSUME(SELECT([Min] [Max(SL, PT) + PT + Max(SL, 0)]
[Q1]), [Q2] [Last])) < OBJ_VALUE(Q2, ASSUME(SELECT([Min] [PT + Max(SL, 0)]
[Q1]), [Q2] [Last])))

        THEN {SELECT [Min] [Max(SL, PT) + PT + Max(SL, 0)] [Q1]}

        ELSE {SELECT [Min] [PT + Max(SL, 0)] [Q1]}

PUT [Q2] [Last]

JOB1 = SUBSET(Q2, 1, step:0)

PUT [JOB1] [Q3] [Last]

}
```

Algoritma 1 Açıklaması:

Bu çizelgeleme algoritmasında toplam üç sanal kuyruk bulunmaktadır. Birinci sanal kuyruk (Q1) çizelgelenmemiş işleri barındırırken, çizelgeleme algoritması başlangıcında Q2 ve Q3 sanal kuyrukları boştur. Çizelgeleme algoritması DD/PT oranı en küçük işi seçip Q2'ye aktararak işlemlerine başlamaktadır. Ardından Q1 sanal kuyruğunda kalan işlerden "Max(SL, PT) + PT + Max(SL, 0)" denklemi değeri en küçük iş ile "PT + Max(SL, 0)" değeri en küçük olan iş belirlenmektedir. Seçilen bu işlerin ayrı ayrı Q2'de bulunan işin ardından işlenmesi halinde ortaya çıkan performans ölçütü değerine bakarak en iyi sonucu veren iş sırasına ait iş Q2'deki işin ardına eklenmektedir. Aktarma işleminin ardından Q2'de bulunan ilk iş Q3 sanal kuyruğuna aktarılmakta ve işlemler çizelgelenmemiş işler bitene tekrarlanmaktadır.

Özet olarak bu çizelgeleme algoritması iki farklı algoritma denklemi ile seçilen işlerden hangisinin performans ölçütü değerini daha az etkilediğini (kötüleştirdiğini) bulmakta ve seçim işlemini buna göre yapmaktadır.

Algoritma 2:

```
K = 0
SELECT [Min] [PT + Max(PT, SL) + Max(SL, 0)] [Q1]
PUT [Q2] [Last]
IF (CT – DD > 0)
        THEN {K = K + 1}
IF (K > 1)
        THEN {FOREACH_JOB[Q1]
        {
                SELECT [Min] [PT + Max(SL, 0)] [Q1]
                PUT [Q2] [Last]
        }
}
```

Algoritma 2 Açıklaması:

Bu çizelgeleme algoritması çizelgelenmemiş işler arasından "PT + Max(PT, SL) + Max(SL, 0)" algoritma denklemi değeri en küçük olan işi seçmekte ve geçici iş sıralamasına eklemektedir. Ardından, eklenen işin mevcut geçici iş sıralamasındaki pozisyonuna göre gecikip gecikmediğini kontrol etmektedir. Gecikmesi durumunda K değişkeni değerini bir arttırmaktadır. Geçici iş sıralamasına her yeni eklenen işin iş sıralamasındaki pozisyonuna göre gecikmesi kontrol edilmektedir. Geçici iş sıralamasında geciken işlerin sayısı birden fazla olduğu durumdan itibaren iş sırası

oluşturma işlemi, çizelgelenmemiş işler arasından "PT + Max(SL, 0)" algoritma denklemi değeri en küçük olan iş seçilip geçici iş sırasının sonuna eklenmesi şeklinde devam etmektedir.

Özet olarak, bu çizelgeleme algoritması ikinci geciken işin ortaya çıkmasına kadar iş seçimi için "PT + Max(PT, SL) + Max(SL, 0)" algoritma denklemini kullanmaktadır. İkinci geciken işin ortaya çıkması ile iş seçimine "PT + Max(SL, 0)" algoritma denklemi kullanılarak devam edilmektedir.

Algoritma 3:

```
IF (OBJ_VALUE(SORT [AZ] [PT + Max(SL, 0)] [Q1]) < OBJ_VALUE(SORT [AZ] [PT + Max(SL, 0) + Max(SL, PT)] [Q1]))
THEN
{
    IF (OBJ_VALUE(SORT [AZ] [PT + Max(SL, 0)] [Q1]) < OBJ_VALUE(SORT [AZ]  [Max(SL, PT)] [Q1]))
        THEN {SORT [AZ][PT + Max(SL, 0)][Q1]}
        ELSE {SORT [AZ][Max(SL, PT)][Q1]}
}
ELSE
{
    IF (OBJ_VALUE(SORT [AZ] [PT + Max(SL, 0) + Max(SL, PT)] [Q1]) < OBJ_VALUE(SORT [AZ]  [Max(SL, PT)] [Q1]))
        THEN {SORT [AZ][PT + Max(SL, 0) + Max(SL, PT)][Q1]}
        ELSE {SORT [AZ][Max(SL, PT)][Q1]}
}
```

Algoritma 3 Açıklaması:

Bu çizelgeleme algoritması kendi içerisinde üç adet algoritma denklemi kullanmaktadır. Bu algoritma denklemleri aşağıdaki gibidir.

Algoritma Denklemi 1: PT + Max(SL, 0)

Algoritma Denklemi 2: PT + Max(SL, 0) + Max(SL, PT)

Algoritma Denklemi 3: Max(SL, PT)

Çizelgeleme algoritması ilgili üç algoritma denklemine göre işlerin sıralandığı durumlarda elde edilen performans ölçütü değerlerini temel alarak en iyi iş sırasını bulmaktadır. Bunun için her bir algoritma denklemi ile oluşturulan iş sıraları diğerleri ile tek tek karşılaştırılmaktadır. Bulunan en iyi iş sırası çözüm olarak sunulmaktadır.

Algoritma 4:

SORT_BASED_BEST [Q1] [(PT + Max(SL, 0) + Max(SL, PT)); (Max(SL, PT)); (PT + Max(SL, 0)); (SL – PT)] [Min]

Algoritma 4 Açıklaması:

Bu çizelgeleme algoritması MEPAL'e başlangıçta verilmeden MEPAL tarafından çizelgeleme algoritması öğrenme sürecinde geliştirilmiş olan 4 adet algoritma denklemini temel alarak çalışmaktadır. Bu algoritma denklemleri aşağıda verilmiştir.

Algoritma Denklemi 1: PT + Max(SL, 0) + Max(SL, PT)

Algoritma Denklemi 2: Max(SL, PT)

Algoritma Denklemi 3: PT + Max(SL, 0)

Algoritma Denklemi 4: SL – PT

Çizelgeleme algoritması ilgili dört algoritma denklemi ile mevcut işleri ayrı ayrı sıralayarak bu sıralamalardan hangisinin toplam gecikme performans ölçütünü eniyilediğine bakmakta ve ilgili iş sırasını çözüm olarak sunmaktadır.

Algoritma 5:

```
EQ_LIST_1 = {(PT + Max(SL, 0) + Max(SL, PT)); (Max(SL, PT)); (PT + Max(SL, 0)); (PT + DD)}
FOREACH (EQ_LIST)
{
    QUEUE_(index) = CLONE(SORT[AZ][EQ_LIST[index]][Q1])
}
DIFF(QUEUE(ALL))
```

Algoritma 5 Açıklaması:

Bu çizelgeleme algoritması MEPAL'e başlangıçta verilmeden MEPAL tarafından çizelgeleme algoritması öğrenme sürecinde geliştirilmiş olan 4 adet algoritma denklemini temel alarak çalışmaktadır. Bu algoritma denklemleri aşağıda verilmiştir.

Algoritma Denklemi 1: PT + Max(SL, 0) + Max(SL, PT)

Algoritma Denklemi 2: Max(SL, PT)

Algoritma Denklemi 3: PT + Max(SL, 0)

Algoritma Denklemi 4: PT + DD

Çizelgeleme algoritması ilgili dört algoritma denklemi ile mevcut işleri ayrı ayrı sıralayarak bu sıralamalardan sanal kuyruk görüntüleri (imajları) oluşturmaktadır. Ardından sanal kuyruk imajlarından hangisinin toplam gecikme performans ölçütünü eniyilediğine bakmakta ve ilgili iş sırasını çözüm olarak sunmaktadır.

MEPAL ile keşfedilen ve yukarıda kendileri ve işleyiş bilgileri verilen beş çizelgeleme algoritması kullanılarak üçüncü test problemleri grubundaki toplam 600 test problemi için en iyi toplam gecikme değeri bulunan toplam problem sayıları bazında sonuçlar Çizelge 5.10'da verilmiştir.

Çizelge 5.10: Keşfedilen çizelgeleme algoritmaları ile üçüncü test problemleri grubundaki optimum çözülen test problemi sayıları

İş Sayısı	Çiz. Alg. 1	Çiz. Alg. 2	Çiz. Alg. 3	Çiz. Alg. 4	Çiz. Alg. 5
5	89	95	100	100	100
6	90	96	100	100	100
7	88	95	99	100	100
8	85	93	99	99	100
9	79	92	97	100	100
10	80	95	98	99	100
Toplam	511 / 600	566 / 600	593 / 600	598 / 600	600 / 600

Elde edilen sonuçlar incelendiğinde çizelgeleme algoritması 1 %85,16 (511 / 600), çizelgeleme algoritması 2 %94,33 (566 / 600), çizelgeleme algoritması 3 %98,83 (593 / 600), çizelgeleme algoritması 4 %99,67 (598 / 600) ve çizelgeleme algoritması 5 %100,00 (600 / 600) oranında üçüncü test problemleri grubundaki test problemlerini optimum çözebilmiştir. Dolayısıyla çoklu ifade programlama tabanlı algoritma öğrenme sistemimiz toplam gecikme performans ölçütü göz önüne alınarak yapılan çizelgeleme algoritması keşfinde problemlerin karakteristiklerini ve değişimlerini öğrenebilmiş ve buna uygun olan çizelgeleme algoritmalarını keşfedebilmiştir.

BÖLÜM 6

GENEL SONUÇLAR

Bu çalışmada öncelik kuralları ve çizelgeleme algoritmalarının keşfedilmesi amacıyla çoklu ifade genetik programlama tabanlı bir öncelik kuralı öğrenme sistemi ve çizelgeleme algoritması öğrenme sistemi geliştirilmiş ve tek tezgahlı çizelgeleme problemleri üzerinde çalışılmıştır. Geliştirilen öğrenme sistemleri bütün üretim sistemlerine uygulanabilme kapasitesine sahip olmasına rağmen uygulamalarımız literatürde kullanılan tek tezgah ve tek ölçütlü OR Library test problemleri üzerinde yapılmıştır.

Çalışma iki ana aşamada gerçekleştirilmiştir. Birinci aşamada geliştirilen çoklu ifade genetik programlama tabanlı öncelik kuralı öğrenme sistemi ile işlerin toplam gecikmesi ölçütüne bağlı olarak, tek tezgahlı çizelgeleme problemleri üzerinde çalışılmıştır. Öncelik kurallarının geliştirilmesi için uygun terminal ve fonksiyon kümeleri belirlenmiştir. Bu terminal ve fonksiyon kümeleri öğrenme sistemi tarafından keşfedilecek olan öncelik kurallarının yapı taşlarını oluşturmaktadırlar. Geliştirdiğimiz öğrenme sistemi, belirlenmiş olan bu yapı taşlarını kullanarak probleme özel yapı taşlarını türetme ve kullanma yeteneğine sahiptir.

Geliştirdiğimiz öncelik kuralı öğrenme sistemi içerisinde standart seçim ve çaprazlama genetik operatörlerini kullanırken, yeni aday çözümlerin araştırılması sırasında bilginin rafine edilmesi bileşeni tarafından yönlendirilen bir mutasyon operatörü kullanmaktadır. Bu yönlendirme işleminde öğrenme sistemi ile geliştirilen öncelik kurallarının yapı taşlarının çözüme ne kadar katkısı olduğu belirlenmekte ve bu katkılardan yararlanarak çözüm uzayının daha bilinçli aranması sağlanmaktadır. Aday çözümlerin problemi ne kadar iyi çözebildiklerini belirlemek amacıyla üretim sistemi simülasyonu bileşeni kullanılmaktadır.

Geliştirdiğimiz öncelik kuralı öğrenme sisteminin öncelik kurallarını keşif için kullandığı test problemleri literatürde birçok araştırmacının kullandığı OR Library test problemleri türetme yöntemleri ile oluşturulmuştur. Oluşturulan test problemleri üzerinde yaptığımız çalışmalar sonucunda her bir test problemi için bilinen en iyi çözüm keşfedilen öncelik kuralları ile bulunabilmiştir. İzleyen aşamada oluşturulan test problemlerinin ortak karakteristiklerine uygun olarak çözüm sunabilecek öncelik kurallarının keşfedilebilmesi amacıyla test problemleri çeşitli şekillerde gruplanmıştır. Bu test problemleri alt kümeleri için öncelik kuralı geliştirme çalışmaları sonucunda keşfedilen öncelik kuralları sadeleştirilmiş ve en başarılı öncelik kuralları seçilmiştir. Seçilen öncelik kurallarının literatürde karşılaştırma amacıyla kullanılan klasik öncelik kurallarına göre oldukça iyi çözümler sunmaktadır. Ayrıca klasik öncelik kurallarının iş sayısının artması sonucunda eniyi çözüm değerinden sapma artışları keşfedilen öncelik kurallarında çok daha az olmaktadır.

Kullanılan test problemleri aynı problem türetme yöntemlerine göre oluşturulmasına rağmen işlerin toplam gecikmesi performans ölçütü göz önüne alındığında literatürde optimum çözüm veren bir öncelik kuralı ya da çözüm yöntemi bulunmamaktadır. Bunun nedenlerinden birisi olarak problemlerin farklı karakteristiklere sahip olduğu düşünülmüştür. Bu aşamada, keşfedilen öncelik kurallarının problem karakteristiklerindeki değişikliklerin üstesinden gelip gelemeyeceği araştırılmıştır. Araştırmalar sonucunda keşfedilen öncelik kurallarının birbirini tamamlayan şekilde ve işbirliği içerisinde kullanımının en iyi çözümlere ulaşma konusunda çok etkili olduğu görülmüştür. Aynı etki klasik öncelik kurallarının işbirliği içerisinde kullanımı ile sağlanamamaktadır. Böylece geliştirdiğimiz öğrenme sisteminin hem problemlerin en iyi çözümünü veren öncelik kurallarının keşfi için kullanılabileceğini hem de farklı problem karakteristiklerini göz önüne alarak birbirini tamamlayan öncelik kuralları keşfedebildiğini göstermektedir.

Çalışmamızın ikinci aşamasında çoklu ifade genetik programlama tabanlı bir çizelgeleme algoritması öğrenme sistemi geliştirilmiştir. Geliştirilen çizelgeleme algoritması öğrenme sistemi ile tek tezgahlı çizelgeleme problemlerinde geç kalan işlerin toplam sayısını enküçükleyen çizelgeleme algoritması keşfi, iki tezgahlı akış tipi üretim sistemlerindeki çizelgeleme problemlerinde en büyük tamamlanma zamanını

enküçükleyen çizelgeleme algoritması keşfi ve tek tezgahlı çizelgeleme problemlerinde işlerin toplam gecikme miktarını enküçükleyen çizelgeleme algoritması keşfi çalışmaları yapılmıştır. Bu çalışma çizelgeleme algoritmalarının otomatik keşfi konusunda literatürde yapıldığını bildiğimiz ilk çalışmadır.

Çizelgeleme algoritmalarının keşfedilmesi işlemi öncesinde tek tezgahlı çizelgeleme problemleri, iki tezgahlı çizelgeleme problemleri ve literatürde kullanılan akademik çalışmalar sonucu oluşturulmuş çizelgeleme algoritmaları incelenerek terminal ve fonksiyon kümeleri oluşturulmuştur. Bu terminal ve fonksiyon kümeleri öğrenme sistemi tarafından keşfedilecek olan çizelgeleme algoritmalarının yapı taşlarını oluşturmaktadırlar.

Geliştirdiğimiz çizelgeleme algoritması öğrenme sistemi tek nokta çaprazlama, tek nokta mutasyon standart genetik operatörlerini kullanırken, aday çözümlerin kodlandığı kromozom, algoritma yapısını ve akışını sağlayabilecek şekilde yeni geliştirilmiştir.

Tek tezgahlı çizelgeleme problemlerinde geç kalan işlerin toplam sayısını enküçükleyen çizelgeleme algoritması keşfi çalışmalarında elde ettiğimiz çizelgeleme algoritmaları her bir test problemi için bilinen en iyi çözümü elde etmeyi başarmıştır. Bu problem türü için literatürde Moore algoritması en iyi çözümü garanti etmektedir. Çalışmalarımızda bulduğumuz çizelgeleme algoritmalarının bir kısmı, algoritma adımları bazında tam olarak Moore algoritması ve onu andıran algoritmalar olmakla beraber bir diğer kısmı literatürde yer almayan çizelgeleme algoritmalarıdır.

İki tezgahlı akış tipi üretim sistemlerindeki çizelgeleme problemlerinde en büyük tamamlanma zamanını enküçükleyen çizelgeleme algoritması keşfi çalışmalarında elde ettiğimiz çizelgeleme algoritmaları her bir test problemi için bilinen en iyi çözümü elde etmeyi başarmıştır. Bu problem türü için literatürde Johnson algoritması en iyi çözümü garanti etmektedir. Çalışmalarımızda bulduğumuz çizelgeleme algoritmalarının bir kısmı, algoritma adımları bazında tam olarak Johnson algoritması ve onu andıran algoritmalar olmakla beraber bir diğer kısmı literatürde yer almayan çizelgeleme algoritmalarıdır.

Tek tezgahlı çizelgeleme problemlerinde işlerin toplam gecikme miktarını enküçükleyen çizelgeleme algoritması keşfi çalışmalarında elde ettiğimiz çizelgeleme algoritmaları her bir test problemi için bilinen en iyi çözümü elde etmeyi başarmıştır. Bu çizelgeleme algoritmaları arasında her problemde başarılı olan tek bir çizelgeleme algoritması bulunmamakta ve çeşitlilik arz etmektedir. Bu çeşitliliğin azaltılması ve eniyi çözümü bulma sayısı bakımından daha baskın çizelgeleme algoritmalarının keşfedilmesi amacıyla test problemleri çeşitli şekillerde gruplandırılmıştır. Elde edilen test problemleri alt kümeleri için ortak çizelgeleme algoritması keşfi işlemleri sonucunda beş adet çizelgeleme algoritması bulunmuştur. Bu çizelgeleme algoritmaları en iyi çözümü bulma konusunda %85,16 ile %100,00 arasındaki sonuçları ile kullandığımız test problemlerini çözmede oldukça başarılı olmuşlardır. Dolayısıyla çoklu ifade programlama tabanlı algoritma öğrenme sistemimiz çizelgeleme algoritması keşfinde kullanılan performans ölçütleri bazında problemlerin karakteristiklerini ve değişimlerini öğrenebilmiş ve buna uygun olan çizelgeleme algoritmalarını yüksek başarı ile keşfedebilmiştir.

Geliştirdiğimiz öğrenme sistemleri ile yaptığımız çalışmalar, ele aldığımız problemlerin çözümlerini elde etmekten ziyade bu problemlere çözüm türetecek öncelik kuralları ve çizelgeleme algoritmalarının keşfedilmesine yöneliktir. Dolayısıyla öğrenme sistemlerimiz öncelik kurallarının ve çizelgeleme algoritmalarının keşfedilmesi amacıyla bir altyapı sunmaktadır. Bu altyapının testleri genel olarak tek tezgahlı çizelgeleme problemleri üzerinde yapılmış olmasına rağmen çok tezgahlı her türlü çizelgeleme problemine farklı performans ölçütleri kullanılarak (tekli ve çoklu) uygulanabilir durumdadır. Bu uygulamalar yeni öncelik kuralları ve çizelgeleme algoritmaları keşfetme çalışmaları olabileceği gibi öğrenme sistemlerimizin önemli bir faydasının problemlerin sınıflandırılması aşamasında ortaya çıkacağını düşünmekteyiz. Keşfedilen öncelik kuralları ve çizelgeleme algoritmalarının yapısına göre problemlerin sınıflandırılması sağlanabilir. Ayrıca çizelgeleme ve optimizasyonun gerekli olduğu her türlü problemde (çok işlemci çizelgeleme, ağlarda paket yönetimi ve yönlendirmesi, lojistik, insansız araçlar vb.) öğrenme sistemlerimizin uygulamasının yapılabileceği görülmektedir.

KAYNAKLAR DİZİNİ

Aarts, B.J.M., 1996, A paralel local search algorithm for the job shop scheduling problem, Masters Thesis, Eindhoven University of Technology.

Abdul-Razaq, T.S., Potts, C.N, and Van Wassenhove, L.N., 1990, A survey of algorithms for the single machine total weighted tardiness scheduling problem, Discrete Applied Mathematics, Vol:26, 235-253.

Adriaans, P., and Zantige, D., 1996, Data Mining, Harlow: Addison-Wesley.

Alikalfa, M., and Kapanoğlu, M., 2010, A Set of Priority Rules for Minimizing Total Tardiness on Single Machines, IMS'2010, 10th International Symposium on Intelligent Manufacturing Systems – Progress in Intelligent Technologies for Society, 2010, Sarajevo, Bosnia Herzegovina.

Alpar, P. and Srikanth, R., 1989, Closed-shop scheduling with expert systems techniques, In A. Kusiak (ed.), Knowledge-Based Systems in Manufacturing, Taylor & Francis.

Arizono, I., Yamamuto, A., and Ohta, H., 1992, Scheduling for minimizing total actual flow time by neural Networks, International Journal of Production Research, Vol:30, 503-511.

Armentano, V.A., 2000, Tabu search for minimizing total tardiness in a job shop, International Journal of Production Economics, Vol:63, 131-140.

Aytug, H., Bhattacharyya, S., and Koehler, G.J., 1998, Genetic learning through simulation: An investigation in shop flor scheduling, Annals of Operations Research, 78.

Aytug, H., Koehler, G.J., and Snowden, P., 1994, Genetic learning of dynamic scheduling within a simulation environment, Computers & Operations Research, Vol 21 No:8, 909-925.

Baker, K., 1974, Introduction of sequencing and scheduling, John Wiley & Sons.

Barnes, J.W., and Chambers, J.B., 1995, Solving the job shop scheduling problem using tabu search, IEE Transactions, Vol:27, 257-263.

Bel, E., Bensana, E., Dubois, D., Erscher, J., and Esquirol, P., 1989, A knowledge-based approach to industrial job-shop scheduling (OPAL), In A. Kusiak (ed.), Knowledge-Based Systems in Manufacturing, Taylor & Francis, 207-246.

Bhaskaran, K., and Pinedo, E., 1991, Dispatching, In G. Salvendy (ed.), Handbook of Industrial Engineering, John Wiley & Sons.

KAYNAKLAR DİZİNİ (devam)

Biegel, J.E., and Davem, J.J., 1990, Genetic Algorithms and Job Shop Scheduling. Computers and Industrial Engineering, vol.19, No.1-4.

Blackstone, J., Phillips, D., and Hogg, G., 1982, A state-of-the-art survey of dispatching rules for manufacturing job shop operations, International Journal of Production Research, Vol:20, No:1, 27-45.

Blazewics, J., Eckler, K.H., Schmith, G., and Wedlarz, J., 1994, Scheduling in Computer and Manufacturing Systems, Second Ed., Springer-Verlag Publications.

Bruns, R., 1997, Chapter 1.5: Applications of Evolutionary Computation: Scheduling, in Bäck, T., Fogel, D., and Michalewicz, Z. (eds): Handbook of Evolutionary Computation, Oxford University Press.

Burgin, M., 2005, Super-Recursive Algorithms, Springer.

Cerny, V., 1985, Thermodynamical Approach to the travelling salesman problem: An efficient simulation algorithm, Journal of Optimization Theory and Applications, Vol:40, 41-51.

Chang, Y.L., Sueyoshi, T., and Sullivan, R.S., 1996, Ranking dispatching rules by data envelopment analysis in a job-shop environment, IIE Transactions, Vol:28 No:8, 631-642.

Chen, CC, and Yih, Y., 1996, Identifying attributes for knowledge-based development in dynamic scheduling environments. International Journal of Production Research., 34(6):1739–55.

Chen, Z., 2001, Data Mining And Uncertain Reasoning, A Wiley-Interscience Publication, USA.

Chen, H., and Chengbin, C., 1995, A more efficient lagrangian relaxation approach to job shop scheduling problems, IEEE International Conference on Robotics and Automation, 496-501.

Cheng, R., Gen, M., and Tsujimura, Y., 1996, A tutorial survey of job-shop scheduling problems using genetic algorithms, part I: representation, Computers & Industrial Engineering, Vol:30 No:4, 983-997.

Cheng, R., Gen, M., and Tsujimura, Y., 1999, A tutorial survey of job-shop scheduling problems using genetic algorithms, part II: hybrid genetic search strategies, Computers & Industrial Engineering, Vol:36, 343-364.

KAYNAKLAR DİZİNİ (devam)

Chiang, T.W., 1996, Solving job insertion problem in job shop scheduling using iterative improvement, Working paper, National Taiwan University.

Chou, F.D., 2009, An experienced learning genetic algorithm to solve the single machine total weighted tardiness scheduling problem, Expert systems with Applications, Vol.36, 3857-3865.

Copas, C., and Browne, J., 1990, Rule-based scheduling system for flow type assambly, Intenational Journal of Production Research, Vol:28 No:5, 981-1005.

Corcoran, A.L., and Sen, S., 1994, Using real-valued genetic algorithms to evolve rule sets for classification, Proc. IEEE Conference on Evolutionary Computation, 120-124.

Crowston, W.B, Glover, F., Thompson, G.L., and Trawick, J.D., 1963, Probabilistic and parametric learning combinations of local job-shop scheduling rules, ONR Research Memorandum No. 117, Carnegie Institute of Technology.

De Jong, K.A., 1975, An analysis of the behavior of a class of genetic adaptive systems, Doctoral Dissertation, University of Michigan, Dissertation Abstarct International, 36(10), 5140B.

De Jong, K.A., 1990, Genetic algortihm based learning, in Y. Kodratoff ve R. Michalski (eds), Machine Learning, Volme III, Morgan Kaufmann.

De Jong, K.A., and Spears, W.M., 1991, Learning concept classification rules using genetic algorithms, In proceedings of the 1991 International Joint Conference on artificial intelligence, 651-656.

Dell'Amico, M., and Trubian, M., 1993, Applying tabu search to the job-shop scheduling problem, Annals of Operations Research, Vol:41, 231-252.

Deshpande, M., and Karypis, G., 2002, "Using conjunction of attribute values for classification," CIKM'02, Nov. 4–9, McLean, VA, 356–364.

Dimopoulos, C., and Zalzala, A.M.S., 2001, Investigating the use of genetic programming for a classic one-machine scheduling problem. Advances in Engineering Software, 32(6), 489–498.

Dorndorf, U., and Pesch, E., 1995, Evolution based learning in a job-shop scheduling environment, Computers and Operations Research, Vol:22 No:1, 25-40.

KAYNAKLAR DİZİNİ (devam)

Fisher, H., and Thompson, G.L., 1963, Probabilistic learning combinations of local job-shop scheduling rules, in J.F. Muth and G.L. Thompson (eds.), Industrial Scheduling, Prentice Hall, 225-251.

Fox, M.S., and Smith, S.F., 1984, ISIS: A knowledge-based system for factory scheduling, Expert systems, Vol:1 No:1, 25-49.

French, S., 1982, Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop, John Wiley & Sons.

Geiger, C.D., Uzsoy, R., and Aytug, H., 2003, Autonomous learning of effective dispatch policies for flowshop scheduling problems, in Proceedings of the Industrial Engineering Research Conference (IERC'03).

Geiger, C.D., Uzsoy, R., and Aytug, H., 2006, Rapid Modeling and Discovery of Priority Dispatching Rules: An Autonomous Approach, Journal of Scheduling 9: 7–34.

Gen, M., and Cheng, R., 1997, Genetic algorithms and engineering design, John Wiley & Sons.

Gere, W.S.Jr., 1966, Heuristics in job-shop scheduling, Management Science, Vol:13, 167-190.

Giffler, B., and Thompson, G.L., 1960, Algorithms for solving production scheduling problems, Operations Research, Vol:8 No:4, 487-503.

Goldberg, D.E., 1989, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley.

Grabot, B., and Geneste, L., 1994, Dispatching rules in scheduling: A fuzzy approach, International Journal of Production Research, Vol:32 No:4, 903-915.

Han, J., and Kamber, M., 2001, Data Mining: Concept and Techniques. Morgan Kaufmann Publishers, San Francisco, 24-703.

Hancock, P.J.B., 1994, An empirical comparison of selection in evolutionary algorithms, Department of Psychology University of Stirling, Scotland.

Hara, H., 1995, Job-shop scheduling by minimal conflict search, Japanese Arfificial Intelligence Society 10/1, 88-93.

Haupt, R., 1989, A survey of priority rule based scheduling, OR Spektrum 11, 3-16.

KAYNAKLAR DİZİNİ (devam)

Hershauer, J., Karim, A., Owens, H., and Philippakis, A., 1989, A Field observation study of an expert system prototype development, Information and Management, Vol:17 No:2, 107-116.

Hilliard, M.R., Liepins, G.E., and Palmer, M., 1988, Machine learning applications to job shop scheduling, International conference on Industrial and engineering applications of artificial intelligence and expert systems archive, Proceedings of the first international conference on Industrial and engineering applications of artificial intelligence and expert systems, Vol:2, 728 – 737.

Holland, H., 1975, Adaptation in Natural and Artificial Systems, University of Michigan Press. Ann Arbor.

Holland, H., 1986, Escaping brittleness: the possibilities of general purpose learning algorithms applied to paralel rule-based systems, in R. Michalski, J. Carbonell ve T. M. Mitchell (eds), Machine Learning, an artificial intelligence approach: Volume II, Morgan Kaufmann.

Holsapple, C.W., Jacop, V.S., Pacart, R., and Zaveri, J.S., 1993, A genetic-based hybrid scheduler for generating static schedules in flexible manufacturing contexts, IEEE Trans. on System, Manufacturing and Cybernetics, Vol:23 No:4, 953-971.

Holthaus, O., and Ziegler, H., 1997, Improving job shop performance by coordinating dispatching rules, International Journal of Production Research, Vol:35 No:2, 539-549.

Jackson, J.R., 1955, Scheduling a production linet o minimise maximum tardiness, Research Report 43, Management Science Research Projects, University of California.

Jones, A., and Rabelo, L.C., 1998, Survey of Job Shop Scheduling Techniques, National Institute of Standards and Technology, Washington.

Jones, A., Rabelo, L., and Yih, Y., 1995, Hybrid approach for real-time sequencing and scheduling, International Journal of Computer Integrated Manufacturing, Vol:8 No:2, 145-154.

Kapanoğlu, M., and Alikalfa, M., 2010, Multi Expression Programming for Designing Priority Rules in Machine Scheduling, IMS'2010, 10th International Symposium on Intelligent Manufacturing Systems – Progress in Intelligent Technologies for Society, 2010, Sarajevo, Bosnia Herzegovina.

KAYNAKLAR DİZİNİ (devam)

Kapanoglu, M., and Alikalfa, M., 2008, A Hyper-Genetic Algorithm for Computer-Integrated Job Shop Scheduling with Due Date-based Objectives, Proceedings of the 13th Annual International Conference on Industrial Engineering Theory, Applications and Practice, p611-617.

Kapanoglu, M., and Alikalfa, M., 2006, Developing Scheduling Policies In Dynamic Job Shops Using Pitts-Based Learning, GECCO 2006, Late breaking paper at Genetic and Evolutionary Computation Conference, 8-12 July, 2006, Seattle, WA, USA.

Kapanoglu, M., and Alikalfa, M., 2005, Interval-based Scheduling for Dynamic Job Shops using Genetics-based Machine Learning, Proceedings of the 10th Annual International Conference on Industrial Engineering - Theory, Applications & Practice, 4-7 December, 2005, Clearwater, Florida, USA.

Kapanoglu, M., and Alikalfa, M., 2004, Genetics-based machine learning in job shop scheduling, IMS'2004, 4th International Symposium on Intelligent Manufacturing Systems, 2004, Sakarya, Turkey, p1049-1056.

Kapanoğlu, M., Miller, W.A., and Sunol, A.K., 2001, Concurrent planning and scheduling of flexible manufacturing systems by using parallel genetic algorithms, Proceedings of the ICC&IE/IEMS 2001, 780-785.

Kapanoğlu, M., and Miller, W.A., 2004, An evolutionary algorithm-based decision support system for managing flexible manufacturing, Robotics & Computer Integrated Manufacturing, Ref. No: RCM451.

Kelly, J.D., and Davis, L., 1991, A hybrid genetic algorithm for classification, in proceedings of the international joint conference on artificial intelligence, 645-650.

Kerr, R.M., 1992, Expert system in production scheduling: Lessons from a failed implementation, Journal of Systems and Software, Vol:19 No:2, 123-130.

Kerr, R.M., and Ebsary, R.V., 1988, Implementation an expert system for production scheduling, European Journal of Operational Research, Vol:33 No:1, 17-29.

Kirkpatrick, S., Gelatt, C.D.Jr., and Vecchi, M.P., 1983, Optimization by simulated annealing, Science, Vol:220 No:4598, 671-680.

Krishna, K., Ganeshan, K., and Janaki, Ram D., 1995, Distributed simulated annealing algorithms for job shop scheduling, IEEE Transactions on Systems, Man and Cybernetics, Vol:25 No:7, 1102-1109.

Krucky, J., 1994, Fuzzy family setup assignment and machine balancing, Hewlett-Packard Journal, 51-64.

KAYNAKLAR DİZİNİ (devam)

Kobu, B., 2006. Üretim Yönetimi. Beta Yayınevi, 13.Baskı, 602s., Eylül 2006, İstanbul.

Kolonko, M., 1999, Some new results on simulated annealing applied to the job shop scheduling problem, European Journal of Operations Research, Vol:113, 123-136.

Koza, J., 1992, Genetic Programming: On the Programming of Computers by Means of Natural Selection,MIT Press, Cambridge, 1992.

Koza, J., 1994, Genetic Programming II: Automatic Discovery of Reusable Programs.MIT Press, Cambridge, MA, USA.

Köksalan, M., and Keha, A.B., 2003, Using genetic algorithms for single-machine bicriteria scheduling problems, European Journal of Operational Research 145, 543-556.

Lageweg, B.J., Lenstra, J.K., and Rinnooy Kan, A.H.G., 1977, Job shop scheduling by implicit enumeration, Management Science, Vol:24, 441-450.

Laguna, M., Barnes, J.W., and Glover, F.W., 1993, Intelligent scheduling with tabu search: an application to jobs with linear delay penalties and sequence-dependent setup costs and times, Journal of Applied Intelligence 3, 159-172.

Laguna, M., Barnes, J.W., and Glover, F.W., 1991, Tabu search methods for a single machine sheduling problem, Journal of Intelligence Manufacturing 2, 63-74.

Lawrance, S., 1984, Supplement to resource constrained Project scheduling: an experimental investigation of heuristic scheduling techniques, Graduate School of Industrial Administration, Carnegie-Mellon University.

Lee, J., and Moon, I., 2000, Genetic algorithm application to the job shop scheduling problem with alternative routings, Working Paper, Pusan National University.

Lee, S.E., and O'Keefe, R.M., 1994, Developing a strategy for expert system verification and validation, IEEE Trans. on Sys., Man. and Sybernetics, Vol:24 No:4, 643-655.

Lee, C.Y., Piramuthu, S., and Tsai, Y.K., 1997, Job shop scheduling with a genetic algorithm and machine learning, International Journal of Production Research, Vol:35 No:4, 1171-1191.

Lesh, N., Zaki, M.J., and Ogihara, M., 1999, "Mining features for sequence classification," in 5th ACMSIGKDD International Conference on Knowledge Discovery and Data Mining.

KAYNAKLAR DİZİNİ (devam)

Li, X., and Olafsson, S., 2005, Discovering Dispatching Rules using Data Mining, Journal of Scheduling 8: 515–527.

Liu, J.S., and Sycara, K., 1996, Multiagent Coordination in Tightly Coupled Task Scheduling, International Conference on Multi-Agent Systems.

Luh, P.B., Hoitomt, D.J., Max, E., and Pattipati, K.P., 1990, Scheduling generation and reconfiguration for paralel machines, IEEE Trans. Robotics and Automation, Vol:6, 687-696.

Matsuo, H., Suh, C.J., and Sullivan, R.S., 1988, A controlled search simulated annealing method for the general job-shop scheduling problem, Working paper, The University of Texas at Austin.

Maza, M.D., and Tidor, B., 1991, Increased Flexibility in Genetic Algorithms: The Use of Variable Boltzmann Selective Pressure to Control Propagation, Proc. of the ORSA CSTS Conference - Computer Science and Operations Research: New Developments in their Interfaces, pp 425-440.

Metropolis, N., Rosenbluth, A.W., Teller, M.N,. and Teller, E., 1953, Equation of state calculations by fast computing machines, The Journal of Chemical Physics, Vol:21 No:6, 1087-1092.

Michalski, R.S., 2003, Sanken Symposium on Data mining and Semantic Web, Osaka University, Japan.

Montazer, M., and Wassenhove, V., 1990, Analysis of scheduling rules for an FMS, International Journal of Production Research, Vol:28, 785-802.

Nakasuka, S., and Yoshida, T., 1992, "Dynamic scheduling system utilizing machine learning as a knowledge acquisition tool," International Journal of Production Research, 30(2), 411–431.

Ntuen, C.A., and Park, E.H., 1995, An experiment in scheduling and planning of non-structured jobs: Lessons learned from artificial intelligence and operational research toolbox, European Journal of Operational Research, Vol:84, 96-115.

Orciuch, E., and Frost, J., 1984, ISA: Intelligent scheduling assistant, IEEE, 314-320.

Oltean, M., 2006, Multi Expression Programming, Technical Report, Babes-Bolyai Univ, Romania.

Oltean, M., 2007, Evolving Evolutionary Algorithms with Patterns, Soft Computing, Springer-Verlag Vol. 11, Issue 6, pp. 503-518.

KAYNAKLAR DİZİNİ (devam)

Panwalker, S., and Iskander, W., 1977, A survey of scheduling rules, Operations Research, Vol:25 No:1, 46-61.

Parunak, H., Irish, B., Kindrick, J., and Lozo, P., 1985, Fractal actors for distributed manufacturing control, Proceedings of the Second IEEE Conference on Artificial Intelligence Applications, 653-660.

Pinedo, M., 1995, Scheduling: Theory, Algorithms and Systems, Prentice Hall.

Piramuthu, S., Raman, N., and Shaw, M.J., 1994, Learning-based scheduling in a flexible manufacturing flow line, IEEE Transactions on Engineering Management, 41(2), 172–182.

Piramuthu, S., Raman, N., Shaw, M.J., and Park, S., 1993, Integration of simulation modeling and inductive learning in an adaptive decision support system, Decision Support Systems, 9, 127–142.

Pezzella, F., and Merelli, E., 2000, A tabu search method guided by shifting bottleneck fort he job shop scheduling problem, European Journal of Operations Research, Vol:120, 297-310.

Pham, D.T., and Pham, P.T.N., 1988, Expert system in mechanical and manufacturing engineering, International Journal of Advanced Manufactuing Technology, Vol:3 No:3, 3-21.

Quinlan, J., 1986, Induction of decision trees, Machine Learning, Vol:1, 81-106.

Rao, H.R., and Lingaraj, B.P., 1988, Expert system in production and operations management: classification and prospects, Interfaces, Vol:18 No:6, 80-91.

Rabelo, L.C., and Alptekin, S., 1989, Integrating scheduling and control functions in computer integrated manufacturing using artificial intelligence, Computers & Industrial Engineering, Vol:17 No:1-4, 101-106.

Russell, S., and Norvig, P., 1995, Artificial Intelligence: A Modern Approach, Prentice Hall.

Sabuncuoğlu, I., and Gürgün, B., 1996, A neural network model for scheduling problems, European Journal of Operation Research, Vol:93, 288-299.

Schuh, G., 2008. Produktionsmanagements I, WZL/FIR, Acchen Universitat.

KAYNAKLAR DİZİNİ (devam)

Shaw, M.J., Park, S., and Raman, N., 1992, Intelligent scheduling with machine learning capabilities: the induction of scheduling knowledge, IIE Transactions, 24(2), 156-168.

Sim, S.K., Yeo, K.T., and Lee, W.H., 1994, An expert neural network system for dynamic job shop scheduling, International Journal of Production Research, Vol:32 No:8, 1759-1773.

Sivanandam, S.N., and Deepa, S.N., 2008, Introduction to Genetic Algorithms, Springer.

Slany, W., 1994, Scheduling as a fuzzy multiple criteria optimization problem, Technical Report 94/62, Technical University of Vienna.

Smith, W.E., 1956, Various optimizers for single stage production, Naval Research Logistics Quarterly 3, 59-66.

Smith, S.F., 1980, A learning system based on genetic adaptive algortihms, PhD thesis, University of Pittsburg.

Smith, F.S, Ow, P.S., Potwin, J., Muscettola, N., and Matthys, D.C., 1990, An integrated framework for generating and revising factory schedules, Journal of the Operational Research Society, Vol:41, 539-552.

Steger-Jensen, K., 2003, Introduction to Scheduling and Scheduling Systems, Cenertrykkeriet, Aalborg University.

Tamaki, H., Ochi, M., and Araki, M., 1998, Genetics-based machine learning approach to production scheduling : a case of in-tree type precedence relation, ISIE-1998.

Tamaki, H., Sakakibara, K., Murao, H., and Kitamura, S., 2003, Rule acquisition for production scheduling, SICE-2003.

Taşgetiren, M.F., Cedimoğlu, İ.H. ve Öztemel E., 1995, Sinir ağı çizelgeleme sistemi: Uzman çizelgeleme sisteminden esinlenen bir yaklaşım, YA/EM'95 Bildiri, ODTÜ.

Tay, J.C., and Ho, N.B., 2008, Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems, Computers & Industrial Engineering 54, 453–473.

Thomsen. S., 1997, Metaheuristics combined with branch & bound, Technical report, Copenhagen Business School.

KAYNAKLAR DİZİNİ (devam)

Tsujimura. Y.S, Park. S., Chang. S.. and Gen. M., 1993, An effective method for solving flow shop scheduling problems with fuzzy processing times, Computers & Industrial Engineering, Vol:25, 239-242.

Turing. A.M., 1939, Systems of Logic Based on Ordinals, Proceedings of the London Mathematical Society, Series 2:45, 161-228.

Van Laarhooven. P.J.M., Aarts. E.H.L.. and Lenstra. J.K., 1992, Job shop scheduling by simulated annealing, Operations Research, Vol:40 No:1, 113-125.

Wang, L.C., Chen, H.M., and Liu, C.M., 1995, Intelligent scheduling of FMSs with inductive learning capability using neural networks, International Journal of Flexible Manufacturing Systems, 7:147-175.

Wesley, J., 1995, Solving the JJSP with tabu search, IEE Transactions, Vol:27, 263-269.

Wilkerson, L., and Irwin, J., 1971, An improved algorithm for scheduling independent tasks, AIIE Transactions, Vol:3, 239-245.

Yamada, T., Rosen, B.E., and Nakano, R., 1994, A simulated annealing approach to job shop scheduling using critical block transition operators, IEEE ICNN'94 International Conference on Neural Networks, Vol:6, 4687-4692.

Yevich, R., 1997, Data Mining, Englewood Cliffs, Prentice-Hall, N.J., 309-321.

Yih, Y., 1990, Trace-driven knowledge acquisition (TDKA) for rule-based real-time scheduling systems, Journal of Intelligence Manufacturing, Vol:1 No:4, 217-230.

ÖZGEÇMİŞ

Mete Alikalfa 8 Mayıs 1977 tarihinde Köln'de (Almanya) doğmuştur. Lisans öğrenimine 1996 yılında Eskişehir Osmangazi Üniversitesi Mühendislik Mimarlık Fakültesi Endüstri Mühendisliğinde başlamış ve 2000 yılında mezun olmuştur. Yüksek lisans öğrenimini 2004 yılında Eskişehir Osmangazi Üniversitesi Endüstri Anabilim Dalı, Endüstri Bilim Dalında "Genetik Tabanlı Otomatik Öğrenmeye Dayalı Bir Atölye Çizelgeleme Sistemi" başlıklı teziyle tamamlamıştır. İş hayatına 2000 yılında Eskişehir Osmangazi Üniversitesi Mühendislik Mimarlık Fakültesi Bilgisayar Mühendisliğinde başlamış ve halen aynı bölümde akademik çalışmalarına devam etmektedir.