T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

EŞİT OLMAYAN HAZIRLAMA SÜRELİ TEK MAKİNE TOPLAM AĞIRLIKLI GECİKME PROBLEMLERİNİN PARÇACIK SÜRÜ OPTİMİZASYONU İLE ÇÖZÜMÜ

YÜKSEK LİSANS TEZİ

Serdar ÖZER

Enstitü Anabilim Dalı : **ENDÜSTRİ MÜHENDİSLİĞİ**

Tez Danışmanı : **Doç. Dr. Tarık ÇAKAR**

Mart 2017

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

EŞİT OLMAYAN HAZIRLAMA SÜRELİ TEK MAKİNE TOPLAM AĞIRLIKLI GECİKME PROBLEMLERİNİN PARÇACIK SÜRÜ OPTİMİZASYONU İLE ÇÖZÜMÜ

YÜKSEK LİSANS TEZİ

Serdar ÖZER

Enstitü Anabilim Dalı **ENDÜSTRİ MÜHENDİSLİĞİ**

Bu tez 08 / 03 / 2017 tarihinde aşağıdaki jüri tarafından Oybirliği / Oyçokluğu ile kabul edilmiştir.

Doç. Dr. Tarık ÇAKAR	Prof. Dr. Raşit KÖKER	Yrd. Doç. Dr. Celal ÖZKALE
Jüri Başkanı	Üye	Üye

BEYAN

Tez içindeki tüm verilerin akademik kurallar çerçevesinde tarafımdan elde edildiğini, görsel ve yazılı tüm bilgi ve sonuçların akademik ve etik kurallara uygun şekilde sunulduğunu, kullanılan verilerde herhangi bir tahrifat yapılmadığını, başkalarının eserlerinden yararlanılması durumunda bilimsel normlara uygun olarak atıfta bulunulduğunu, tezde yer alan verilerin bu üniversite veya başka bir üniversitede herhangi bir tez çalışmasında kullanılmadığını beyan ederim.

Serdar ÖZER

08/03/2017

TEŞEKKÜR

Yüksek lisans eğitimim boyunca, bana desteğini her zaman hissettiğim, bilgi ve deneyimlerinden yararlandığım, sürekli olarak beni motive eden araştırmanın bütün süreçlerinde yer alan ve beni yönlendiren değerli danışman hocam Doç. Dr. Tarık ÇAKAR'a ve Sakarya Üniversitesi Fen Bilimleri Enstitüsü Endüstri Mühendisliği Anabilim Dalı öğretim üyelerine teşekkür ve saygılarımı sunarım.

Yüksek lisans serüvenimin her dakikasında yanımda olduğunu hep hissettiğim sevgili eşim ve hayat arkadaşım Burcu ÖREN ÖZER'e, tezimin son senelerinde dünyaya gelerek beni ayrı bir motivasyona sürükleyen sevgili oğlum Rüzgar Ali ÖZER'e hayatımıza kattığı neşe, enerji ve mutluluk için teşekkür ederim.

İÇİNDEKİLER

SİMGE VE KISALTMALAR LİSTESİ

CR	: Kritik Oran
EDD	: En Erken Teslim Zamanı
FCFS	: İlk Gelen İlk İşlem Görür
GA	: Genetik Algoritma
IEEE	: Uluslararası Sinir Ağları Konferansı
LPT	: En Uzun İşlem Süresi
MDD	: Modifiye Teslim Tarihi
MST	: En Küçük Bolluk
PSO	: Parçacık Sürü Optimizasyonu
RDD	: Teslim Tarihi Oranlı Aralık
SMTWT	: Tek Makine Toplam Ağırlıklı Gecikme
SPT	: En Küçük İşlem Süresi
TFF	: Ortalama Gecikme Faktörü
WDD	: Ağırlıklandırılmış Teslim Zamanı
WMDD	: Ağırlıklandırılmış Modifiye Teslim Tarihi
WPD	: Ağırlandırılmış Proses Teslim Zamanı
WSPT	: Ağırlıklandırılmış En Küçük İşlem Süresi

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

-

ÖZET

Anahtar Kelimeler: Tek Makine Çizelgeleme ve Sıralama, Parçacık Sürü Optimizasyonu, Toplam Ağırlıklı Gecikme, Öncelik Kuralları.

Verimliliğin ve etkinliğin son derece önemli olduğu günümüz rekabet ortamında, işletmeler rekabet avantajı sağlamak için hıza önem vermektedirler. Siparişlerin istenilen zamanda ve en az gecikme ile teslim edilmesi bütün işletmelerin ortak amaçları arasında yer almaktadır. Çizelgeleme işlemlerinin yapısındaki çeşitlilik ve karmaşıklık nedeniyle birçok çizelgeleme kuralı ve algoritma oluşturulmuştur. Çoğu algoritma kabul edilebilir yaklaşık sonuçlar bulmakta, bazıları ise optimum çözüm sunmaktadır.

Parçacık Sürü Optimizasyonu (PSO) kuş sürüleri arasındaki sosyal etkileşimden yola çıkılarak oluşturulmuş bir optimizasyon tekniğidir. PSO, uygulamadaki kolaylık ve hızlı çözüm bulma özellikleri ile diğer optimizasyon teknikleri arasından ön plana çıkmaktadır.

Bu çalışmada Parçacık Sürü Optimizasyonu (PSO) Algoritması kullanılarak Eşit Olmayan Hazırlama Süreli Tek Makine Toplam Ağırlıklı Gecikmenin optimum şekilde çözümlenmesi açıklanmıştır. Elde edilen sonuçlar toplam ağırlıklı gecikmenin PSO algoritması kullanılarak azaltılacağını göstermiştir

PARTICLE SWARM OPTIMIZATION APPROACH TO SOLVE SINGLE MACHINE TOTAL WEIGHTED TARDINESS PROBLEM WITH UNEQUAL RELEASE DATE

SUMMARY

Keywords: Single Machine Scheduling and Sequencing, Particle Swarm Optimization, Total Weighted Tardiness, Priority Rules.

In today's competitive environment, where efficiency and effectiveness are extremely important, businesses value speed to provide competitive advantage. Delivery of orders at the desired time and with minimum delay is among the common goals of all enterprises. Due to the diversity and complexity of the structure of scheduling processes, many scheduling rules and algorithms have been created. Most algorithms find acceptable results, while others offer the optimum solution.

Particle Swarm Optimization (PSO) is an optimization technique based on the social interaction between bird flocks. PSO with the ease at application and speed of finding a solution comes to the forefront among the other optimization techniques.

In this study, the optimal solution of Unequal Preparation Time Single Machine Total Weighted Delay using Particle Swarm Optimization (PSO) Algorithm is explained. The results show that the total weighted delay will be reduced by using the PSO algorithm.

BÖLÜM 1. GİRİŞ ve LİTERATÜR TARAMA

1.1. Giriş

Doğada bulunan sürülerin davranışları keşfedildikten sonra, bu sürülerin davranışlarının modellenmesi üzerine birçok çalışma yapılmıştır. Özellikle kuş, karınca, balık, arı gibi sürülerin davranışları aralarındaki iletişim mantığı ve yiyecek aramada kullandıkları yollar bilgisayar yazılımcıları tarafından çeşitli şekillerde modellenmiştir.

Günümüzde kuş sürülerinden esinlenilerek oluşturulmuş yöntemler hesaplama problemlerinin çözümünde kullanılmaktadır. Bu tip sosyal sistemler özellikle sürüye üye olan bireylerin çevresiyle ve diğer bireylerle olan iletişimlerini ve davranışlarını ele alarak problemlerin çözümüne gidilmiştir.

Sürü zekası olarak adlandırılan bu yaklaşımların optimizasyon problemlerinde başarı elde etmesinden dolayı yapılan çalışmaların sayısı artmıştır. Sürü zekası bireylerin diğer üye bireyler ile birlikte kolektif bir şekilde zeka geliştirmesi olarak tanımlanabilir. Bu zekanın oluşmasında hem çevresi ile hem de diğer üyeler ile etkileşimde bulunması önemli bir etkendir. Kesin kurallar olmadan sürünün davranışı zeki bir sistemi yaratmış olur. Sürü üyelerinin tek başlarına yapmaları mümkün olmayan işlemler sürüdeki koordinasyon ve iletişim sayesinde gerçekleştirilmiş olur.

Sürü içerisindeki bir üyenin hata yapması durumunda, sürünün geniş olmasından dolayı bu hata diğer üyeler tarafından telafi edilir. Buda sistemi devamlı iyiye götürecek bir yaklaşımdır.

Parçacık Sürü Optimizasyonu (PSO), 1995 yılında J.Kennedy ve R.C. Eberhart tarafından kuş sürülerinin davranışlarından esinlenerek geliştirilmiş popülasyon tabanlı stokastik bir optimizasyon tekniğidir. Genetik algoritmalar gibi evrimsel hesaplama yöntemlerine benzemektedir.

PSO algoritması klasik optimizasyon türlerine göre hesaplanması ve çözüm adımları görece olarak basit bir algoritmadır ve türev bilgisine ihtiyaç duymamaktadır. PSO'da parçacık olarak adlandırılan olası çözümler çözüm uzayında dolaşarak en uygun çözümü ararlar. Rassal olarak başlatılan PSO en uygun çözüme en yakın parçacığa çözüm adımları uygulanarak yaklaşmaya çalışırlar. Bu sayede birbirleri ile iletişim kurarak sürü zekası oluşturulmuş olur. Parçacık Sürü Optimizasyonu 'nun literatür araştırması aşağıda sunulmuştur.

1.2. Literatür Tarama

Birçok araştırma – geliştirme, analiz, fabrikasyon, satış gibi faaliyet alanını içeren mühendislik bilimi, sadece çalışan bir sistemi geliştirmek için değil aynı zamanda "en iyi" sistemi geliştirmek için çalışmalar yapıldığı bir bilim dalıdır. En iyi olarak nitelendirilebilecek konular en verimli, en hızlı, çok fonksiyonlu, dayanıklı vb. olabilir. Bu sayede bir sistemin dizaynı optimizasyon problemi olarak formüle edilip çözülebilir. Optimizasyon problemi en iyi ile kastedilen konuları amaçlayarak, matematiksel bir fonksiyon ile temsil edilerek, belirtilen sınırlar dahilinde bu fonksiyonların amaca uygun olarak çözülmesidir. Bu tip bir optimizasyon probleminin iç önemli bileşeni mevcuttur. Bunlar; değişkenler, sınırlayıcılar ve amaç fonksiyonudur (Yüksel ve Ülker 2008).

Parçacık Sürü Optimizasyonu (PSO), 1995 yılında J.Kennedy ve R.C.Eberhart tarafından; kus sürülerinin davranışlarından esinlenilerek geliştirilmiş popülasyon tabanlı stokastik optimizasyon tekniğidir. Tasarlanmasındaki amaç doğrusal olmayan problemlerin çözümü içindir. Ayrıca çok değişkenli ve çok parametreli optimizasyon problemleri için de kullanılmaktadır. PSO yapısı gereği genetik algoritmalar gibi evrimsel hesaplama yöntemleri ile benzerlik gösterir. Çözüm rasgele çözümler ile

başlatılır ve itersyonlar ile güncellenerek optimum çözüme ulaşmaya çalışır. PSO da yer alan olası çözümler parçacık olarak adlandırılır ve o andaki optimuma en yakın parçacığı izleyerek çözüm uzayında dolaşırlar. PSO'nun Klasik optimizasyon yöntemlerinden en önemli farkı çözüm adımlarında türev bilgisine ihtiyaç duymamasıdır. Uygulanması basit olan PSO parametresinin az olması nedeniyle tercih edilen bir algoritmadır. PSO fonksiyon optimizasyonu, yapay sinir ağı eğitimi, bulanık sistem kontrolü gibi birçok alanda kullanılmaktadır (Tamer ve Karakuzu 2006).

PSO kuş sürülerinden esinlenilerek oluşturulan bir benzetimdir. Kuşların yiyecek aramaları çözüm uzayında en uygun çözümü bulmaya benzetilir. Kuş sürüleri yiyecek ararken yiyeceğe en yakın kuşu takip ederler. PSO 'da parçacık olrak adlandırılan tekil çözümler çözüm uzayında bir kuştur. Parçacık hareket ettiğinde, koordinatlarını bir fonksiyona gönderir ve uygunluk değeri ölçülmüş olur. Parçacıklar hızlarını, koordinatlarını, şimdiye kadarki en iyi uygunluk değerini ve koordinatlarını hatırlamalıdır. Çözüm uzayındaki her iterasyonda hızının, yönünün nasıl değişeceğini, diğer kuşların ey iyi koordinatları ve kendi kişisel koordinatlarının bir bileşimi olarak hesaplanacaktır (Tamer ve Karakuzu 2006).

Parçacık Sürü Optimizasyon algoritması popülasyon temelli bir optimizasyon metodudur. Parçacık sürü optimizasyonu (PSO) zor ve karmaşık problemleri çözümünde etkili bir algoritma olarak kullanılmaktadır. PSO yapay sinir ağ eğitimi ve fonksiyon minimizasyonu gibi bir çok optimizasyon probleminin çözümünde kolaylıkla kullanılmıştır (Aslantaş v.d. 2006).

PSO çok boyutlu bir arama uzayında sürü parçacıklarının arama davranışına dayalı iteratif bir yöntemdir. Her bir iterasyonda, tüm parçacıkların hızları ve pozisyonları güncellenir. PSO'da her parçacık problem için bir çözüm adayı olarak değerlendirilebilir (Karakuzu 2007).

PSO optimum ya da optimuma yakın çözüm bulmak için önce her biri çözüm adayı olan parçacıklar oluşturur. Bu bireyler belli sınırlar içerisinde rasgele seçilir. Bireylerin bir araya gelmesiyle çözüm için gerçekleştirilen popülasyon oluşturulur.

Parçacık hareket ettiğinde koordinatlarını bir fonksiyona gönderir ve parçacığın uygunluk değeri (optimum çözüme olan uzaklığı) ölçülmüş olur. Parçacığın konum bilgisi (koordinatlarını), hızı (çözüm uzayında ne kadar hızla ilerlediği) ve güncel en iyi uygunluk değeri ile bu değeri elde ettiği koordinatları hafızada tutulmalıdır. Çözüm uzayındaki her boyutta hızının ve yönünün nasıl değişeceği, komşularının en iyi koordinatları ve kendi kişisel en iyi koordinatlarının bir birleşimi olacaktır (Der v.d. 2008).

Algoritma problemin çözümü için aday parçacık popülasyonunun (sürünün) rasgele oluşturulması ile başlar. Sürüdeki her bir parçacığın belirlediği çözüm için aday parametrelerle sistem çözümü yapılır. Elde edilen çözümden problemin yapına göre önceden belirlenen bir ölçüt ile parçacığın uygunluğu sayısal olarak belirlenir. Bu işlem sürüdeki tüm parçacıklar için tekrarlanır ve içlerindeki en iyi uygunluk değerine sahip olan parçacık küresel en iyi olarak etiketlenir. İkinci iterasyon için yeni parçacık değerleri belirlenir. İkinci ve sonraki adımlarda her parçacık için yerel en iyi parçacık, önceki adımlardaki en iyi uygunluk değerini alan aynı indisli parçacık olarak belirlenirken, küresel en iyi parçacık ise o iterasyona kadarki tüm parçacıklar içinden en iyi uygunluk değerini veren parçacık olarak etiketlenir (Karakuzu 2007). Bu öğrenilmiş iki en iyi koordinatın yanı sıra rassal seçilen bir parçacığın koordinatları da algoritmanın güncellenmesine katkı sağlar.

Popülasyonlardaki iki boyutlu davranışlar; çevrelerine adapte olabilme, zengin yiyecek kaynakları bulabilme ve avcılardan kaçabilme gibi ‘bilgi paylaşma’ yaklaşımı gerektirmektedir. Bilgi paylaşımı sayesinde sürüler, yiyecek ararken ya da avcıdan kaçarken, hedefe en yakın olan sürü elemanını takip ederler ve kendi hızlarını ve konumlarını en başarılı elemana göre güncellerler (Der v.d. 2008).

Parçacık sürüsü optimizasyonu (PSO) ilk olarak; 1995 yılında “Particle Swarm Optimization” isimli çalışmayla Uluslarası Sinir Ağı Konferansı’nda (International Neural Network Conferance (IEEE)) duyurulmuştur (J.Kennedy ve Eberhart 1995). Kennedy ve Eberhart (1995) özellikle biyolog Frank Heppner tarafından geliştirilen ve kuş sürülerinin davranışının incelendiği bir model ile ilgilenmişlerdir. Bu modelde

kuşların yiyecek aramaları ve yiyecek bulunduktan sonra kuşların sürü halinde yiyeceğe doğru yönelmelerinin modellenmesi gerçekleştirilmiştir. Bu modelde, yiyeceğe ilk ulaşan kuşun, diğerlerine rehberlik etmesi ve sosyal bir düzlemde bireyler arasındaki sosyal bilgi paylaşımı konu edinilmiştir. Kennedy ve Eberhart (1995) çalışmalarında; algoritmanın şimdiki durumunu almadan önceki denemeleri inceleyerek ortaya atılan çeşitli yöntemleri açıklanmış ve son olarak algoritmanın bu günkü hali verilmiştir.

1998 yılında yayınlanan "A Modified Particle Swarm Optimizer" isimli çalışmada gelişmiş PSO yöntemi ortaya atılmıştır (Shi ve R. Eberhart 1998). Bu yöntemde çözümü yakınsama üzerinde büyük etkisi olan "eylemsizlik ağırlığı" isimli parametre algoritmaya dahil edilerek geliştirilmiştir.

Yine 1998 yılında R. Eberhart ve Yuhui Shi tarafından yayınlanan bir çalışmada, algoritma içerisindeki bazı parametrelerin alması gereken değerlerle ilgili örnek bir sistem üzerinde elde edilen sonuçlar sunularak parametrelerin alması gereken tipik değerler verilmiştir. Uygun değerler seçilmesi halinde algoritmanın başarımının arttığı görülmüştür (Shi ve R. C. Eberhart 1998). Daha sonra 2004 yılında Zhang Li-ping, Yu Huan-jun ve Hu Shang-xu tarafından benzer bir çalışma yayınlanmıştır (Li-Ping v.d. 2005).

Algoritmanın başarımını arttırmak için günümüzde de çeşitli çalışmalar yapılmaktadır. Araştırmacılar bu çalışmalarda genel olarak hibrit (karma) algoritma yapıları üzerinde durmaktadır. Bu yapılarda çeşitli algoritmaların üstün olan yönleri bir araya getirilerek PSO algoritmasının daha iyi sonuçlar vereceği düşünülmektedir. Literatürde bu yönde yapılan çalışmalardan bazıları aşağıda sunulmuştur.

2004 yılında Chia-Feng Juang tarafından yapılan çalışmalar sonucunda Genetik algoritma ile PSO algoritmasının birleşiminden elde edilen, genetik olarak PSO üzerinde duran ve HPSOGA olarak adlandırılan hibrit bir algoritma sunulmuştur (Juang 2004). Bu çalışmasında PSO algoritmasının bulanık-nöral kontrolör eğitimi gerçekleştirilmiştir.

Yine 2004 yılında Uluslararası Güç Sistemleri Konferansında sunulan bir çalışmada önerilen yöntemlerle parçacıklar y, rasgele seçilen bir parçacığın konumu dikkate alınarak bireyler arası bilgi paylaşımının artması sağlanmış ve algoritmanın yerel çözümlere takılması engellenmeye çalışılmış (He v.d. 2004).

Yine 2006 yılında yapılan diğer bir çalışmada ise mutasyon olarak adlandırılan bir işlemle, sürüdeki parçacıklardan rasgele seçilen birinin yeri değiştirilerek bir araya toplanan parçacıklardan birinin topluluktan ayrılması sağlanıp algoritmanın yerel çözümlere takılması engellenmeye çalışılmıştır (Esmin ve Lambert-Torres 2006).

Parçacık sürüsü algoritması günümüzde, fonksiyon optimizasyonu, bulanık sistem kontrolü, yapay sinir ağı eğitimi gibi birçok alanda başarıyla uygulanabilmektedir. Algoritmanın kullanımıyla ilgili literatürden seçilen bazı çalışmalar aşağıda verilmiştir.

2005 yılında yayınlanan bir çalışmada, üç farklı problemin çözümü için, üç farklı yapıdaki yapay sinir ağını PSO ile eğitmişler ve elde ettikleri sonucu, ağ eğitiminde kullanılan klasik geriye yayılım algoritmasıyla yapılan eğitim sonuçlarıyla karşılaştırmışlardır (Zhao v.d. 2005).

2005 yılında C. F. Juang ve Chao-Hsin Hsu tarafından yayınlanan çalışmada, yinelemeli bulanık ağ ile ters sistem yapısında kontroller tasarlanarak ağ eğitiminde simplex metodu ile PSO dan oluşan hibrit bir yapı kullanılmıştır. Bu yapıda kontroller gerçekleştirilmiş ve su sıcaklığı kontrol edilmiştir (Juang ve Hsu 2005).

SMTWT sorununun kesin çözümü için genellikle dal-sınır algoritması ve dinamik programlama kullanılmaktadır. Bir diğer etkili sezgisel yöntem ise ortalama gecikmeyi en aza indirgemek için kullanılan bitişik eşli değişim (API) yöntemidir (Fry v.d. 1989).

Eşit olmayan hazırlık süreli tek makine toplam ağırlıklı gecikme problemi şu şekilde sunulmuştur; $1|r_j|\sum w_i T_i$ burada herhangi bir yaklaşımda alternatiflerin sayısının azaltılması için Akturk ve Özdemir'in $1|r_j|\sum w_i T_i$ kuralı kullanılabilir (Akturk ve

Ozdemir 2001). Çakar, eşit olmayan hazırlık süreli tek makine toplam ağırlıklı gecikme problemleri için nörolojik bir yaklaşımı kullanmıştır (Çakar 2011). Mahnam ve Moslehi yaptıkları çalışmada ise eşit olmayan hazırlık süreli tek makine toplam maksimum gecikme ve erken tamamlanmanın en küçüklenmesi problemi için çözüm önerilerinde bulunmuşlardır (Mahnam ve Moslehi 2009). Bu problemin NP açısından zor olduğu ve dolayısıyla kesin bir yöntem olarak dal-sınır algoritmasını geliştirdiği ispatlanmıştır. Eren çalışmasında, eşit olmayan hazırlık süreli tek makine çizelgeleme probleminde öğrenmenin etkili olacağını düşünmüştür (Eren 2009). Van der Akker ve ark., n adet işin hazırlık zamanları, teslim tarihleri, gecikme cezaları ve eşit işlem süreleri ile tek makine planlama sorunu üzerinde çalışmışlardır (Van den Akker v.d. 2010). Bu çalışmadaki amaç toplam ağırlıklı gecikmenin minimize edilmesidir. Kooli ve Serairi SMTWT (tek makine toplam ağırlıklı gecikme) yi karışık tamsayı programlama yaklaşımını kullanarak eşit olmayan hazırlık zamanı ile çözmüştür (Kooli ve Serairi 2014). Yin ve diğ. de eşit olmayan hazırlık süresine sahip SMTWT nin çözümü için bal arısı optimizasyon yöntemi ile dal-sınır algoritmasını kullanmıştır (Yin v.d. 2012). Wuetal çalışmasında "tavlama benzetimi" yöntemini kullanarak eşit olmayan hazırlık süreli SMTWT yi çözmüştür (Wu v.d. 2011). Matsuo ve ark. çalışmalarında Benzetimli tavlama algoritmasını kullamıştır (Matsuo v.d. 1989). Crauwels ve ark. SMTWT problemi için tabu arama, Genetik Algoritma ve Sinir ağları gibi birkaç sezgisel yöntemin karşılaştırmalı bir çalışmasını sunmuş ve sonuç olarak bu karşılaştırmalarda arasında en iyi sonucu tabu arama vermiştir (Crauwels v.d. 1998). Den Besten ve ark. Ve Merckle ve Middendorf SMTWT problemi için karınca sürü optimizasyonu yöntemini kullanmıştır (Den Besten v.d. 2000; Merkle ve Middendorf 2000). Laguna ve ark. Tek makine çizelgeleme problemlerinin çözümü için tabu arama yöntemi içinde üç yerel arama stratejisinin kullanılması ile ilgili bir çalışma sunmuşlardır (Laguna v.d. 1991). Taşgetiren ve ark. SMTWT sorununu çözmek için PSO kullanmışlardır (Tasgetiren v.d. 2004). Panneerselvam tek makine problemini çözmek için basit bir sezgisel yöntem önermiştir (Panneerselvam 2006). Sen ve ark. SMTWT hakkında hakkında ayrıntılı bir araştırma raporu yayınlamıştır (Sen v.d. 2003). Yang ve ark. PSO nun performansını arttırmak için PSO ve Sinir ağları ile ilgili kombine bir sistem önermiştir (Pan v.d. 2006).

BÖLÜM 2. TEK MAKİNE ÇİZELGELEME ve SIRALAMA

2.1. Tek Makine Çizelgeleme

Tek makine çizelgeleme konusu birçok teorik çalışmaya zemin oluşturmuştur. Çoklu makine sistemlerinin daha iyi dizaynı ve analizi için teorilerin anlaşılması önemli bir hale gelmektedir.

Tek makine çizelgeleme modellerinde genel olarak kabul edilen varsayımlar aşağıdadır.

1. Makine çizelgeleme döneminde sürekli çalışır durumdadır.
2. Makinede işlemler birer birer gerçekleşir.
3. Her bir işin makine üzerindeki proses zamanı tam olarak bilinir ve önceki işlere bağlı değildir.
4. Proses zamanı hem hazırlanma zamanını hem de gerçek makine zamanını içerir.
5. Diğer iş ile ilgili bilgi önceden bilinir. Bu bilgi bitiş tarihi (dj) ve release zamanını (rj) içerebilir.
6. Öncelikli olmayan çizelgelemede, işler kesintisiz işlem görürler. Öncelikli olan çizelgelemede işler operasyon tamamlanmadan makinadan çıkarılabilir (Alharkan 2005).

2.1.1. Çizelgelemenin matematiği

Çizelgeleme parametrelerinde iş j olarak ifade edilir.

$p_j = j\ işinin\ işlem\ süresi$

$S_j = j\ işinin\ başlama\ zamanı$

$W_j = j\ işinin\ bekleme\ zamanı$

$D_j = j\ işinin\ teslim\ tarihi$

$E_j = j\ işinin\ erken\ bitirme\ zamanı$

$r_j = j\ işinin\ hazırlık\ zamanı$

$C_j = j\ işinin\ tamamlanma\ zamanı$

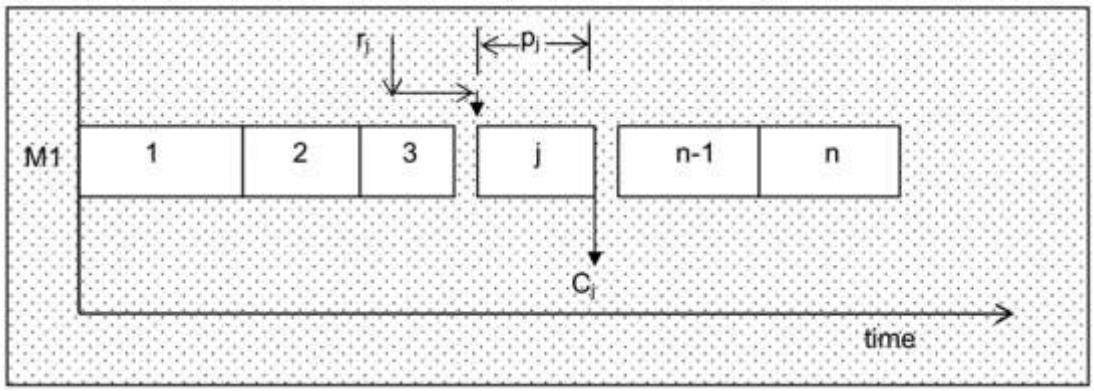
$F_j = j\ işinin\ akış\ süresi$

$L_j = j\ işinin\ sapma\ süresi$

$T_j = j\ işinin\ gecikme\ süresi$

2.1.2. Gant şeması

Gant şeması makine üzerindeki işlerin çizelgelemesini gösteren en popüler yoldur. Gant şemasında yatay düzlemde zaman, dikey düzlemde ise makine gösterilir. *n makine için (Alharkan 2005).



Şekil 2.1. Gant Şeması

Yukarıdaki gant şemasında gösterilen kutucuklar (bar) içerisindeki sayılar iş numaralarını göstermektedir. Proseslerin başlangıç ve bitiş zamanına göre

kutucukların boyu ayarlanır. Kutucuklar (bar) ile temsil edilen işler aynı zamanda makinenin boş kaldığı zamanları da gösterir.

Yukarıdaki gant şemasına göre; j işi için bekleme süresi $W_j = C_j - r_j - p_j$ (Denklem 2.1) , benzer şekilde gecikme (Lateness) (L_j) j işi için $L_j = C_j - d_j$ (Denklem 2.2). İş tardiness T_j pozitif gecikmesidir ve matematiksel gösterimi;

$T_j = L_j, eğer\ L_j > 0\ değilse,$

$T_j = \max(L_j, 0)$

Benzer şekilde iş erken tamamlanma negatif gecikme olmaktadır ve E_j ile gösterilir;

$E_j = L_j, eğer\ L_j < 0\ değilse,$

$E_j = \max(-L_j, 0)$

İş akış süresi (F_j) iki yol (Denklem 2.3, Denklem 2.4)ile gösterilebilir;

$$F_j = C_j - r_j \quad (2.3)$$

$$= W_j + p_j \quad (2.4)$$

$r_j, W_j\ ve\ p_j$ arasındaki ilişki C_j (Denklem 2.5) ile açıklanabilir;

$$C_j = r_j + W_j + p_j \quad (2.5)$$

$$= S_j + p_j$$

Burada $S_j = r_j + W_j$, (Denklem 2.6) eğer $W_j = 0$ ise $S_j = r_j$ olmaktadır. Çizelgelemenin başlangıç zamanı olarak; $S_j = C_j - 1\ eğer\ C_j - 1 > r_j, değilse = r_j$ (Denklem 2.7) olarak alınır.

2.2. Tek Makine Sıralama

Tek makine sıralama problemleri bütün işlerin tamamlanarak bir çizelge olarak düzenlendiği özelleşmiş bir çizelgeleme problemleridir. Hatta en basit tek sıralama problemi tek kaynak veya tek makine olanıdır ve bütün işlem zamanı belirleyicidir. Basit olmasına rağmen tek makine durumu çok önemlidir. Tek makine problemi kolay işlenir modelde çizelgeleme konularının çeşitliliğini anlatmaktadır. Bu durum birçok farklı performans ölçümünün ve çeşitli çözüm teknolojilerinin incelendiği bağlamlar sağlar. Bu bağlamlar çizelgeleme kavramında kapsamlı yaklaşımın gelişimindeki yapı taşlarını oluşturur. Karmaşık sistemin davranışını tamamen anlaya bilmek için onun parçalarını anlamak gerekir ve tek makine problemleri sıklıkla büyük çizelgeleme problemlerinin bir parçası olarak görülmektedir. Bazen tek makine problemlerinden bağımsız olan çözümler olabilir ve bu durum büyük problemlerin çözümünü kapsar. Örnek olarak çok operasyonlu bir işlemde her zaman darboğaz olabilir ve bu darboğazın işleyişi bütün çizelgelemenin özelliklerinin belirlenebildiği tek makine analizi ile anlaşılır.

Tek makinada kısıtlamalara ek olarak, temel problem şu koşullara göre nitelendirilir.

1. n, (zamana 0’ daki) işlem için aynı zamanda kullanılan tek tezgahlı işlerdir.
2. Makinalar bir işin büyük bir kısmını birebir işleye bilmektedir.
3. İşin ayar süresi iş sıralamadan bağımsızdır ve işlem süresini kapsar.
4. İş tanımlayıcılar ilerlemede bilinir ve saptana bilinir.
5. Makinalar (bozulmalar yokken) aralıksız kullanılır.
6. İş beklerken makinalar asla boş durmaz.
7. Bir defa işler başlar ve aralıksız ilerler.

Bu koşullar altında, n tane işin sıralanması ve 1,2,…,n iş göstergesinin sıraları altında birebir uygunluk vardır. Temel tek makine probleminde toplam farklı çözüm sayısı, n tane elemanın farklı sıralama sayısı olan n!' dir. Çizelge tam sayıların sıralaması olarak tanımlandığı zaman tek makine örneğinde öte geliştirilmiş sınıflandırma olan permütasyon sıralaması olarak adlandırılır.

Tek makina problemlerinde işin özelliğinden bahsedilmesi, çizelge kararının sonuçlarından meydana gelen bilgi ve ilerleme ile bilinen bilgi arasındaki farkın ayırt edilmesinde yararlı olacaktır. Çizelgeleme sürecinde "girdi" yi teşkil eden ilerleme ile bilinen bilgi ve bu tip verileri genellikle küçük harf le işaretleyerek ifade edilir. Tek makine örneğinde tanımlanan işlere yardımcı olan 3 temel bilgi aşağıdaki gibidir.

İşlem süresi (p_j) j işine göre talep edilen iş miktarıdır.
Hazırlık zamanı (r_j) j işindeki işlem için hazırlık zamanıdır.
Teslim tarihi (d_j) j işindeki işlem için zamanın tamamlanmasıdır.

Bu üç koşula göre işlem süresi p_j genellikle fabrika kurum zamanını ve direk işlem zamanını içermektedir.

Çizelge kararının sonuçlarından meydana gelen bilgi, çizelge fonksiyonunda "çıktı" olarak ifade edilir ve genellikle bu tip verileri büyük harf ile ifade edilir. Çizelgeleme kararı değerlendirme çizelgelerinde en temel veriler kullanılarak belirlenir.

Tamamlanma zamanı (C_j) işin bitiş zamanıdır.

Değerlendirilen çizelgeler için nicel ölçümler, iş bitiş zamanın genel fonksiyonlarıdır. İki önemli nicelik:

Akış süresi (F_j) (Denklem 2.8) j işi için sistemde harcanan zaman:

$$F_j = C_j - r_j \tag{2.8}$$

Sapma (L_j) (Denklem 2.9) j işinin tamamlanma zamanının son tarihi aşmadığı zaman miktarı,

$$L_j = C_j - d_j \tag{2.9}$$

Bu iki değişken işin iki cinsini yansıtmaktadır. Akış süresi iş için yalnız talebe dair sistem etkisinin ölçülmesidir ve geliş ve gidiş arasındaki işlerin bekleme arlığını ifade etmektedir. Sapma son tarihi verilen çizelgenin uygunluğunun ölçülmesidir ve iş erken tamamlandığı zaman negatif değer almaktadır. Negatif sapma işin istenilenden erken bittiğini, pozitif sapma ise işin istenilenden geç bitiğini ifade etmektedir. Birçok durumda, pozitif sapmanın farklı cezaları vardır. Fakat negatif sapmanın faydası yoktur. Böylelikle bu sadece negatif sapmanın olduğu değişken ile birlikte faydalıdır.

Gecikme (T_j) (Denklem 2.10) j işi teslim tarihinde başarılamazsa veya sıfırsa bu işin geç kalma zamanı,

$$T_j = \max\{0, L_j\} \tag{2.10}$$

Çizelgeleme tek boyutlu performans ölçümleri sonucunda elde edilen bütün işlet hakkındaki bilgiyi kapsayan toplam değişkenlere göre değerlendirilir. Çizelge performans ölçümleri genellikle çizelgenin tamamlama zamanının ayarlandığı fonksiyonlardır. Örnek olarak n tane işin çizelgelenmesi gerekirse toplam performans ölçümleri şu (Denklem 2.11) şekilde tanımlanabilir.

$$F = \sum_{J=1}^{n} F_j \tag{2.11}$$

Toplam akış süresi (Denklem 2.12):

$$T = \sum_{J=1}^{n} T_j \tag{2.12}$$

Maksimum akış süresi (Denklem 2.14):

$$F_{max} = max_{1 \le j \le n}\{F_j\} \quad (2.13)$$

$$U = \sum_{J=1}^{n} \delta(T_j) \quad (2.14)$$

Geciken iş sayısı veya toplam birim ceza (Denklem 2.15):

$\delta(x) = 1\ ise\ x > 0\ ve\ \delta(x) = 0$ (2.15) ise diğer tercihler yapılır.

Maksimum tamamlanma süresi (Denklem 2.16):

$$C_{max} = max_{1 \le j \le n}\{C_j\} \quad (2.16)$$

Temel varsayımlara göre $C_{max} = F_{max} = \sum p_j$ ve bu değişkenler yayılma süresi olarak bilinir. Ancak bir takım varsayımlar altında bu üç performans ölçümü birbiriyle aynı olmayabilir (Baker ve Trietsch 2009).

Bu gösterim biçimiyle F problemi ve bezer şekilde olan T problemi, C_{max} problemi ve benzer problemler için toplam akış süresinin en aza indirilmesinde uygundur. Örnek olarak toplam iş süresi her bir işin akış süresinin basitçe toplamıdır. Bu fonksiyon çeşidinde her bir iş performans ölçümüne direk katkı sağlamaktadır. Çünkü her biri birbirinden ayrı akış süresi toplamının parçalarıdır. Diğer taraftan bazı işler F_{max} problemi için performans ölçümüne sadece dolaylı yoldan katkı sağlayabilmektedirler. Çok fazla akış zamanına ulaşmamak için j işi çizelgelenmeye bilir. Toplam akış zamanı yerine performans ölçümleri olan ortalama akış zamanı aynen alınabilir. Ortalama değer F/n sayısı veya iş sayısı olarak ayılmış toplam değerdir. Benzer şekilde toplam gecikme kabul edilen ortalama gecikme zamanının 1/n' i olarak ölçülmektedir ve U işi gecikme miktarını ölçmektedir.

Bu ölçümlerin her biri işin tamamlanma zamanını ayarlayan fonksiyonlardandır. Bu fonksiyonun yaygın biçimi (Denklem 2.17):

Z= f (C_1 C_2, … , C_n) (2.17)

Üstelik bu değişkenler devamlı ölçümler olarak adlandırılan performans ölçümlerinin en önemli kısmı ile ilgilidir. Z performans ölçümleri devamlı ise;

1. Çizelgelemenin amacı Z' yi en aza indirmektir.
2. Çizelgeleme artışında tamamlanma zamanında en az birinin artması kaydıyla Z arttırıla bilinir.

2.2.1. Akış süresi

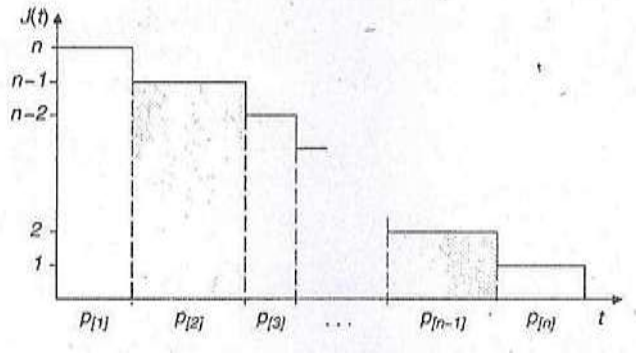
Bazen maliyetler sistemde harcanan zamanı yansıttığından müşterilere yapılan hizmeti kapsayan çizelgeleme kararları ile ilişkilendirilir ve çizelgeleme amacı hızlı (iş verip alma) geri dönüştür. Diğer durumlarda, maliyetler işleme stoklarının davranışları ile belirtildiğinden sistem araştırmalarında yatırımı içerir ve çizelgeleme amacı düşük seviyede sürdürür. Bu iki amaç arasındaki yakın ilişki tek makine modeli ile anlatılabilir.

Sistemdeki iş için harcanan süre akış süresidir ve geri dönüş(iş alıp verme) amacı en düşük toplam akış süresi olarak yorumlanabilir. "düşük stok" seviyesi sistemdeki en düşük ortalama iş sayısı olarak yorumlanabilir. J, t zamanında sistemdeki iş sayısını belirtmektedir ve J(t) fonksiyonunun zaman ortalamasıdır. Tek makine modelinde J(t)' nin davranışını göstermek basittir. 0 anında sistemde -n tane iş vardır ve J(0) = n dir. $F_{[1]} = P_{[1]}$ (Denklem 2.18) zamanında oluşan ilk iş tamamlanana kadar J(t) de değişme olmaz. Ayrıca J(t) (n-1)'e düşer ve $F_{[2]} = P_{[1]} + P_{[2]}$ (Denklem 2.19) zamanında oluşan 2.iş tamamlanana kadar sabit kalır. Bu şekilde devam eder ve Şekil 2.1' de görüldüğü gibi J(t) çizelgenin süresi boyunca azalan adım fonksiyonu olduğu görülebilir. Üstelik çizelgenin süresi, iş süresince sıralamadan bağımsız $F_{max} = P_1 + P_2 + \ldots + P_n$'e (Denklem 2.20) eşittir [0,F_{max}] süresi için, toplam (Denklem 2.21) hesaba katılmalıdır.

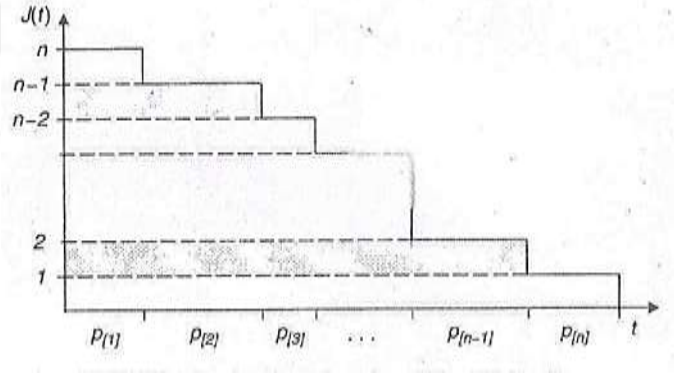
A= $np_{[1]} + (n-1)p_{[2]} + \ldots + 2p_{[n-1]} + p_{[n]}$ (2.21)

Bu toplam, J(t) fonksiyonunun altında kalan alandır. Şekil 2.1'de dikey hatta toplamı ifade eder. Bu nedenle J=A/F_{max}'tır (Denklem 2.22)).

F=$F_{[1]} + F_{[2]} + \ldots + F_{[n]}$ (2.22)



Şekil 2.2. J(t) fonksiyonu



Şekil 2.3. J(t) fonksiyonunun alternatifi.

Bu toplam, Şekil 2.2.'de gösterilen yatay hattın toplamı olarak ifade edilen A'ya eşittir. Böylece F=A (Denklem 2.23)olur. Bu iki ilişki yeniden düzenlenir ve birleştirilirse, cebirsel sonuç:

A = F = J.F_{max} (2.23)

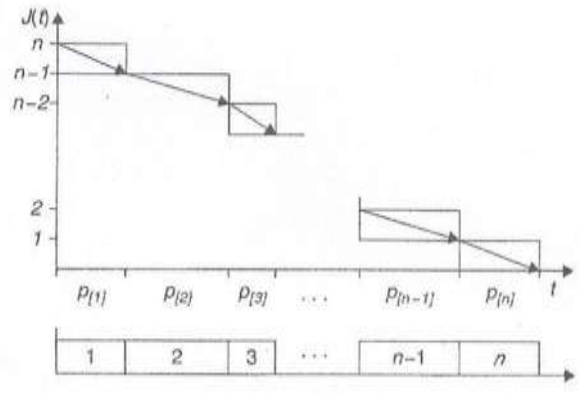
F_{max} sabit olduğundan dolayı, J F ile direkt doğru orantılıdır. Sonuç olarak F'yi minimize eden (toplam akış zamanı) iş sırası aynı anda J' yi de minimize eder.

Üstünlük noktası minimize stok seviyesinin veya optimize müşteri servisinden birinin olup olmadığıdır, bu problemin benzeri; F minimize edilmesiyle oluşan sıralamanın bulunmasıdır.

2.2.2. Toplam akış süresinin en aza indirilmesi

F, akış süresi en aza indirilmesi problemlerinde ve J(t) grafiklerinde göz önünde bulundurulur. (Eşdeğer) Denklik problemi, J(t) fonksiyonu altında kalan alanın en aza indirilmesi ile mümkündür. Sıralama seçimi J grafiğinde (F_{max},0) noktasından (0,n) noktasına çizilen yol (doğru) olarak yorumlanabilir. Bu yol -1/PJ eğimini veren n vektörlerinden meydana gelmektedir. Sıralama için J(t) grafiği Gantt diyagramı ile birlikte Şekil 2.4.'de gösterilmiştir.

Açıkçası bu alan sol tarafa yerleştirilmiş en dik eğim, ayrıca sonraki en dik eğim ve benzerlerine göre en aza indirilebilir. Bu durum azalmayan siparişlerde işlem süresinin sıralanması anlamına gelmektedir.



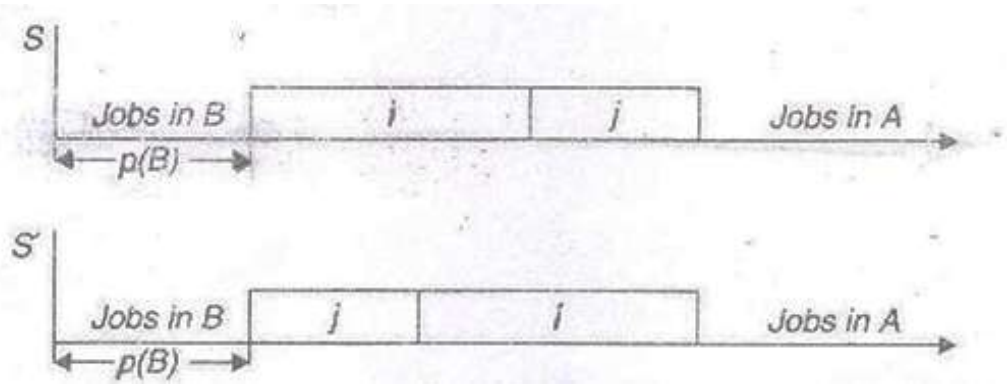
Şekil 2.4. Çizelgeye uygun J(t) fonksiyonu ve Gantt Diyagramı

İşlem süresinde azalmayan siparişli işlerin sıralanması belirli sebeplerden dolayı en kısa işlem süreli (SPT) sıralama olarak bilinir. Fakat çeşitli isimlerinin olduğu da bilinmektedir. SPT, toplam akış süresinin en kısa süreli işlem sıralamasına göre minimize edilmesidir ($P_{[1]} \leq P_{[2]} \leq \ldots \leq P_{[n]}$).

İşlem süresinde azalmayan siparişli işlerin sıralanması belirli sebeplerden dolayı en kısa işlem süreli (SPT) sıralama olarak bilinir. Fakat çeşitli isimlerinin olduğu da bilinmektedir. Örneğin en kısa faaliyet süresi ve en kısa mutlak faaliyet aşağıda SPT' nin optimizasyonu formülize edilmiştir ve bu önerme komşu ikilileri yer değiştirme metodu olarak adlandırılan yöntemi anlatmaktadır.

Toplam akış süresinin en kısa süreli işlem sıralamasına göre minimize edilmesidir. $(P_{[1]} \leq P_{[2]} \leq \ldots \leq P_{[n]})$

SPT sıralaması olmayan S sıralamasının hesaba katılması yani S' de i ve i.'yi takip eden J bitişik işleri bulunmaktadır, öyle ki $P_i > P_j' dir.$ Yeni sıralama S', S ile benzer zamanda tamamlanır ve sıralamada i ve j işlerinin yerleri değişmektedir. Bu durum Şekil 2.4'te gösterilmektedir. Burada B, her iki çizelgede i ve j işlerinin öncesindeki iş sayını ifade etmektedir. A, her iki çizelgede i ve j işlerinin sonrasında ki iş sayısını ifade etmektedir. k işi, A grubu üyesi olduğu zaman k ∈A işaretlenmesi (gösterimi/formülü) kullanılır. Ek olarak, $p_{(B)}$ B grubundaki işler için toplam işlem süresini ifade etmektedir. Yani S' de i işi ve S' de j işi $p_{(B)}$ zaman noktasından sonra başlar.



Şekil 2.5. Komşu işlerin ikili değişimi

Ayrıca S çizelgesi altındaki k işinin akış zamanını ifade eden $F_{k(s)}$ gösterimini geçici olarak kabul edebilir.

$\sum_{S=1}^{n} Fk$, olan ilk gösterim (Denklem 2.24) S' S' den küçüktür.

$$\sum_{j=1}^{n} F_k\,(S) = \sum_{k\in B} F_k(S) + F_i(S) + F_j(S) + \sum_{k\in A} F_k(S)$$

$$= \sum_{k\in B} F_k(S) + (p(B) + p_i) + (p(B) + p_i + p_j) + \sum_{k\in A} F_k(S)$$

$$\sum_{j=1}^{n} F_k\,(S') = \sum_{k\in B} F_k(S') + F_i(S') + F_j(S') + \sum_{k\in A} F_k(S')$$

$$= \sum_{k\in B} F_k(S') + (p(B) + p_i) + (p(B) + p_i + p_j) + \sum_{k\in A} F_k(S') \qquad (2.24)$$

Kurama göre (Denklem 2.25);

$$\sum_{k\in B} F_k(S) + \sum_{k\in A} F_k(S) = \sum_{k\in B} F_k(S') + \sum_{k\in A} F_k(S') \qquad (2.25)$$

Dolayısıyla (Denklem 2.26);

$$= \sum_{k\in 1} F_k(S) - \sum_{k\in 1} F_k(S') = p_i - p_j > 0 \qquad (2.26)$$

i ve j işlerinin değiştirilmesi F' in değerini azaltır. Bu nedenle, SPT sıralaması yapılmayan bazı sıralamalar, komşu işlerin değişimiyle F ile ilgili iyileştirilebilir. Buradan şu sonuç çıkar ki, SPT sıralamasının kendisi en uygun olmalıdır.

Bu yargının nedeni, çelişkiden önce önermedir. İlk olarak SPT sıralaması olmayan bazı varsayımlar "en uygun (optimal)" olarak kabul edilir. Ayrıca, kesin ilerlemenin bu "optimal (en uygun)" sıralamada yapılabilinen komşu işleri, ikili değişimler ile gösteririz. Bu nedenle en uygun olan SPT sıralaması olmadığı için mümkün olmadığı sonucuna varılır.

1. Bazı SPT sıralaması olmayan ile başlanır.
2. i' yi izleyen j ile beraber, komşu i ve j işleri bulunur. Öyle ki $P_i > P_j{}'dir$
3. Sıralamada i ve j işleri yer değiştirir, böylece performans ölçümleri iyileştirilir.

4. Sonuçta SPT sıralaması olana kadar iyileştirilen performans ölçümlerinin her bir zamanı için adım 2'ye geri dönülüp tekrarlanır.

Her iki önerinin hükmü $P_i = P_j$ nedeni ile iş çiftinin bulunmasına bakılarak etkilenmez. Üstelik komşu çiftlerin yer değiştirmesi metodu diğer (daha sonra gösterilecek) durumlarda da faydalıdır.

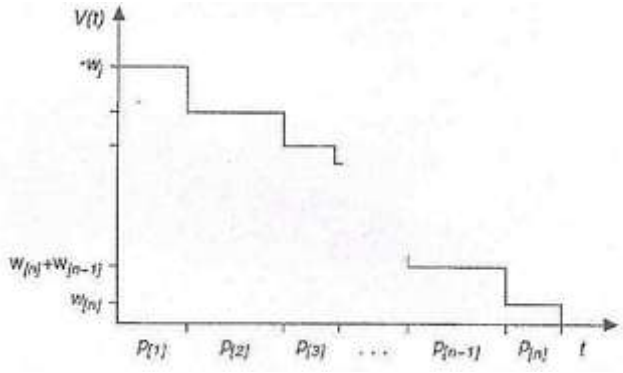
2.2.3. Toplam ağırlıklı akış süresinin en aza indirilmesi

F probleminin ortalama farkından işler eşit döneme sahip değildir. İşleri ayırt etmenin tek yolu her bir iş ve performans ölçümü ağırlıklarının birleşmesiyle oluşan w_j değerinin veya ağırlığının atanmasıdır. Toplam akış zamanının ağırlık biçimi "toplam ağırlıklı akış zamanı" (Denklem 2.27) aşağıdaki gibi tanımlanır.

$$F_w = \sum_{j=1}^{n} w_j F_j \tag{2.27}$$

Burada birim gecikme fiyatı gibi ağırlıkları düşünebilir. Ağırlıklar sayesinde işlemdeki stok değerinin elde tutma maliyetine oranı tanımlanmıştır. J işinin tamamlanması beklenirken bu iş, işlemdeki stoğun toplam değeri w_j' ye katkıda bulunur ve t zamanında sistemde bulunan toplam stok değerini V(t) fonksiyonu ile tanımlanabilir. V(t) fonksiyonu basamak fonksiyonudur. Fakat J(t)' ye benzemez. Bu basamak fonksiyonu bir adımından daha ziyade w_j adımlarında azalır. Şekil 2.5' te V(t) gösterilmektedir. Eğer V işlem aralığı boyunca V(t)' nin ortalama zamanını gösterirse, V(t) grafiğinin altında kalan alan için elde edilmiş iki ifadeyi tekrarlayabiliriz. Özet olarak şekilde düşey şeritler Şekil 2.1' e benzer ve aşağıdaki (Denklem 2.28) gibi bulunur.

$$A = \sum_{j=1}^{n} P_{[1]} \sum_{i=1}^{n} w_i = VF_{max} \tag{2.28}$$



Şekil 2.6. V(t) Fonksiyonu

Özet olarak şekildeki yatay (Denklem 2.29) şeritler Şekil 2.2' ye benzer ve yukarıdaki gibi bulunur.

$$A = \sum_{j=1}^{n} w_j F_j = F_w \tag{2.29}$$

Eğer A için iki ifade eşitlenirse (Denklem 2.30) genel olarak akış zamanı-stok arasındaki ilişkiyi buluruz.

$$F_w = V.F_{max} \tag{2.30}$$

F_{max} sabit olarak incelendiğinde V' yi F_w ile direk orantılı olarak ve diğerini küçülten bir sıralama olarak hesaplanır.

Bununla beraber toplam akış zamanının en aza indirilmesi için en iyi kural en küçük birinci sıralamadır. Toplam ağırlıklı akış zamanı için en iyi kuralın da ise SPT biçiminin ağırlıklı olmasını beklenir. Önceki gibi en iyi kural grafik modelinden çıkartılabilir. Bu durumda V(t) grafiğinde ($\boldsymbol{F_{max}}$,0) noktası ile (0, $\sum_{j=1}^{n} \boldsymbol{w_i}$) noktasını birleştiren yolu aranır. Bu sefer yolun bir araya getirilmesi ile oluşan vektörler $\boldsymbol{w_j/p_j}$ eğimine sahiptir ve V(t) grafiğinin altındaki alanı en aza indirir. Endik eğimi ilk yerine konur ve en iyi kural en küçük ağırlıklı işlem süresi (WSPT) sıralamasıdır.

Toplam ağırlıklı akış zamanının WSPT sıralaması ile en aza indirilmesidir. $(\boldsymbol{P_{[1]}/w_{[1]} \leq P_{[2]}/w_{[2]} \leq \ldots \leq P_{[n]}/w_{[n]}})$

Son olarak SPT ve WSPT genellikle farklı sıralamalar olarak ifade edilir. Dolayısıyla iş kümesi eşit olmayan ağırlıklar içerir. WSPT $\boldsymbol{F_w}$ve V' yi en aza indirir. Fakat sistemde veya toplam akış zamanında işlerin ortalama sayısına gerek yoktur (Pinedo 2005; Alharkan 2005; Baker ve Trietsch 2009).

BÖLÜM 3. TEK MAKİNE PROBLEMLERİ İÇİN OPTİMİZASYON METODLARI

3.1. Bitişik İkili Değişim Metotları

Bir bitişik ikili değişim metotlarının kesin sıralama kurallarının en iyi olarak kullanılmaktadır. Bitişik ikili değişim metodunun doğruluğu aşağıdaki gibi belirtilebilir: Bir sıralama daha düşük performansa izin veren bütün bitişik ikili değişimler için aranır; bu optimal bir sıra olacaktır. Bunu tanımlamak önemlidir, ancak bu yaklaşım için sınırlamalar vardır.

Bitişik ikili değişim metodu sadece sıralama kurallarının sınırlı bir sınıfı için optimal olarak kanıtlamak için yeterlidir. Bireysel işler hakkında bir sıralama oluşturmada bilgiyi kullanan sıralama kuralları için optimal iş sırasının geçişliliği çok önemli bir özellik içerir. T'yi ölçme durumunda, ancak ve ancak optimal sıralama kuralının geçişli olmadığı sonucuna varılabilir.

Bu incelemeler, yeni sıralama problemi çözmede bitişik ikili değişim metotlarını kullanmanın basit bir yolunu gösterir. İlk olarak bir değişimi analiz edilir ve iki işin nasıl sıralanabileceğini belirten bir şart türetilir. Eğer bu şart geçişliye dönüşürse, sıralama gerçekten optimal olacaktır. diğer bir deyişle bir optimum bulmak için daha karmaşık bir yaklaşıma ihtiyaç olacaktır.

3.2. Dinamik Programlama Yaklaşımı

Performansın düzenli bir ölçümü olan Z, işin tamamlanma zamanının bir fonksiyonudur ve fonksiyonu ek olduğunda aşağıdaki (Denklem 3.1) gibi yazılabilir.

$$Z = \sum_{j=1}^{n} g_j(C_j) \tag{3.1}$$

Örneğin, eğer Z toplam gecikme ise, o zaman (Denklem 3.2),

$$g_j(C_j) = \max\{0, C_j - d_j\} \tag{3.2}$$

Başka örnekteki gibi eğer Z, son işlerin ağırlıklı sayısı (Denklem 3.3) ise, o zaman

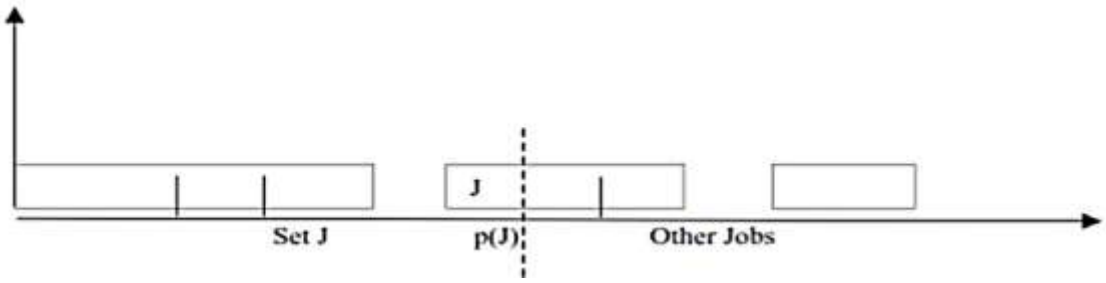
$$g_j(C_j) = w_j\delta(\max\{0, C_j - d_j\}) \tag{3.3}$$

Eğer Z ek bir forma sahipse, bu örneklerdeki gibi, dinamik programlama yaklaşımı ile optimal bir sıralama bulunabilir. Dinamik programlama, ardışık kararlar vermek için genel bir optimizasyon tekniğidir. Burada, örneğin, biz hangi işin ilk, hangi işin ikinci v.b. olacağına karar vermek zorundayız. Kararların her birisinin alt kümesini içeren alt kümelere bölümlenmiş olabilen problemler için dinamik programlama uygulamaları, en iyilik prensibi yoluyla aşağıdaki gibi açıklanır: Varsayalım ilk k kararlarını belirledik (optimal yada değil), sonra kalan (n-k) kararları, onları içeren sadece alt problemler dikkate alınarak optimize edilebilir. Örneğin, varsayalım San Francisco'dan New York'a en kısa yürüyüş yolunu bulmak istiyoruz. Eğer Chicago'ya doğru giden bir yol düşünürsek o zaman oraya nasıl gidersek gidelim, eğer düşündüğümüz yol optimal mesafeye ulaşmak ise Chicago'dan New York'a en kısa yolu takip etmek zorunda olacağız. En iyilik prensibi sıralamada sağlanabilir. (diğer bir deyişle bir sıralama problemi uygun olarak bölümlenmiş olabilir).

Sıralama problemimizde dinamik programlama uygulamak için, J işleri birkaç alt kümeyi gösterir ve p(J), oluşturulan j'de işlerin süreci için gereken toplam zamanı gösterir. kolaylık için, kaldırılan j elemanı ile J dizgisini göstermek için (J-j)'yi kullanılır.

Varsayalım ki bir sıralama diğer bütün işlerden önce J dizgisindeki işler içinde inşa edilmiştir. G(J) = J dizgisindeki işlerden oluşan alt problemler için minimum maliyettir.

Sonra, varsayalım ki, j işi bu alt problemde son konuma atanırsa Şekil 3.1'de gösterildiği gibi p(J) zamanında tamamlanır.



Şekil 3.1. Dinamik programlamada bir sıralama formu

Son gelen j işi göz önüne alındığında, G(J)'nin değeri iki terimin toplamıdır, bu iki terim j işi tarafından yapılan maliyet ve kalan işler tarafından yapılan minimum maliyettir. G(J-j) olarak yazılabilen bu son terim, sadece (J-j) dizgisindeki işleri içeren alt problemin çözümü ile elde edilmiş optimal değerdir. Eğer J dizgisindeki son gelen olağan bütün j işlerini kıyaslarsak ve en iyisini seçersek J dizgisi için minimum maliyet bulunabilir. Formülde (Denklem 3.4),

$$G(J) = \min_{j \in J}\{ g_j[p(j)] + G(J-j)\} \tag{3.4}$$

$$G(\emptyset) = 0 \; ve \; \emptyset \; bo\text{ş} \; k\ddot{u}meyi \; g\ddot{o}sterir \tag{3.5}$$

Sonuç olarak, X, bütün işlerin dizgisini gösterir. Çünkü maliyet fonksiyonu G işlerin alt kümeleri üzerinde tanımlıdır, G(X) (Denklem 3.6) olarak yazılabilen minimum toplam maliyet,

$$G(X) = \min_{j \in X}\{g_j[p(X)] + G(X-j)\} \tag{3.6}$$

Her aşamada G(J) fonksiyonu, J dizgisi çizelgenin başında meydana geldiğinde ve alt kümeler olarak sıralandığında J dizgisindeki işler tarafından katkıda bulunulan toplam

maliyeti ölçer. Öz yineleme ilişkisi (Denklem 3.4), k büyüklüğünün herhangi özel alt kümesi için G değerini hesaplamak için, ilk (k-1) büyüklüğünün alt kümleri için G'nin değerini bilmek zorunda olduğumuzu gösterir. Bu yüzden işler (Denklem 3.6)'dan, sıfır büyüklüğünün alt kümeleri için G değeri ile başlar, o zaman, (Denklem 3.4) kullanılarak, 1 büyüklüğünün bütün alt kümeleri için G değerini, sonra 2 büyüklüğünün bütün alt kümeleri için G değerini vb. hesaplayabiliriz. Bu şekilde, işlem eninde sonunda son çizelgelenmiş olan işi belirlemek için (Denklem 3.6) kullanılarak giderek büyüyen J dizgilerini dikkate alır. Z'nin optimal değeri G(X)'tir. G(X)'i bulduktan sonra (3.4)'teki en küçüğün her aşamada meydana geldiğini göz önünde tutarsak eğer optimal sıra yeniden bulunabilir.

Özet olarak dinamik programlamanın bilgisayar uygulamaları iki etkili beceri geliştirir, alt küme etiketlemek için bir plan ve alt kümeleri üretmek için bir algoritma. Etiketleme planı öncelikli tahlil edilmiş alt kümeler için değeri etkili bir eğilim tedarik eder ki bununla birlikte uygun bir sırada bütün alt kümeler algoritmalar üreterek gerçekleşir.

3.3. Dal-Sınır Yaklaşımı

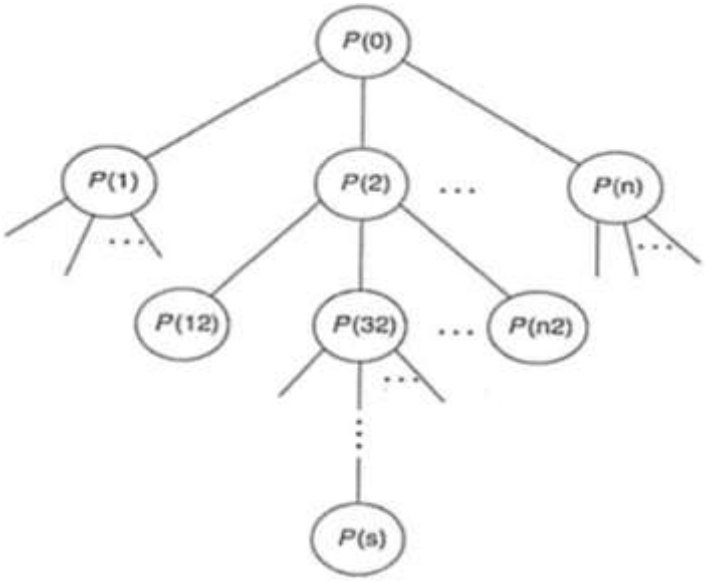
Dal-sınır olarak bilinen genel amaçlı strateji, bir çok kombinasyonel problemin çözümü için kullanışlı bir metottur. Adından da anlaşılacağı gibi yaklaşım iki ana prosedürü içerir. *Dallandırma*, büyük bir problemi iki ya da daha fazla alt problem olarak bölme işlemidir. *Sınırlandırma* ise verilen alt problemin optimal çözümünde daha düşük bir sınır hesaplama işlemidir.

Dallandırma prosedürü aşağıdaki yeni problem setleriyle orijinal problemi değiştirir.
Orjinalin karşılıklı ayrıcalık ve detaylı alt problemleri
Orijinalin kısmen çözümlenen parçaları ve
Orijinalden daha küçük problemler.

Ayrıca, alt problemler benzer şekilde kendi içlerinde bölünebilirler. Dallandırma prosedürüne örnek olarak, P(0), n iş içeren tek makineli sıralama problemini göstersin.

P(0) problemi sırada son pozisyonu atayarak n tane alt probleme bölünebilir P(1), P(2), … P(n). Böylece P(1) aynı problem olur fakat iş 1 son duruma sabitlenir. P(2)de aynı şekilde fakat iş 2 son duruma sabitlenir ve benzer şekilde gider. Açıkça bu altproblemler P(0)'dan daha küçüktür çünkü sadece (n-1) pozisyon atamadan kalmıştır ve belliki heri bir P(i), P(0)'ın kısmen çözümlenmiş parçalarıdır. Ek olarak alt problem seti olan P(i), her bir P(i)'nin çözümlenmesi anlamında P(0)'ın karşılıklı ayrıcalık ve detaylı parçalarıdır. n çözüm arasında en iyi çözümün P(0)'ın çözümü olarak gösterilir. bu nedenle P(i), (a),(b) ve (c) koşullarını sağlar.

Daha sonra her bir alt problem bölünebilir (şekil 3.4). örneğin P(2) P(12),P(32),…P(n2) içinde bölünebilir. P(12)'de 1 ve 2 ve P(32) 3 ile 2 de sırada son iki pozisyonu işgal eder. Bu nedenle birinci seviye bölünme P(i)'nin P(0)aynı ilişkiyi taşıması gibi ikinci seviye bölünme P(i2), P(2) ile aynı ilişkiyi taşımaktadır. Bu nedenle herbir seviyedeki bölünmeler (a),(b) ve (c) koşullarını sağlar. k seviyesinde her bir alt problem k sabit pozisyonu içerir ve (k+1)seviyesindeki parçaların (n-k) alt problemi içinde daha fazla bölünebilir. Eğer bu dallandırma prosedürü tamamen gerçekleştirilirse orijinal problemin her bir farklı uygun çözümleriyle ilgili n.seviyede n! alt problem oluşur. Başka bir deyişle dallanma ağacının detaylı araştırması tüm sıranın sayımını sağlamayla eşdeğerdir. Dallanma prosesi fonksiyonu bu sayımı kısıtlamak için ortalamalar sağlar.



Şekil 3.2. Tek tezgahlı problemler için bir dallanma şeması

Dallanma prosedürü, dallanma prosesindeki geliştirilen her bir alt probleme çözümdeki daha düşük dalları hesaplar. Bazı ara aşamalarda Z'nin performans ölçüsü ile ilişkili olan tam çözümün elde edildiğini varsayalım. Aynı zamanda bu alt problemin $b > Z$ düşük sınırıyla ilişkili olan dallanma prosesi içinde karşılaştırıldığı varsayalım. Daha sonra optimum için aramada daha fazla alt problem dikkate almaya gerek kalmaz. Yani alt problemin nasıl çözüldüğüyle ilgilenilmez, çıkan çözüm Z den daha iyi bir değere sahip olmaz. Böyle bir alt problem bulunduğunda dalı derinliğini ölçmek söylenir. Derinliği ölçülen dallardan daha fazla dallanma olmadan numaralandırma prosesi kısalır çünkü derinlerdeki alt problemlere ait mümkün çözümler açıkça düzenlenme olması yerine kesinlikle değerlendirilir.

Dalların derine inmesini sağlayan tam çözüm deneme çözüm olarak adlandırılır. Bu sezgisel prosedürü uygulayarak başlangıçta da elde edilebilir.(örneğin sınırlı hesaplama gücü ile iyi sonuçlar elde etme yeteneğine sahip bir alt optimal metot) veya ağaç araması sırasında belki olabildiğince hızla direkt en alt dalına ulaşmasıyla elde edilebilir (Alharkan 2005; Baker ve Trietsch 2009; Pinedo 2005).

BÖLÜM 4. TEK MAKİNE PROBLEMİ İÇİN SEZGİSEL METODLAR

Bazı amaç fonksiyonları için, örneğin toplam akış zamanı, optimal çözümün işleri sınıflandırmak kadar basit bir prosedür ile elde edilebilir. Diğer amaç fonksiyonları için, örneğin toplam ağırlıklı gecikme, herhangi basit bir çözüm prosedürü uygun değildir ve kombinatoryal optimizasyonun daha fazla genel tekniklerine başvurulması gerekmektedir.

Kombinatoryal prosedürler kullanarak problemleri çözmek için gereken hesapsal çaba problemlerin boyutu arttıkça hızla büyür. Örneğin dinamik programlama algoritması için bir bilgisayar uygulamasının saniyede 1.000.000 alt grup oluşturmamıza ve değerlendirmemize izin verdiğini varsayalım. O zaman 25 iş probleminin çözümü kabaca yarım dakika bilgisayar zamanını alırdı; ancak 35 iş probleminin çözümü 9 saat, 45 iş probleminin çözümü ise bir yıldan uzun sürerdi. Eğer 45 iş problemine hızlı bir cevap almak istiyorsak dinamik programlama yaklaşımı pek uygun olmayacaktır. Dal-sınır algoritmaları uygulaması durumunda, daha iyi bir performans garanti edemeyiz çünkü hesapsal çabasını tam olarak tahmin etmek imkansızdır, her spesifik problemin parametrelerine bağlıdır.

Her ne kadar tipik pratik problemler gibi herhangi bir problem boyutunu belirlemek zor olsa da, 30-50 iş içeren problemleri çözme becerisinin çoğu pratik ihtiyaç için yeterli olacağına inanılmaktadır. Fakat tek makine modeli ile öncelik kısıtları, birden fazla makine ya da iş başına birden çok işlem gibi özelliklerle ilgili daha karmaşık problemlerin bir bileşeni olarak da karşılaşabilir. 30 iş tek makine problemini çözme yeteneği, daha karmaşık problemlerde 30 için optimum biçimde çözebileceğimizi ifade etmemektedir. Çoklu makine modellerinde, tek makine alt modelleri belki de 2^n kat kadar tekrar tekrar çözülmelidir. Bu nedenle, bir optimizasyon tekniğinin kullanımı düşünüldüğünde hesapsal taleplerini değerlendirmek önemlidir. Bu talepler önemli

olduğunda, sınırlı hesaplama çabasıyla iyi çözümler bulma yeteneğine sahip en uygun yöntemleri veya sezgisel yöntemleri düşünmek daha yararlı olacaktır. Dinamik programlama ya da dal-sınır gibi metodolojilerin aksine, bu teknikler optimumun bulunabileceğini garanti etmez ancak nispeten basit ve etkilidir.

Bu bölümde, çizelgeleme problemlerini çözmede kullanışlı olduklarını kanıtlayan bazı genel sezgisel prosedürler anlatılmaktadır. Bunların deterministik tek makine problemlerine uygulanmasını açıklanmakta, ancak esasen örnekleme: aynı prosedürler stokastik tek makine problemlerine ve diğer çeşitli çizelgeleme problemlerine uyarlanabilir.

Sezgisel prosedürler, optimal çizelgeleri güvenilir bir şekilde üretmediğinden, optimale en yakın nasıl olacağını sormak mantıklıdır. Deneysel bir ortamda, bir araştırmacı, sezgisel bir prosedür kullanarak birkaç problemi çözerek ve optimal çözümlerin üretildiği frekansı veya optimallikten ortalama sapmayı hesaplamaya çalışarak bu soruyu yanıtlamaya çalışabilir. Bu tür performans ölçütleri, belirli bir prosedürün içindeki güvenilirliği kavramamızı sağlar. Bu bölümde, bu yaklaşımı kullanarak sezgisel prosedürlerin nasıl değerlendirildiği gösterilmektedir.

4.1. Öncelik ve Oluşturma Yöntemleri

En basit çözüm yöntemlerinden bazıları yalnızca işlerin sıralanmasını gerektirir. Örneğin, F probleminde, işleri SPT'ye göre sıralamak, optimal sıralamayı üretir. Aslında, makine boşta kaldığında, bekleyen tüm işleri sıralamak gerçekten gerekli değildir en kısa süreli bekleyen işi belirleyip bunu bir sonraki iş için sıralanmalıdır. Daha spesifik olarak, sıralama terimini, iki işin göreceli sıralamasının zamanla değişmediği özelliğiyle bir sıralama şemasını tanımlamak için kullanıyoruz. Başka bir deyişle sıralama, statik öncelikleri içerir. Buna ek olarak, sıralanmış bir kümeye yeni bir iş eklenirse, orijinal işlerin göreceli sıralaması değişmez. Yeni işin bir sonraki iş planında yapılması gerekip gerekmediğini belirlemek için, tüm işler dizisini yeniden sıralanmak zorunda değildir, sadece yeni iş ile mevcut işi yüksek önceliklerine göre karşılaştırması gerekmektedir. Daha genel olarak, öncelik terimini, makine boş

kaldığında her seferinde sonraki işi seçmek için bir karar kuralını kullanan bir prosedürü tanımlamak için kullanılır. Öncelik, dinamik ve statik sıralama kurallarını içerir. Dinamik bir versiyonu örneklemek için T problemini düşünelim. Basit ama etkili bir sezgisel kuralı, işleri MDD kriterine göre sıralamıştır. t zamanında j (Denklem 4.1) işinin revize edilmiş teslim zamanı aşağıdaki gibi ifade edilir;

$$d_j(t) = \max\{d_j, t + p_j\} \tag{4.1}$$

Aynı zamanda şunu gördük ki, i ve j işleri t zamanında başlayacak işlere adaylarsa, daha önce revize edilmiş teslim zamanına sahip işler önce gelmelidir. Sonrasın da eğer bu kuralı öncelik prosedürü olarak kullanacaksak, optimal çözümü elde edilemeyebilir.

Başka bir dinamik öncelik örneği için, daha karmaşık TW problemine dönersek, sezgisel bir yaklaşım MDD'yi WMDD (weighted modified due date) kuralına genelleştirir, azalmayan siparişin miktarı $\max\{d_j - t, p_j\}/w_j$ olarak tanımlanır. Bu kurala göre öncelik, işleri sıralama ve bir sonraki iş olarak en küçük ağırlıklı revize edilmiş termin zamanı olan işi seçmektir. Sıralama dinamiktir çünkü öncelik kriterleri t ye bağlıdır. Sezgisel yargılamanın yararlı bir yolu, davranışlarını özel durumlarda izlemek ve iyi karar kurallarına indirgemelerini kontrol etmektir. Örneğin tüm ağırlıklar eşitse, WMDD MDD kuralına indirgenir. Aynı zamanda, tüm teslim zamanları sıfırsa, WMDD WSPT ye indirgenir, tüm işler geç ise optimaldir. Bununla birlikte, MDD'nin aksine, WMDD'nin bile iki işi en iyi şekilde sıralaması garanti edilmez.

Bir diğer yaygın olarak kullanılan kuruluş prosedürü, aşağıdaki gibi çalışan araya girme prosedürüdür. Sadece 1 ve 2 numaralı işlerden oluşan alt problemi düşünelim. Sıralarını optimize edin (1-2 ve 2-1 seçeneklerini karşılaştırarak). Sonra, ilk iki işin göreceli sırasını sabit tutarak, 3. işi eklemek için en iyi yeri bulun. Bir başka deyişle, eğer iki iş sıralama alternatifi içinde 1-2 daha iyiyse, üçlü sıralama alternatiflerini 3-1-2, 1-3-2 ve 1-2-3 şeklinde düşünün. Eğer 2-1 sıralaması iki iş sıralamalarının içinde daha iyiyse, üçlü sıralama alternatiflerini 3-2-1, 2-3-1 ve 2-1-3 şeklinde düşünün.

Aşama k' da, ilk k işlerinden oluşan k-iş alt problemine bir çözüm getirilir. Ardından $(k+1)$ aşamasında, ilk k işlerin göreli sırasını bu sırayla sabit tutuyor ve $(k+1)$ olası konumların her birine $(k+1)$ yerleştirmeyi düşünüyoruz. Bu $(k+1)$ seçeneklerinin en iyisini bir sonraki aşamada değerlendirmek üzere seçiyoruz ve en iyi n-iş alternatifini üretildiğinde durulur.

4.2. Rasgele Örnekleme

Deterministik çizelgeleme problemleriyle bağlantılı rasgele örnekleme yöntemlerinden konuşmak şaşırtıcı görünebilir. Bununla birlikte, rasgele örnekleme diğer kombinasyonal ortamlarda doğrudan uygulanmıştır ve birçok çizelgeleme problemi için uygulanabilir bir çözüm stratejisi sağlayabilir (Baker ve Trietsch 2009).

Bir örnekleme prosedürünün esas niteliğini tarif etmek oldukça kolaydır. Bazı rasgele cihazlar kullanarak N sıralamalarını oluşturulur, değerlendirilir ve numunedeki en iyi diziyi belirlenir. Rasgele örneklemeyi, özel bir sezgisel prosedür ve optimizasyon prosedürü arasındaki süreklilikte uzanan bir çözüm yöntemi olarak görebiliriz. Rasgele bir örnekleme prosedürü, sıralamadaki bazı ara sayıları oluşturur ve en iyi olanı seçer. Örnekleme planının tasarımı iki taktik sorunu çözmelidir:

1) Örnekleme yapmak için belirli bir cihazı nasıl belirleyebiliriz?
2) Numunedeki en iyi sıralama hakkında sonuçlar çıkarabilir miyiz?

Örnekleme teknikleri üzerine yapılan literatürün birçoğu, bu soruların yanıtlarını biraz daha ayrıntılı olarak keşfetmeye çalışmaktadır; bu soruları daha yakından inceleyecek olursak, Örnekte bulunan en iyi sıralama hakkında kesin sonuçlar çıkarmak kolay değildir. İdeal bilgi, bir örneğin optimumluğu veya optimumluğu arasındaki mesafe ihtimalini içermesidir. Ne yazık ki, bu ilişkiler genellikle yalnızca niteliksel olarak bilinir: daha büyük bir numune, optimum içerdiği için daha küçük bir numuneye kıyasla daha olasıdır ve daha büyük bir numunedeki en iyi sıralama da en uygun değere daha yakın olma eğilimindedir. Fakat bu ilişkiler hakkında niceliksel bilgi olmadan, örneklem boyutu seçmek için neredeyse hiç mantıklı yol yoktur. İlke olarak, belirli bir

denemede belirli bir örnekleme prosedürünün belirli bir problem için optimum oluşturacağı belli bir p olasılığı vardır. Bu nedenle, örnekleme esasen değiştirme ile yapılır, çünkü N boyutundaki bir numunede optimum bulunması ihtimali $[1-(1-p)N]'dir.$

Zorluk p yi tahmin etmektir. Temel tek makina probleminde belki de niceliksel bir sonuç çıkarabileceğimiz bir durum vardır.

Sıradaki ilk konumu, sonra ikinci ve benzeri atayarak bir sıralama oluşturulduğunu varsayalım. İlk sıra konumunu atamak için, rastgele bir cihazın kullanıldığını ve her işin 1 / n olasılıkla bu konuma atandığını varsayalım. Bu atamadan sonra kalan her işin 1 / (n - 1) olasılıkla ikinci konuma atandığını varsayalım. Bu şekilde devam edersek, her konuma eşit derecede seçici bir cihaz atayacağız. Bu yapıda, tüm n! sıralamalarının örneklemde eşit olarak yer alması muhtemeldir. Eğer optimum eşsiz ise, p = 1 / n! , bu nedenle bu prosedürde N boyutundaki bir örneğin en iyi sırasının 1 - (1 - 1 / n!) olasılıklı bir optimum olduğu sonucuna varabiliriz. En iyi duruma yakınlık konusunda örneklemdeki en iyi sıralamanın olabileceği konusunda niceliksel sonuçların elde edilmesi hala mümkün değildir.

Rassal örnekleme, basit, düz mantık ve sınırlı hesapsal çabasıyla kombinatoryal problemlere iyi çözümler üretmek için bir prosedürdür. Çizelgeleme alanının hem içinde hem de dışında daha karmaşık problemlerde, örnekleme teknikleri etkili sezgisel işlemler sağlamıştır. Bununla birlikte, bir sonraki hesaplama deneylerindeki sonuçların gösterdiğine göre, örnekleme her zaman diğer genel amaçlı sezgisel yöntemlerle rekabet edemez.

Avantajları uygulama kolaylığı ve esnekliktir. Esneklik birçok taktik seçenekten türemektedir. Bu seçenekler, çapraz numune alma için işlerin başlangıç sıralamasını, olasılıkları pozisyonlara atamak için olasılık dağılımının seçilmesini ve numune boyutunun belirlenmesini içerir. Rassal örneklemenin uygulama sanatı, etkili bir örnekleme prosedürüne ulaşmak için bu taktikleri belirtir. Farklı taktikler farklı problem türlerinde iyi performans gösterebilir, bu nedenle belirli bir uygulamaya en

uygun taktiklerin belirlenmesi için bazı deneyler gerekebilir. Son olarak, rassal örnekleme, diğer sezgisel yöntemlerle kombinasyon halinde potansiyel olarak yararlıdır. Örneğin, her rasgele örnek araya girme sezgiseli uygulanabilir; bu kombinasyon sadece temel örnekleme prosedürünü iyileştirebilir.

4.3. Komşuluk Arama Teknikleri

Komşuluk arama yaklaşımında temel unsurlar, bir komşuluk kavramı ve komşuluk üretme mekanizmasıdır. Oluşturan mekanizma, bir dizinin bir çözüm olarak alınması ve ilgili dizilerin bir kümesi oluşturması için bir yöntemdir. Örneğin, komşu çiftli değişim işlemi, bir üretme mekanizması görevi görebilir. Çözüm sıralaması $1,2,3,\ldots\ldots,N$ aşağıdaki sıralardan herhangi biri, tek bir ardışık çiftleri yer değiştirmeyle oluşturulabilir:

$$2,1,3,4,\ldots\ldots\ldots n-2,n-1,n$$
$$1,3,2,4,\ldots\ldots\ldots n-2,n-1,n$$
$$\ldots\ldots\ldots$$
$$1,2,3,4,\ldots\ldots\ldots n-1,n-2,n$$
$$1,2,3,4,\ldots\ldots\ldots n-2,n,n-1$$

Bu, özel türetme mekanizması için çözüm sıralamasının komşuluğu olarak adlandırılan (n - 1) ayrı dizinin bir listesidir. Komşuluk üretmenin diğer yöntemlerini öngörmek zor değildir. Son araya girme mekanizması, çözümün son işini diğer konumlara ekler. Bu durumda eğer çözüm sıralaması 1, 2, 3, . . ., ,n şeklinde olursa çözümün komşuluğu (n - 1) sıralamalarının bir listesi olarak aşağıdaki gibi olur:

$$n,1,2,\ldots\ldots\ldots n-1$$
$$1,n,2,\ldots\ldots\ldots n-1$$
$$\ldots\ldots\ldots$$
$$1,2,3,.\ldots\ldots n,n-1$$

Bir türetme mekanizmasının seçimi komşuluk büyüklüğünü belirler. Örneğin, komşuluk, yalnızca ardışık değişimlerle değil, tüm çiftleri yer değiştirme tarafından da üretilebilir. Bu çiftleri yer değiştirme komşuluğu, n (n - 1) / 2 dizilerinin bir listesini içerir. Yukarıda açıklanan son araya girme komşuluk genellemesi, j=i iken sıralamadaki işi j nin içindeki i pozisyonuna yerleştirir. Genel olarak, bir çözüm ve bir türetme mekanizması göz önüne alındığında, çözümden, türeten mekanizmanın tek bir uygulaması ile oluşturulabilen herhangi bir sıralama, çözümün komşuluğu olarak tanımlanır. Bu bağlamda, bir arama algoritması, bir türetme mekanizmasının özelliklerini gerektirir. Bir komşuluk arama algoritmasının genel açıklaması aşağıda verilmiştir.

Komşuluk Araması Algoritması,

1. Aşama İlk temel başlangıç çözümü olacak şekilde bir sıralama elde edilir ve bunu performans ölçüsüne göre değerlendirilir.
2. Aşama Çözümün komşuluğundaki tüm sıralamaları oluşturulur ve değerlendirilir. Sıralardan hiçbiri çözümden daha iyi performans ölçüsüne göre değilse, durdurulur. Aksi takdirde devam edilir.
3. Aşama Performans ölçümünü geliştiren komşuluktaki bir sıralama seçilir. Bu sıralamanın yeni çözüm olduğunu düşünülür ve 2. Aşamaya dönülür.

Bu genel çerçevede, belirli taktik seçenekler belirlemeliyiz:

1. Başlangıç çözümünün elde edilmesine yönelik bir yöntem
2. Türetme mekanizması
3. Yeni çözüm olacak belirli bir sıralamanın seçilmesi için bir yöntem

Algoritma 4.1'in arama prosedürü, verilen komşuluk yapısına göre yerel bir optimum olan bir çözüm ile daima sonlanır. Ne yazık ki, terminal sıralamasının küresel bir optimum olup olmadığını bilmek için genel bir yol yoktur. Örneğin, T probleminde MDD'ye göre sıralama, bir çözümün, ardışıkçiftleri yer değiştirme komşuluğuyla ilgili olarak yerel olarak optimal olup olmadığını gösterebilir, ancak çözümün küresel olarak

optimal olup olmadığına bakmaz. Benzer şekilde, Tw probleminin tatmin edici örneği (4.2), yerel optimumluğa eşdeğerdir, ancak küresel optimumluğa eşit değildir.

Diğer arama prosedürleri türlerinde olduğu gibi, temel algoritmayı arttırmak ve çeşitli yollarla global bir optimum bulma şansını artırabilirsiniz, örneğin:

1. Başlangıç çözümü olarak sunulan birkaç sıralama oluşturulur. Her başlangıç çözümü için tam arama prosedürünü kullanılır ve bulunan en iyi terminal dizisini kullanılır.
2. Her komşulukta çözüm üzerinde gelişen tüm sıralamaları takip edilir. Bunların her birini yeni bir komşuluk için çözüm olarak kullanılır.
3. Büyük komşuluklar yaratan bir türetme mekanizması seçilir.

Bu ve diğer çoğaltma metotları son derece mantıklı olmasına rağmen, yine de küresel bir optimumun bulunabileceğine dair bir garanti sunamamaktadır. Yine de, birkaç deneysel çalışma, komşuluk arama algoritmasının bile temel versiyonunun genel amaçlı bir sezgisel prosedür olarak oldukça güvenilir olduğunu göstermiştir.

Komşuluk arama tekniği, genellikle, sıralama problemlerinin çözümü için umut verici bir sezgisel yöntemdir. Bununla birlikte, başlangıç çözümü bulma, türetme mekanizmasını seçme ve yeni çözüm ile ilerme gibi etkili yöntemler içeren çeşitli seçenekler mevcuttur. Bu açık konuların içinde, komşuluk arama prosedürünün uygulanması çok marifetli kalmıştır.

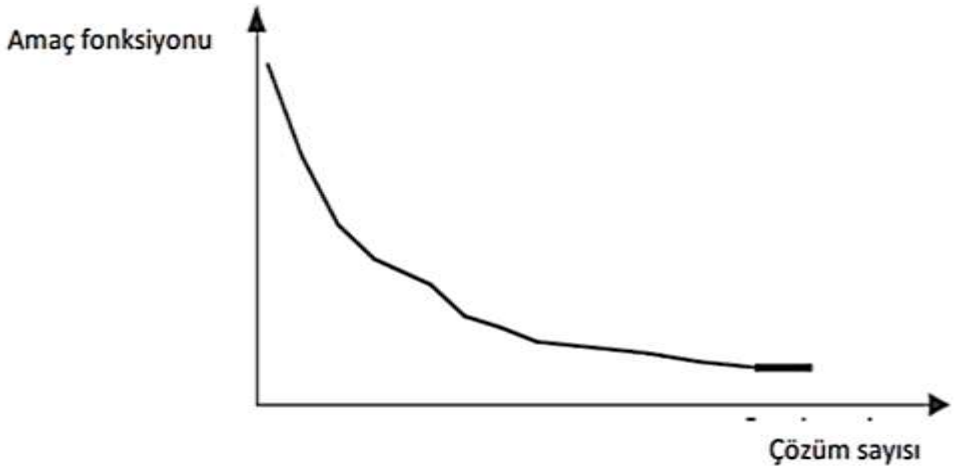
4.4. Tabu Araması

Her yeni çözüm, objektif fonksiyonun daha düşük bir değerini temsil ettiğinden (hedefin en aza indirgendiğini varsayarsak) temel komşuluk arama prosedürüne bazen bir iniş tekniği denir. Çözümün amaç fonksiyonunun değerini çözüm sayısının bir fonksiyonu olarak grafiğe dökerse, grafik azalan bir fonksiyon olarak karşımıza çıkar. Büyük bir problemde, azalma başlangıç aşamalarında hızlı olabilir, ancak Şekil4.1'de olduğu gibi aramanın sonuna doğru daha yavaş olacaktır.

Komşuluk arama yöntemlerinin problemlerinden biri yerel optima'da tuzağa düşme eğilimidir. Şüphesiz, giderek gelişen çözümlerin yolunu izlemek elbette mantıklıdır, ancak böyle bir yol küresel bir optimuma yol açmayabilir. Bazen, eski çözümden daha kötü yeni bir çözüm denemek, tuzaktan kurtulmak ve optimal bir çözüm bulma yolu olarak istenebilir. Bazen daha kötü bir çözüme geçme esnekliği tabu arama yöntemlerinin bir özelliğidir.

Temel olarak, bir tabu arama prosedürü komşuluk araştırmasının revize edilmiş bir şekli olarak görülebilir. Bir komşuluk oluşturulduğunda ve yeni bir çözüm seçildiği her seferde, bir çözümden bir sonraki çözüm değişimini bir hamle olarak adlandırırız. Bir hamle, komşuluk üreten mekanizma ve komşulukta bir çözüm seçme kuralı ile tanımlanır. Tabu aramada, komşuluktaki amaç fonksiyonunun en iyi değerini seçmek teamüldür.

Başlangıçta, bir tabu arama prosedürü komşuluk araştırmasına çok benzer. Bununla birlikte, bir yerel optimuma rastlandığında durmak yerine, bir tabu arama stratejisi, çözüm değeri mevcut çözümden daha kötü olsa bile yeni bir çözüm kabul eder. Elbette, yeni çözüm önceki çözümden daha kötü olduğunda prosedür süresiz olarak devreden çıkabilir. Bu tip bir döngüden kaçınmak için tabu olarak bir önceki çözüme geri hareket belirtiriz.



Şekil 4.1. Komşuluk Aramasındaki Amaç Fonksiyonunun Gelişimi

Aynı düşünceyle, tabu olarak ikinci veya üçüncü önceki çözümlere bir geri hareket belirleyebiliriz. Başka bir deyişle, tabu hamlelerinin bir listesini tutuyoruz, bir hareket uzunluğundan daha uzun olabilecek bir liste. Her aşamada, prosedür komşulukta tabu listesinde olmayanların en iyi çözümünü seçer.

Tabu olarak bir hareketi belirtmek için farklı olasılıklar mevcuttur. Kavramsal olarak en basit olan şey, sıralamaları tabu listesinde tutmak ve böylece daha önce karşılaşılan bir sıralamaya geri dönmeyi yasaklamaktır. Tabu listesi sonlu ve genellikle oldukça küçüktür. Bazı uygulamalarda bir listenin boyutu oldukça etkili olmuştur, ancak tabu aramasının orijinal anlatımı, yedi harekete kadar olan listeleri tavsiye etme eğilimdedir.

Komşuluk arama prosedüründe sonlandırma için dahili bir cihaz bulunurken-yerel optimumun keşfi- tabu arama sonlandırma kuralına dayalıdır. Genellikle, hamle sayısı belirli bir hesaplama çabası sağlamak için başta sabitlenir. Alternatif bir durdurma kuralı, iyileşmenin gerçekleşmediği ardışık hareket sayısını sınırlamaktır.

4.5. Benzetim Tavlaması

Tabu arama, komşuluk aramasının problemlerinden birinin üstesinden gelir yerel optimum tuzağı. Tabu arama savunucuları genellikle yeni bir çözüm seçimi konusunda agresif bir felsefe önermektedir. Bu felsefeye göre komşuluk içindeki en iyi tabu dışı çözüm seçilmelidir. Şekil 4.1'deki grafik açısından, bu taktik her aşamada eğimi mümkün olduğunca dik olarak aşağıya çekme eğilimindedir. Alternatif bir felsefe, eğimi yavaş yavaş aşağıya çekmektir. Bu yaklaşım benzetim tavlama prosedürünün karakteristik özelliğidir.

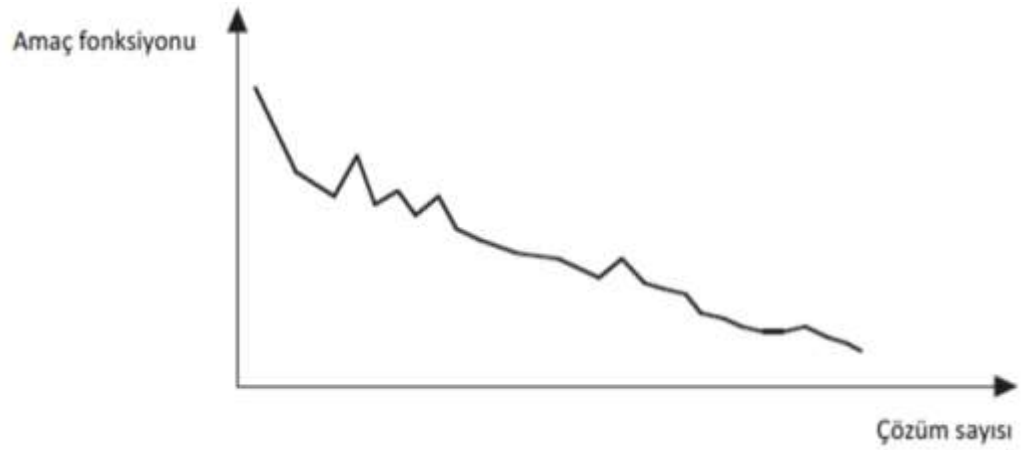
Tavlama, fizik bilimlerinden ödünç alınan bir terimdir. Terim, malzemenin kararlı (donmuş) bir hale gelene kadar yavaşça malzemenin soğutulması işlemini ifade eder. Bu işlemin başlarında, yüksek sıcaklıklarda, malzemedeki tanecikler bazen daha yüksek enerjili durumlara geçer, ancak düşük sıcaklıklarda bu tür davranış daha az olasıdır. Çok düşük sıcaklıklarda, tanecikler, fırsat doğduğunda neredeyse her zaman

daha düşük enerjili durumlara geçer. Eninde sonunda düşük enerjili durumlara doğru hareket donmaya yol açar.

Benzetim tavlamasında, aramaların her bir aşamasının bir önceki aşamada meydana gelen sıcaklıktan daha düşük bir sıcaklıkta gerçekleştirildiğini düşünebiliriz. Amaç fonksiyonunun değeri, soğutulan malzemenin sıcaklığına benzemektedir. Araştırmanın başlangıcında (yüksek sıcaklıklarda) daha kötü bir çözüme geçmek için bir miktar esneklik vardır; ancak daha sonra aramada (düşük sıcaklıklarda) bu esnekliğin daha azı vardır. Dolayısıyla, amaç fonksiyonun değeri, aramanın başlangıcında dalgalanma eğilimi gösterirken, Şekil 4.2'deki gibi aramanın sonuna doğru neredeyse hiç değişmemektedir.

Bu işlemi daha hassas yapmak için, bir amaç fonksiyonunun Z değerini en aza indirgemekle ilgilendiğimizi varsayalım ve bir komşuluk arama mantığını kullanalım. i. aşamadaki amaç fonksiyonu, i. çözümün Z değerine karşılık gelen Z_i 'dir. Prosedür, i. çözümün bulunduğu komşuluktaki çözümlerden rasgele seçer. Z_j amaç fonksiyonu ile j. komşu oluşturulduğunda, bir sonraki çözüm olabilir veya olmayabilir.

$Z_j < Z_i \; ise,$ o zaman, standart iniş yönteminde olduğu gibi, j. komşu bir sonraki çözüm haline gelir. Öte yandan $Z_j \geq Z_i$ mevcut çözümden daha kötü olmasına rağmen, j. komşunun bir sonraki çözüm haline gelme ihtimali hala yüksektir. $\Delta Z = Z_j - Z_i$ olarak (Denklem 4.2) alalım. Daha sonra, i. aşamadaki j. komşunun bir sonraki çözüm haline gelme ihtimali,



Şekil 4.2. Benzetim Tavlamasındaki Amaç Fonksiyonunun Gelişimi

$q_{ij} = \min\{1, e^{-\Delta Z/T(i)}\}$ burada (Denklem 4.3) T(i) i. aşamadaki sıcaklığı belirtir. Bu olasılık fonksiyonunun iki özelliği önemlidir. Birincisi, sıcaklık düştükçe olasılık azalır, diğer şeyler eşit olur. Yani, arama ilerledikçe, daha kötü bir çözüme geçme olasılığı azalıyor. İkincisi, bir adayın bir sonraki çözüm olarak seçilme olasılığı eğer amaç fonksiyonunda bir gelişme varsa her zaman 1'dir; fakat amaç fonksiyonu artarsa, olasılık artışı ile ters orantılı olarak değişir. Son olarak, arama prosedürü bir sıcaklık çizelgesi gerektirir. Çözüm komşuluğundan belirli sayıda deneme yapıldıktan sonra sıcaklığı düşürüp aramaya devam edilir. Örneğin, sıcaklık çizelgesi, $0 < \pi < 1$ olduğunda ve T (1) ortalama işlem süresine eşit olduğunda (Denklem 4.4), $T(i + 1) = \pi T(i)$ ile geometrik bir yol izleyebilir.

4.6. Genetik Algoritmalar

Bir genetik algoritma (GA), gördüğümüz birkaç sezgisel metod ile benzer özelliklere sahiptir, aynı zamanda radikal olarak farklı bir mantık olan bir komşuluk arama prosedürü olarak da görülebilir. Normalde, bir GA, her aşamada b gelecek vaat eden çözümlerin bir listesini tutar ve algoritmik iterasyonlar (yinelemeler) özel bir komşuluk aramak suretiyle daha iyi olanları üretmeyi amaçlar. GA, tek bir sıralamayı değiştirerek bir komşu tanımlamak yerine birinin bazı özelliklerini diğerinin de kalan özelliklerini seçerek iki var olan sıralamayı birleştirir. (Prensipte, bir GA, mevcut iki sıralamadan fazlasını bir araya getirebilir. Yeni adaylar var olanların çocukları olarak görülebilir çünkü terminoloji evrim ve genetikten ödünç alır, beslenir. Böylece, her

kuşakta, eski nesillerin (iterasyonlar) en güçlü (en iyi performans gösteren) hayatta kalanlarından b ebeveynlerinden (sıralamalar) başlanır. Ebeveynlerin çiftleri, çocukları üretmek için genellikle rastgele seçilir. Her ebeveyn genler (alt sıralamalar) çocuğa katkıda bulunur ve mutasyonlar da (rasgele değişiklikler) ortaya çıkabilir. Algoritma, önceden belirlenmiş nesil sayısından sonra sona erer, ancak diğer durdurma kuralları uygulanabilir. Son jenerasyonlardaki tüm hayatta kalanların en güçlüsü çözüm olarak seçilir.

Genetik Algoritma,

1. Aşama Popülasyon boyutu b>2 ve jenerasyon sayısı K yı seçin. Diğer sezgisel yöntemlerle (rasgele arama gibi) b başlangıç çizelgesini seçin. k=0 olarak alın.
2. Aşama k'yı 1 arttırın. Bireylerin çiftlerinden minimum b/2 çocuk üretin (Çocuklar rasgele mutasyonlara maruz kalabilirler).
3 .Aşama Çocuğu değerlendirin. k<K ise tüm ebeveynlerin ve çocukların arasından en iyi b zaman çizelgelerini seçin ve 2. adıma geri dönün. Eğer k = K ise, durdurun (şimdiye kadar bulunan en iyi çizelgenin çözümdür).

Bu genel çerçeve içinde bazı taktiksel soruları cevaplamalıyız:

1. İlk jenerasyon çizelgelerini nasıl elde edeceğiz?
2. Ebeveynler nasıl çocuk üretir?
3. Yetiştirmek için ebeveynleri nasıl eşleştiririz?

Birinci nesil çizelgeler rasgele oluşturulabilir veya sezgisel prosedürlerden birini uygulayarak oluşturulabilir. Örneğin, b farklı öncelik prosedürlerini uygulayarak b çizelgelerini oluşturabilir.

Sıralama problemlerinde, çocuklar üretmenin en basit mekanizması, ilk birkaç iş için bir üst ebeveyni kalan işler için de diğer ebeveyni almak şeklinde izlenir. Bununla birlikte, son birkaç iş sadece diğer ebeveynden kopyalanamaz, çünkü bu yeni

sıralamada işlerin çoğaltılmasına ve diğer işlerin ihmallerine neden olabilir. Bu nedenle, son bir kaç işin bir ebeveynden kopyalanacağı ve ilk işlerin diğer ebeveyni gibi aynı sırada göründüğü tamamlayıcı bir çocukluk oluşturulabilir. Bazı şemalar, daha sonra bir oğul ve bir kız olarak atıfta bulunabilecek her iki çocuk kullanır. Her bir ebeveynden seçilecek işlerin sayısı ikincil bir tasarım seçeneğidir. Bu parametreyi rasgele ayarlayabiliriz veya bu mekanizmaya dayalı olarak olası tüm çocukları üretebiliriz. Buna ek olarak, evrim benzerini takiben, bir GA bir çocuk yaratmada az sayıda rastgele ekleme yaparak yaratılan rasgele mutasyonlara izin verir.

Ebeveynlerin uyumu rasgele olabilir veya bazı sistematik prosedür kabul edilebilir. Örneğin; en iyi ebeveyn b. en iyi ebeveyn ile eşleşebilir, ikinci en iyi ebeveyn (b-1). ebeveyn ile eşleşebilir, böyle devam eder. Alternatif olarak, en iyi ve en iyi ikinci ebeveyn eşleştirilebilir, daha sonra üçüncü ve dördüncü en iyi ebeveyn eşleştirilebilir, böyle devam eder. Doğada, en iyi hayatta kalanların eşleşmelerinden orantılı paylarının üstünde olması beklenir ve bu özellik de taklit edilebilir (Alharkan 2005; Baker ve Trietsch 2009; Pinedo 2005).

BÖLÜM 5. PARÇACIK SÜRÜ OPTİMİZASYONU

5.1. PSO'nun Tanımı ve Tarihçesi

1927 yılında Karl Von Frisch arıların peteklerine geri dönerken beraberinde polen ve nektar ile birlikte bilgi de getirdiklerini keşfetmiş ve iletişim becerilerini anlamaya çalışmıştır. İşçi arılardan biri verimli bir alan bulduğunda diğerlerinin de o bölgeye yönelmeleri bunu göstermektedir. Sürüdeki diğer bireylerin çağrısına cevap veren işçi arılar başlangıç konumuna yakın bir şekilde rastal bir noktaya varırlar ve etrafta daha iyi bir noktaya denk gelinirse o noktaya doğru ilerlerler (Clerc 2006).

Parçacık sürü optimizasyonu (PSO) 1995 yılında Kennedy ve Eberhart tarafından geliştirilen bir algoritmadır. PSO popülasyon temelli bir arama algoritması olarak geliştirilmiştir. PSO'nun başlangıcında parçacık olarak isimlendirilen rastsal çözüm değerlerinin oluşturduğu bir popülasyon ile başlanır. Her bir parçacık geçmiş davranışları ile ilişkili olarak dinamik olarak ayarlanan hızları ile arama uzayında hareket ederler. Bu nedenle parçacıkların arama sürecinde daha iyi arama bölgelerine uçma eğilimleri vardır. Literatürde birçok araştırma mevcuttur ve devamlı olarak PSO ile ilgili yapılan çalışmaların sayısı artmaktadır. Evrimsel Hesaplamalar Kongresi başta olmak üzere birçok konferans ve kongrede 1998'den beri PSO ile ilgili çalışmalar ele alınmıştır. 2003 yılında Sürü Zekası ile ilgili ilk IEEE Sempozyumu Indianapolis, Indiana, ABD 'de gerçekleştirilmiştir. PSO'ya adanmış ilk kitap James Kennedy, Russell Eberhart ve Yui Shi tarafından Sürü Zekası (Swarm Intelligence) adıyla yayınlanmıştır (Kennedy ve Eberhart 1995).

PSO algoritmasının başlangıç noktası arı, balık, karınca, kuş gibi hayvanların davranışlarının incelendiği sürü teorisidir. Sürü teorisi, sosyal böcek veya diğer canlı kolonilerinin kollektif davranışlarından esinlenerek algoritma oluşturma veya dağınık

problem çözme planları tasarımına yönelik bir teşebbüsü kapsayan bir alan olarak tanımlanabilir (Bonabeau v.d. 1999). Sürü halinde hareket eden hayvanların yiyeceklerini ararken izledikleri yol ve iletişim şekli PSO algoritması için zemin oluşturmuştur.

Orijinal pso algoritması sosyal model simülasyonundan esinlenerek oluşturulmuştur. PSO kuş, balık sürüleri ve sürü teorisi ile ilişkilidir. Diğer evrimsel algoritmalarda olduğu gibi PSO algoritması da popülasyon temelli başlangıç değerlerinin rasssal atandığı ve popülasyondaki bireyler arasında etkileşim olan arama yöntemidir. Diğer evrimsel algoritmalardan farklı olrak, PSO ‘da her parçacık çözüm uzayında uçmaktadır, geçmiş en iyi değeri bellekte tutmaktadır ve nesilden nesile hayatta kalmaktadır. Ayrıca evrimsel algoritmalardan kıyaslandığında PSO’nun orijinal versiyonu çözüme başlangıçta çok hızlı yakınsarken, hassas ayar yakınsamasında yavaştır (Shi ve Eberhart 1999).

PSO, evrimsel hesaplama tekniklerinden olan Genetik Algoritmalar ile birçok benzerlik göstermektedir. Algoritma rassal çözümlerden oluşan bir popülasyon ile başlatılır ve en iyi çözüm için iterasyonlar yapılarak arama yapar. PSO ‘da parçacık denilen olası çözümler, mevcut en iyi çözümü takip ederek problem uzayında gezinirler ve en iyi çözüm bulununcaya kadar gezinmeye devam ederler.

5.2. PSO Algoritması

Kuş veya balık sürüsü davranışlarından yola çıkılarak PSO algoritması oluşturulmuş popülasyon tabanlı evrimsel arama algoritmasıdır. PSO senaryosunda sürünün bir bölgedeki yiyecek arayışı, sürüye üye kuşların o bölgede rasgele dağılımı ile gerçekleşir. Kuşlar yiyeceğin nerede olduğunu bilmezler. Sürüdeki bütün kuşlar eş zamanlı olarak arama bölgesine yiyecek ararlar. Sonrasında eş zamanlı olarak bir araya gelerek yiyeceğin nerede olduğu konusunda bilgi paylaşımında bulunurlar. Sürü içerisindeki kuşlar yiyeceğe ne kadar yakın olduklarını ve en yakın kuşun pozisyonunu belirler. Sürüdeki kuşlar bu bilgiler ile yiyeceğe ulaşırlar (Hu v.d. 2004).

Belirli bir alanda yiyecek arayan kuş sürüsünün hedefinin alandaki tek bir yiyecek olduğunu varsayalım. Kuşlar yiyeceğin nerede olduğunu bilgisine sahip değiller fakat her bir iterasyon sonunda yiyeceğin ne kadar uzakta olduğu bilgisine sahip olsunlar. Bu durumda uygulanacak en iyi strateji yiyeceğe en yakın kuşu takip etmek olacaktır.

PSO bu senaryo çerçevesinde çalışır ve optimizasyon problemlerinin çözümü için kullanılır. Algoritmanın çözümünde kuşlar parçacık olarak isimlendirilir. Tüm parçacıkların, optimize edilecek bir uygunluk fonksiyonu tarafından değerlendirilen bir uygunluk değeri ve uçuşların yönlendirilmesi için kullanılan hız bilgileri vardır. Parçacıklar problem uzayında mevcut optimum parçacıkları takip ederek uçarlar.

PSO diğer optimizasyon teknikleri ile kıyaslandığında bazı özellikleri ile diğer optimizasyon tekniklerinden daha üstün olduğu görülmektedir (Del Valle v.d. 2008).

Görece olarak basit bir yapıya sahip olan algoritma çok fazla parametreye sahip değildir. Bu sayede algoritma diğer optimizasyon yöntemlerine nazaran uygulanması daha kolay olan bir yönteme sahiptir. Ayrıca PSO ‘da arama yapılan çözüm uzayında en iyi değerlere sahip parçacıktan yararlanılır ve çözüm uzayında değişiklik olmaz, genetik algoritmada kullanılan kötü sonuçların devre dışı bırakılması PSO ‘da söz konusu değildir. Bu özellik de arama uzayında PSO’ nun yerel optimumlara takılmasını engellemektedir.

PSO ‘da parçacık olarak adlandırılan çözümler bireylerden oluşturmaktadır. PSO ‘daki her parçacık D-düzlemli problem uzayında her birey kendi ve sürüdeki diğer bireylerin uçuş tecrübeleri ile dinamik olarak ayarlanan süreç ve hızlarda uçmaktadırlar. PSO algoritmasında parçacıklar hız ve konumlarını her iterasyonda değiştirmektedirler. Bu çerçevede hız güncelleme denkleminin üç ayrı bileşeni bulunmaktadır. Hız güncelleme denklemindeki ilk kısım bir önceki hızın eylemsizliğini gösterirken bazı kaynaklarda “momentum kısmı” olarak adlandırılmaktadır. Gerçek dünyada maddelerin eylemsizliği kanunundan dolayı hız birden bide değişemez. Var olan bir hızın değişimi mümkün olmaktadır. İkinci kısım parçacığın kendi kendine düşünmesini ve geçmiş tecrübesini gösteren “idrak (algılama) kısmı” olarak

adlandırılmaktadır ve nihayet üçüncü kısım ise parçacıklar arasındaki işbirliğini ve birlikte hareketi temsil eden "sosyalleşme kısmı" olarak adlandırılabilmektedir. Bu kısımda ise parçacıklar sürünün uçuş tecrübesinden yararlanmaktadırlar. Eğer hızlanmanın toplamı bireylerin maksimum hızı geçmesine sebep olursa o zamanda hız maksimum hız doğrultusunda sınırlandırılmaktadır. Mevcut pozisyon ve hedef pozisyon arasındaki alan dikkate alınarak maksimum hız paremetresi algoritmanın kullanıcısı tarafından saptanır ve belirlenir (Xie v.d. 2002). Sürüdeki bütün bireyler maksimum hız çerçevesinde ilerler ve hızını maksimum hız doğrultusunda ayarlarlar.

Diğer taraftan PSO algoritması üç adımdan oluşan temel yapıya sahiptir. Bu adımlardan birincisi; parçacıkların konumlarının ve hızlarının oluşturulmasını kapsarken ikinci adımda hızların güncellenmesi ve yenilenmesi el alınır ve son adımda ise bireylerin konumların yenilenmesi ön plana çıkmaktadır. Parçacıklar konum uzayındaki noktalar olarak ele alınabilir. Parçacıklar konumlarını her iterasyonda güncellenen hızlarını dikkate alarak değiştirmektedirler. Parçacıklara başlangıç konumu ve hızı atanırken şu denklemler kullanılabilir (Hassan v.d. 2005);

d boyutlu bir problemde sürüdeki i. parçacığın konum vektörü (Denklem 5.1) $X_i = (x_{i1} , x_{i2} , \dots , x_{ik} , \dots , x_{id})$ olsun. Sürünün en iyi parçacığı diğer bir değişle sürüde en iyi uygunluğa sahip olan parçacık (global en iyi) gbest olarak adlandırılsın. Sürüdeki her bir parçacığın nesiller boyu elde ettiği en iyi uygunluk değeri de (kişisel en iyi) pbest olarak adlandırılsın. Buna göre sürüdeki i. parçacığın pbest (Denklem 5.2) değerleri

$P_i = (P_{i1}, P_{i2}, \dots \dots , P_{ik}, \dots \dots , P_{id})$ olarak tespit edilecektir. i. parçacığın yer değişim vektörü yani hız vektörü (Denklem 5.3) ise $V_i = (V_{i1}, V_{i2}, \dots \dots . . V_{ik}, \dots \dots . V_{id})$ olarak kabul edilirse göre nesiller boyunca güncellenen i. parçacığın hız vektörü ve buna bağlı olan konum vektörünün formülleri şu şekilde (Denklem 5.3, Denklem 5.4) oluşacaktır;

$$V_{ij}^{t+1} = wV_{ij}^{t} + c_1 r_1 \left(P_{ij} - x_{ij}\right) + c_2 r_2 \left(G_{gbest} - x_{ij}\right) \qquad (5.3)$$

$$X_{ij}^{t+1} = X_{ij}^{t} + V_{ij}^{t+1} \quad (5.4)$$

$t; iterasyon\ sayısı, j: 1, 2, \ldots\ldots, d$

Denklemler popülasyondaki parçacıkların uçuş yansımalarını belirlemektedir. Konum denklemi uçan parçacıkların konum yenilemelerini gösterir.

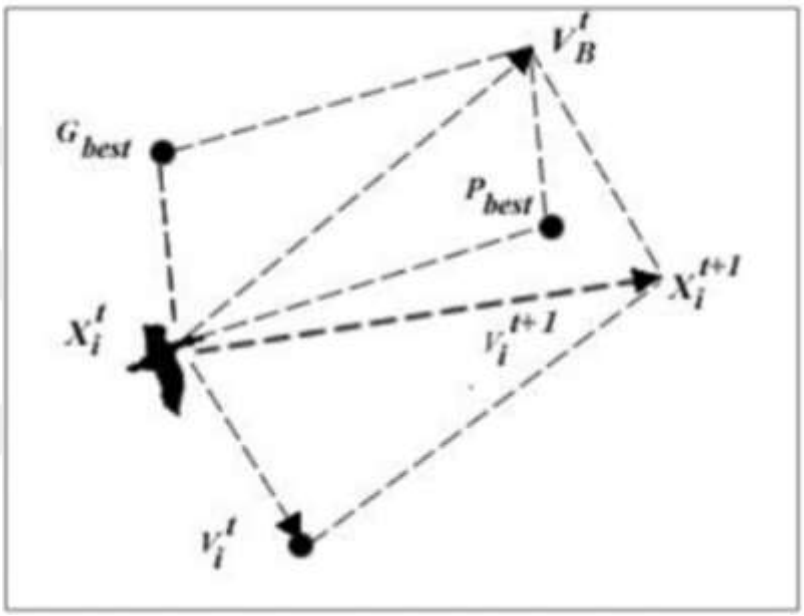
Denklem 5.3 sonunda t. İterasyonda i. parçacığın (t+1). İterasyondaki hız vektörü bulunmuş olur. Formül 5.4.'te ise bulunan hız vektörü (V_{ij}^{t+1}), i. parçacığın t. iterasyondaki pozisyon vektörüne eklenerek (t+1). iterasyondaki pozisyon vektörü (X_{ij}^{t+1}), bulunmuş olur. Bulunan bu pozisyon vektörü probleme yeni bir çözüm önerisi demektir. Buradaki w atalet ağırlığı değeridir. $c_1\ ve\ c_2$ değerleri ise hızlandırma katsayılarıdır. $r_1\ ve\ r_2$ ise [0,1] rasgele değerler alan ve PSO'nun rasgeleliğini sağlayan parametrelerdir.

$$V_{ij}^{t+1} = \underbrace{wV_{ij}^{t}}_{\text{Birinci Kısım}} + \underbrace{c_1 r_1 (P_{ij} - x_{ij})}_{\text{İkinci Kısım}} + \underbrace{c_2 r_2 (G_{gbest} - x_{ij})}_{\text{Üçüncü Kısım}} \quad (5.5)$$

Yukarıdaki ifadede görüldüğü gibi denklem 5.5 üç kısımdan oluşmaktadır; birinci kısım hız değerinin atalet ağırlığını göstermektedir. Parçacıkların hız değerlerinde ani değişikliklerin olmaması gerekmektedir, dolayısıyla parçacıklar bir önceki hızlarına bağlı kalarak hız güncellemesi yapmalıdırlar. Aksi takdirde parçacıklar çözüm uzayında anlık yön değişiklikleri yaparak uygun bir arama gerçekleştiremeyeceklerdir. İkinci kısmı oluşturan kişisel hafıza kısmı aracılığıyla paracıkların konumları, geçmişte elde edebildikleri en iyi konuma (pbest) doğru çekilmektedir. Burada elde edilen değer $c_1 r_1$ değeri ile ölçeklendirilmektedir. Formül 5.3'ün üçüncü kısmı ise sosyal hafıza kısmı oluşturmaktadır, bu kısımda parçacıklar sürünün en iyi konum değerine (gbest) doğru çekilirler (Eberhart ve Kennedy 1995). Bu kısımda da elde edilen değerler $c_2 r_2$ değeri ile ölçeklendirilmektedir.

Hız denkleminde eğer üç kısmın toplamı kullanıcı tarafından belirlenen sabit bir sayıyı geçerse hızın o boyuttaki bileşeni Vmax'a atanır. Hızın her boyuttaki bileşeni Vmax

ile sınırlandırılmalıdır çünkü hız maksimum hızı geçememektedir. Dolayısıyla Vmax PSO kullanıcısı tarafından belirlenen önemli bir paremetredir. Büyük Vmax değeri parçacığın en iyi noktadan uzaklaşıp geçerek uçmasına, küçük Vmax değeri ise parçacıkların yerel optimum etrafında takılmasına ve daha iyi noktalara uçmamasına imkan vermektedir. Genellikle Vmax kullanıcı tarafından sabit bir değer olarak atanmaktadır. Ama iyi tasarlanmış ve dinamik olarak değişen Vmax PSO'nun performansını arttırabilmektedir (Del Valle v.d. 2008; Helwig v.d. 2009).



Şekil 5.1. Parçacığın Pozisyon Değiştirmesi

X_i^t: $i.parçacığın\ t.iterasyondaki\ konumu,$

V_i^t: $i.parçacığın\ t.iterasyondaki\ hızı,$

G_{best}: $Sürüdeki\ en\ iyi\ konuma\ sahip\ parçacığın\ konumu,$

P_{best}: $i.parçacığın\ kişisel\ en\ iyi\ konumu,$

V_B^t: $t.iterasyonda\ G_{best}\ ve\ P'_{best}\ in\ bileşkesi,$

X_i^{t+1}: $i.parçacığın\ (t+1).\ iterasyondaki\ konumu,$

V_i^{t+1}: $i.parçacığın\ (t+1).iterasyondaki\ hızıdır.$

5.2.1. Başlangıç değerleri

Sürü içinde toplam p tane parçacık olduğunu kabul edilirse, bu p tane parçacığın konum ve hız değerleri aşağıdaki formüllere (Denklem 5.5) göre hesaplanır.

$$X_i = X_{min} + (X_{max} - X_{min}) * random( )$$

$$V_i = V_{max} * random( )$$

$$V_{max}: Hız\ değişkeninin\ alabileceği\ en\ büyük\ değer, i = 1, 2, \dots \dots, p \qquad (5.6)$$

5.2.2. Konum değeri

PSO da her bir parçacığın pozisyon değeri ilgili probleme yeni bir çözüm önerisi getirilmektedir. $[X_{min}, X_{max}]$ çözüm aralığında değerler alan bir parçacığın konum değeri X_i ile belirtilen aralığın dışına çıkarsa parçacığa çeşitli kısıtlamalar getirilebilir. Bu kısıtlamalar şu şekilde (Denklem 5.6) gibi tanımlanabilir (Robinson ve Rahmat-Samii 2004).

$$Eğer\ X_i > X_{max}\ ise\ X_i = X_{max}\ ve\ V_i = 0;$$
$$Eğer\ X_i < X_{min}\ ise\ X_i = X_{min}\ ve\ V_i = 0;$$
$$Eğer\ X_i > X_{max}\ veya\ X_i < X_{min}\ ise\ V_i = -V_i;$$
$$Eğer\ X_i > X_{max}\ veya\ X_i < X_{min}\ ise$$
$$X_i' nin\ uygunluk\ değerini\ en\ kötü\ uygunluk\ değeri\ yap$$
$$X_{max} : Arama\ uzayının\ üst\ sınırı$$
$$X_{min} : Arama\ uzayının\ alt\ sınırı \qquad (5.7)$$

5.2.3. Hız değeri

Hız değeri, bir parçacığın çözüm uzayında arama yapmasını sağlayan ve parçacığı yönlendiren en önemli etken olarak dikkat çekmektedir. Hız değerleri pozitif ve negatif değerler alıp parçacıkların çözüm uzayında çok yönlü hareket ederek arama yapmasını sağlamaktadırlar. Hız değeri kontrol edilemediği takdirde parçacıklar çözüm alanının

dışına çıkabilir ve uygun olmayan değerler alabilir. Bunu engellemek için hız değerine Vmax (Denklem 5.8) gibi bir kısıtlayıcı bir limit konulmuştur. Vmax değerinin belirlenmesi probleme göre farklılık gösterebilir. Bazı uygulamalarda Vmax=Xmax olarak kullanılmaktadır (Shi ve Eberhart 1999). Diğer bazı uygulamalarda ise Vmax arama uzayındaki parçacığın konum vektörünün her bir boyutunun alabileceği en büyük değer ile en küçük değerin farkının %10-20 'si aralığında bir değer almaktadır (Robinson ve Rahmat-Samii 2004).

$$V_{max} = (X_{max} - X_{\min}) * \%R \qquad (5.8)$$
$$R \in [10 - 20]$$

Eğer parçacığın hız değeri V_{max} limitini aşarsa aşağıdaki gibi bir kısıtlama getirilir.

$$Eğer\ V_i > V_{max}\ ise\ V_i = V_{max};$$
$$Eğer\ V_i < -V_{min}\ ise\ V_i = -V_{min}$$

5.2.4. Atalet ağırlığı

Kennedy ve Eberhart tarafından önerilen PSO'nun ilk halinde atalet ağırlığı bulunmamaktaydı (Kennedy ve Eberhart 1995). Eberhart ve Shi 'nin daha sonra yaptığı bir çalışmada ise hız güncelleme formülünde birinci kısma atalet ağırlığı (inertia weight) bir çarpan olarak eklenmiştir (Shi ve Eberhart 1999). Böylelikle, atalet ağırlığı kullanılarak parçacığın bir önceki hızının yeni hızına etkisi kontrol altına alınmıştır. Atalet ağırlığının büyük değerler alması, parçacığın çözüm uzayında daha genel (global) aramalar yapmasını, pbest, gbest değerlerinden çok etkilenmeden daha araştırmacı bir yapıda çalışmasını sağlamaktadır. Atalet ağırlığının küçük değerler alması ise parçacığın çözüm uzayında daha bölgesel aramalar yapmasını, pbest ve gbest değerlerinden daha fazla faydalanarak en uygun çözüme yakınsama yapmasını sağlamaktadır.

Atalet ağırlığı, bütün arama işlemi boyunca sabit bir değer olarak kullanılabildiği gibi başlangıçta büyük değer alarak iterasyonlar ilerledikçe azalacak şekilde dinamik

olarak kullanılabilir (Shi ve Eberhart 1999). Dinamik kullanımda, ilk iterasyonlarda parçacıklar daha genel aramalar yaparak çözüm uzayını tararken, ilerleyen iterasyonlarda arama işlemi daha ayrıntılı ve bölgesel bir hal almaktadır. Genel olarak uygulamalarda atalet ağırlığının (w) en büyük değeri $w_{max} = 0{,}9$ en küçük değeri ise $w_{min} = 0{,}4$ olarak alınır. Aşağıdaki formülde de (Denklem 5.9) görüldüğü gibi iterasyon sayısı atalet ağırlığının dinamik değerini belirlemede rol alır.

$$w = w_{max} - \frac{(w_{max} - w_{min})}{iterasyon\ sayısı} * t \qquad (5.9)$$

$t: mevcut\ iterasyon$

Denklem 5.9'de görüldüğü gibi başlangıçta büyük değer alan atalet ağırlığı iterasyonlar ilerledikçe daha küçük değerler almaya başlayacaktır.

5.2.5. Hızlandırma katsayıları

Hızlandırma katsayılarından c_1 parçacıkların pbest değerine, c_2 ise gbest değerlerine doğru çekilmesini kontrol eden katsayılardır. Hızlandırma katsayılarının büyük değerler alması parçacıkların birbirinden uzaklaşıp ayrılmalarına sebep olurken, küçük değerler alması parçacıkların hareketlerinin kısıtlanmasına ve çözüm uzayının yeterince taranamamasına sebep olmaktadır. Hızlandırma katsayıları problemin türüne göre değişik değerler alabilir. $c_1 = c_2 = 2{,}0$ genel olarak önerilen değerdir. $c = c_1 + c_2$ iken c ne kadar büyük değer alırsa arama algoritmasının optimum değer etrafında yaptığı salınım miktarı artar. Ayrıca c_1 ve c_2 katsayıları birbirine eşit olmak zorunda değildirler, farklı değerler alabilirler (Del Valle v.d. 2008).

5.2.6. Uygunluk fonksiyonu

Uygunluk fonksiyonu parçacığın pozisyon vektörünü kullanarak ve varsa problemle ilgili kısıtlamaları da göz önünde bulundurarak uygunluk değeri üreten bir fonksiyondur. Uygunluk değeri, aynı zamanda parçacığın çözüm kalitesini de belirler. Pbest ve gbest değerleri de uygunluğu en iyi olan parçacıklardan seçilmektedir.

5.2.7. Kişisel en iyi değeri

Parçacıklar, iterasyonlar boyunca en iyi konum değerini ararlar. Her bir iterayonda yer alan parçacık, geçmişte bulduğu en iyi konum değeri ile iterasyonda elde ettiği yeni konum değerini kıyaslamaktadır. Eğer yeni konum değeri o ana kadarki en iyi kişisel konum değerinden daha uygun bir değere sahipse yeni kişisel en iyi konumdur. İşte parçacığın iterasyonlar boyunca, sürüdeki diğer parçacıklarla kıyas etmeksizin sadece kendi geçmişindeki değerlere bakarak bulduğu en iyi değerine "kişisel en iyi değer" (pbest) denir. Aynı zamanda pbest değeri parçacığın geçmiş tecrübelerini gösteren bir değerdir. Formül 5.1. deki hız güncelleme denkleminde pbest değeri parçacığın konumunu pbest konumuna doğru çeker.

5.2.8. Global en iyi değer

Sürüde iterasyonlar boyunca parçacıkların konum değerleri hesaba katıldığında bulunan en iyi uygunluğa sahip parçacığın konum değerine global en iyi değer (gbest) denir. Yeni iterasyonda bulunan en iyi değer gbest den daha iyi uygunluğa sahipse yeni gbest değeri olur. Formül 5.1. deki hız güncelleme denkleminde gbest değeri sürüdeki bütün parçacıkların konumunu gbest konumuna doğru çeker. Sürü en iyi küresel değere doğru yol almaktadır. Sürünün içerisinde yer alan her bir parçacığın veya bireyin rolü en iyi global değere ulaşmaktır. Değer, sürünün uzay içerisindeki konumunun geliştirilmesine ve yenilenmesine sebebiyet vermektedir. Uzaydaki her bir parçacık başlangıçta kendi kişisel en iyi değerlerine sahipken çözüme en yakın olan parçacığın sosyal iletişimi sayesinde he birlikte topluluk olarak aldıkları konumla birlikte global en iyi değere ulaşmaktadır.

5.2.9. Sonlandırma kriteri

PSO 'da sonlandırma kriterinin oluşması için bazı durumların gerçekleşmesi gerekmektedir bunlar, kullanıcı tarafından belirtilmiş sabit bir iterasyon sayısına ulaşılması, daha önceden belirlenmiş bir CPU çalışma zamanına ulaşılması veya son

iki iterasyonun gbest değerleri arasındaki farkın kullanıcı tarafından belirlenmiş bir değerin altına düşmesi durumudur.

PSO algoritmasının kolay kuruluma sahip ve hesaplama gücü yüksek olan basit bir yapıya sahiptir. PSO kurulumunun orijinal yapısına ait farklı gösterim şekilleri vardır. Bunlardan bazıları şu şekilde gösterilmektedir;

PSO paremetre katsayılarının ve sürü büyüklüğünün değerlerini belirle
Parçacıkların konumlarının, hızlarının başlangıç değerlerini belirle
Parçacıkların kendi geçmiş en iyi değerlerinin başlangıç değerlerini belirle
Sürüdeki en iyi parçacığı sapta

Döngü
    Hızları güncelle
    Konumları güncelle
    Parçacıkların kendi en iyi değerlerini sapta
    Sürüdeki en iyi değere sahip parçacığı sapta
    Yerel arama (isteğe bağlı)
Durdurma kriteri (Tchomté ve Gourgand 2009).

Tablo 5.1. PSO Algortiması

BAŞLANGIÇ	
Popülasyon	Parçacığın hız ve pozisyon değerlerini oluştur.
TEKRAR	
For I = 1 popülasyon	
	Uyum değerlerini hesapla;
	P_{best} değerini güncelle;
	G_{best} değerini güncelle;
	Hız ve pozisyon değerlerini güncelle
END of FOR	
UNTIL	Durdurma ölçütünü tanımla
END	

BÖLÜM 6. EŞİT OLMAYAN HAZIRLAMA SÜRELİ TEK MAKİNE TOPLAM AĞIRLIKLI GECİKME PROBLEMLERİNİN PARÇACIK SÜRÜ OPTİMİZASYONU İLE ÇÖZÜMÜ

6.1. Deneysel Dizayn

Parçacık sürü optimizasyonunun çözümü ve ispatı için deneysel dizayn tasarlanmıştır. Bu doğrultuda PSO tabanlı çözüm rassal olarak üretilen 4500 örnek seti kullanılarak 100 kez tekrarlanmıştır. Proses zamanı (t_i), gecikme cezası (w_i) ve iş sayısı aşağıdaki tablo 6.1 'deki gibi gösterilmiştir. Burada teslim tarihine oranlı aralık (RDD) ve ortalama gecikme faktörüne (TFF) ait oransal aralık $\{0.1, 0.3, 0.5, 0.7, 0.9\}$ arasından seçilmiştir. d_i tamsayı olarak $[P\left(1 - TF - \frac{RDD}{2}\right), P(1 - TF + \frac{RDD}{2})]$ aralığından üretilmiştir. Burada i iş numarasını göstermek üzere, P toplam proses zamanı $\sum_{i=1}^{n} P_i$ göstermektedir. r_i hazırlık süresinin seçilmesinde $0\ ve\ \beta \sum P_j$ aralığı kullanılır. Bu aralıklar dahilinde uniform dağılım aralıklarıdır ve örnekler oluşturulurken tam sayı olarak rassal seçim yapılır (Cakar ve Koker 2015).

Tablo 6.1. Üretilen Problemlerin Parametreleri

Elemanlar	Dağılım Aralıkları
İşlem Zaman Aralıkları	[1-10], [1-50], [1-100]
Ağırlık Aralıkları	[1-10], [1-50], [1-100]
İş Sayısı	50,100,200,300,400
TF	0.1,0.3,0.5,.0.7,0.9
RDD	0.1,0.3,0.5,.0.7,0.9
β	0.0,0.5,1.0,1.5

6.2. Öncelik Kuralları

Üretim sistemlerinde kısa zamanda en iyi çözümler vermesi için tasarlanmış olan öncelik kuralları (priority rules), yaygın olarak kullanılan bir yöntemdir. Öncelik kuralları tezgahtaki kuyrukta bekleyen işlemlerin hangi sırayla işlem göreceğinin belirlendiği kuraldır. Bir fonksiyon olarak kullanılan öncelik kuralı, işlem görecek işin seçiminde tezgah bilgileri, iş bilgileri, sistem bilgileri gibi verileri kullanarak her bir işe öncelik değeri atayan bir fonksiyondur. Öncelik değerine göre atama sürecinin devamında en önclikli iş tezgahta işlem görmektedir.

Öncelik kuralları çeşitli şekilde sınıflandırılmaktadır (Pinedo 2005). İlk olarak statik ve dinamik olarak ikiye ayrılırlar. Burada statik kurallar zamana bağlı olarak işlemeyen kurallardır. Bu tip kurallar sadece iş veya tezgah bilgilerine göre çalışır. Bu tür kurallara SPT (Shortest Processing Time – En Küçük İşlem Süresi) örnek olarak gösterilebilir. Dinamik kurallar ise zamana bağlı olarak işleyen kurallardır. Bu tip kurallara örnek olarak MST (Minimum Slack Time – En Küçük Bolluk) kuralı verilebilir.

Öncelik kurallarının sınıflandırılmasındaki ikinci yöntem ise, kuralların kullanıldığı bilgiye göre sınıflandırılmasıdır. Bu sınıflandırmaya göre kurallar yerel ve bütünsel olarak ikiye ayrılırlar (Pinedo 2005).

Aşağıda genel olarak kullanılan öncelik kuralları açıklanmıştır. Bu öncelik kuralları PSO algoritmasının çözümünde başlangıç çözümlerinin oluşturulması için kullanılmıştır.

EDD (Earliest Due Date – En Erken Teslim Zamanı); Kuyrukta bulunanlar arasında "en erken teslim tarihi olan" önce işle görür. Tezgahta işlenecek işin belirlenmesinde teslim zamanları kullanılır.

$\min(d_i)$

SPT (Shortest Processing Time – En Küçük İşlem Süresi); Kuyrukta bulunanlar arasında "en kısa" işlem süresi olan önce işlem görür. Tezgahta işlenecek olan iş, işlerin o tezgahtaki işlem sürelerine göre seçilir.

$$\min(p_i)$$

LPT (Longest Processing Time); Kuyrukta bulunanlar arasında "en uzun" işlem süresi olan önce işlem görür. SPT kuralının tam tersidir.

$$\max(p_i)$$

CR (Critical Ratio – Kritik Oran); Kuyrukta bulunan işler kritik oranına göre seçilir. En küçük kritik orana sahip olan iş ilk önce işlem görecek iştir.

$$\min\left(\frac{d_i}{p_i}\right)$$

WSPT (Weighted Shortest Processing Time); Ağırlıklandırılmış SPT/EDD kuralı.

$$\max\left(\frac{w_i}{p_i}\right)$$

WDD (Weighted Due Date); Ağırlıklandırılmış teslim zamanı kuralı.

$$\max\left(\frac{w_i}{d_i}\right)$$

WPD (Weighted Processing Due Date); Ağırlıklandırılmış proses teslim zamanı kuralı.

$$\max\left(\frac{w_i}{p_i d_i}\right)$$

FCFS (First Come First Served – İlk Gelen İlk İşlem Görür); İlk gelen parça ilk önce işlem görür. Tezgahta işlem görecek işler üretim sistemine geliş zamanlarına göre seçilir.

6.3. Problemin Çözümü

Parçacık Sürü Optimizasyonu sahip olduğu yapı sayesinde çözümü görece olarak daha basit olan bir algoritmadır. Doğadaki kuş sürülerinden esinlenerek oluşturulan algoritma, bireyler arası etkileşime sahip bir arama yöntemidir. Algoritmanın 1995 yılında J.Kennedy ve R.C.Eberhart oluşturulmasından itibaren çözümlerin daha iyi sonuç vermesi amacıyla çeşitli çalışmalar yapılmıştır.

Bu bölümde eşit olmayan hazırlama süreli tek makine toplam ağırlık gecikme problemlerinin çözümü için parçacık sürü optimizasyonu algoritması kullanılmıştır. Burada kullanılan PSO algoritması Ebert ve Shi 'nin 1999 yılında PSO algoritmasına eklediği w (inertia weight) atalet ağırlığı dahil edilerek geliştirilen algoritmadır. Bu atalet ağırlığının algoritmaya dahil edilmesi ile birlikte parçacığın hızının yeni hızına etkisi kontrol altına amaçlanmış ve parçacığın daha araştırmacı bir yapıya sahip olması sağlanmıştır.

Parçacık sürü optimizasyonu algoritmasının çözümünde başlangıç değerleri rassal olarak belirlenirken, yapılan çalışmada parçacıkların ilk adımlarının 8 öncelik kuralları değerleri ile başlatılmıştır. Bu doğrultuda öncelik kuralları olan EDD, SPT, LPT, CR, WSPT, WDD, WPD, FCFS kullanılmıştır.

PSO'nun Çözüm Adımları ve Kullanılan Parametreler;

Adım 1; Başlangıç parametrelerinin belirlenmesi (Denklem 6.1),

$$x_i = x_{min} + (x_{max} - x_{min}) * r_1$$
$$x_{min} = 0,\ x_{max} = 4$$
$$v_i = v_{max} * r_2$$
$$v_{max} = (x_{max} - x_{min}) * \%R \qquad (6.1)$$

$R \in [10-20], 15\ olarak\ alınmıştır.$

$r_{1,} r_2 = (0,1)\ düzgün\ dağılıma\ uyan\ rassal\ sayı$

$i: 1,2, \ldots\ldots, 10\ (parçacık\ sayısı)$

Adım 2; İterasyonların Çözümüne Başlanması (Denklem 6.2);

$$t = t + 1 \tag{6.2}$$

Adım 3; Atalet Ağırlığının Belirlenmesi (Denklem 6.3);

$$w = w_{max} - \frac{(w_{max} - w_{min})}{iterasyon\ sayısı} * t \tag{6.3}$$

$t; mevcut\ iterasyon, w_{max} = 0{,}9,\ w_{min} = 0{,}4$

Adım 4; Hız Vektörünün Güncellenmesi (Denklem 6.4);

$$V_{ij}^{t+1} = wV_{ij}^{t} + c_1 r_1 \left(P_{best} - x_{ij}\right) + c_2 r_2 (G_{best} - x_{ij}) \tag{6.4}$$

$t: iterasyon\ sayısı, j: 1,2, \ldots\ldots, d$

$c_1, c_2 = 2$

$r_{1,} r_2 = (0,1)\ düzgün\ dağılıma\ uyan\ rassal\ sayı$

Adım 5; Konum Vektörünün Güncellenmesi;

$$X_{ij}^{t+1} = X_{ij}^{t} + V_{ij}^{t+1}$$

Adım 6; Sıralamanın Bulunması;

Adım 7; Pbest değerinin güncellenmesi;

Adım 8; Gbest değerinin güncellenmesi;

Adım 9; Durma Kriteri

6.3.1. Sıralama kuralının uygulanması ve bir parçacık için gösterimi

PSO'da aranacak çözüm alanı bir matris olarak gösterecek olursak. Bu matrisin her satırı bir parçacığı temsil eder ve eşit olmayan hazırlama süreli tek makine toplam ağırlıklı gecikme problemi için bir iş sırasını temsil eder. Her iş sırasının toplam ağırlıklı gecikmesi kişisel en iyi dereceyi verir ve en iyi toplam ağırlıklı gecikme olan global olarak en iyiyi verecektir.

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & \cdots & x_{2n} \\ \vdots & \vdots & & & \vdots \\ \vdots & \vdots & & & \vdots \\ \vdots & \vdots & & & \vdots \\ \vdots & \vdots & & & \vdots \\ \vdots & \vdots & & & \vdots \\ \vdots & \vdots & & & \vdots \\ x_{n1} & x_{n2} & \cdots & \cdots & x_{nm} \end{bmatrix}$$

Sıralama kuralı kullanılarak x değerine göre bir iş sırası bulunacak ve bu iş sırasına göre toplam ağırlıklı gecikme değeri hesaplanacaktır (Cakar ve Koker 2015).

Burada iş sırasını bulmak için en küçük x değerinden en büyük x değerine kadar başlaması ve böylece Sij iş programının bulunması gerekir. Tablo 6.2 ‘de gösterildiği gibi iş sıralaması 4-6-5-8-1-7-2-3 ‘dür.

Tablo 6.2. PSO ‘daki bir parçacığın sıralanması.

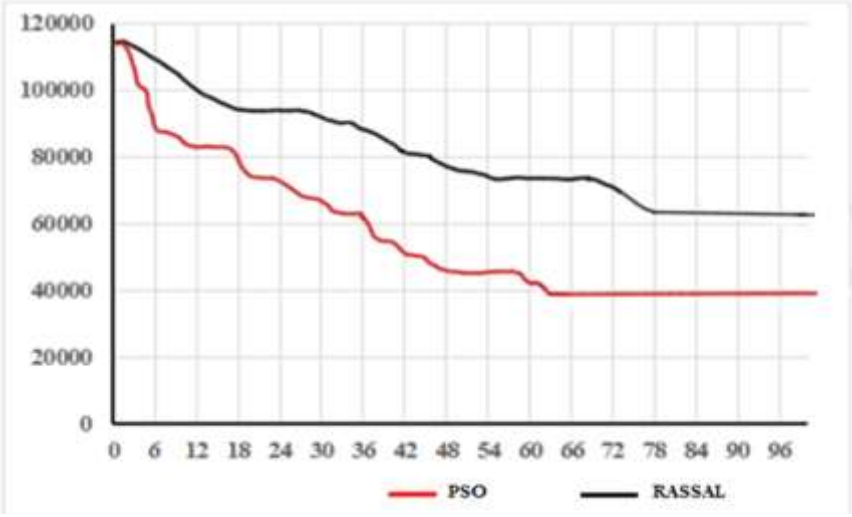
J	1	2	3	4	5	6	7	8
X_{ij}	1,75	2,86	3,12	-0,97	1,12	-0,68	2,22	1,58
S_{ij}	4	6	5	8	1	7	2	3

Tablo 6.3. Üst Sınır (upper bound) ve Alt Sınır (lower bound)' daki Değişmeler

	Toplam Ağırlıklı Gecikmedeki iyileştirmeler (Upper Bound' daki iyileştirmeler)			*Alt sınırdaki (Lower Bound) iyileştirmeler*		
İş Sayısı	Başlangıç Çözümü	PSO Çözümü	Yüzde Değişim	Başlangıç Çözümü	PSO Çözümü	Yüzde Değişim
50	155296	151842	2,22	835694	840699,5	0,59896
100	201564	196982	2,27	1756523	1767512	0,62563
200	405657	392541	3,23	2986510	3005401	0,63254
300	612356	590515	3,57	4056324	4082507	0,64548
400	810719	776254	4,25	4986523	5019561	0,66254

Tablo 6.4. 450000 Örnek için alt sınır (Lower Bound) sonuçlarının t-testi

İş Sayısı	*Daha İyi*	*Eşit*	*En Kötü*	*Toplam*	*t-test*
50	4125	84334	325	90000	5,86
100	7546	77658	3452	90000	5,12
200	7015	79845	2796	90000	7,35
300	6895	77952	4129	90000	5,89
400	6654	78956	3789	90000	6,12



Şekil 6.1. PSO Çözümü

Deneysel dizayn ile üretilen veri setinin PSO algoritması ile çözümü sonucunda Tablo 6.4.'deki sonuçlar elde edilmiştir. Başlangıç çözümü ile kıyaslandığında toplam ağırlıklı gecikmelerin ortalamasının başlangıç çözümlerine göre daha optimum sonuçlar bulduğu görülmektedir. Şekil 6.1.'de 50 adet işin mevcut olduğu iş paketinin PSO algoritması ve rassal çözümlerinin karşılaştırılması grafiksel olarak gösterilmiştir. Bu doğrultuda PSO algoritmasının rassal çözümlemeye göre daha erken iterasyon sayısı ile daha optimum sonuçlar verdiği ve toplam ağırlık gecikme değerlerinin iyileştirilmesini sağladığı görülmektedir.

6.4. Sonuç

Üretim sistemlerinde, üretim planlama ve çizelgeleme verimliliği ve karlılığı etkileyen önemli bir karar verme sürecidir. Özellikle çizelgeleme problemleri yapısı itibariyle en uygun çözümü oldukça zor olan bir problemdir. Yapılan çalışmada "Eşit olmayan hazırlama süreli tek makine toplam ağırlıklı gecikme problemlerinin parçacık sürü optimizasyonu ile çözümü" gerçekleştirilmiştir. Tek makine çizelgelemede ilk olarak deneysel dizayn ile örnek veri seti oluşturulmuştur. Bu çalışmada 50, 100, 200, 300, 400 adet iş paketlerinin her birinden 90.000 adet örnek oluşturularak toplamda 450.000 adet iş paketi ağırlıklandırma kriteri dahil edilmiş olan PSO ile çözülmüştür. PSO 'nun başlangıç çözümünde 10 adet olası çözüm olan parçacığın 8 adedi EDD, SPT, LPT, CR, WSPT, WDD, WPD, FCFS öncelik kurallarına göre çözümlenerek, 2 adet parçacık rassal olarak alınarak PSO'nun çözümüne başlanmıştır. Performans kriteri olarak toplam ağırlıklı gecikme (upper bound) ve alt sınır (lower bound) toplam ağırlıklı gecikme kriterleri hesaplanmıştır. Sonuçlar başlangıç çözümleri ile karşılaştırılarak iyileşme yüzdeleri hesaplanmıştır. Başlangıç çözümüne göre PSO algoritmasının daha kısa sürede ve daha uygun çözümler ürettiği ispatlanmıştır. Bununla beraber 50 işin bulunduğu örneklem seti için PSO çözümü ve rassal çözümler karşılaştırılmış ve PSO algoritmasının rassal çözüme göre daha kısa adımda daha optimum sonuçlar verdiği belirlenmiştir. Bu doğrultuda PSO algoritmasının tek makine çizelgeleme problemlerinde kullanılmasının olumlu sonucuna ulaşılmıştır.

KAYNAKLAR

Van den Akker, J.M., Diepen, G. ve Hoogeveen, J.A., 2010. Minimizing total weighted tardiness on a single machine with release dates and equal-length jobs. Journal of Scheduling, 13(6), pp.561–576.

Akturk, M.S. ve Ozdemir, D., 2001. A new dominance rule to minimize total weighted tardiness with unequal release dates. European Journal of Operational Research, 135(2), pp.394–412.

Alharkan, I.M., 2005. Algorithms for Sequencing and Scheduling, Riyadh: King Saud University.

Aslantaş, V., Doğan, A. ve Kurban, R., 2006. Parçacık Sürü Optimizasyonu ile DWT-SVD Tabanlı Resim Damgalama. Uluslar arası Katılımlı Bilgi Güvenliği ve Kriptoloji Konferansı, Ankara, pp.13–14.

Baker, K.R. ve Trietsch, D., 2009. Principles of Sequencing and Scheduling.

Den Besten, M., Stützle, T. ve Dorigo, M., 2000. Ant colony optimization for the total weighted tardiness problem. In International Conference on Parallel Problem Solving from Nature. Springer, pp. 611–620.

Bonabeau, E., Dorigo, M. ve Theraulaz, G., 1999. Swarm intelligence: from natural to artificial systems, Oxford university press.

Cakar, T., ve Koker, R. (2015). Solving single machine total weighted tardiness problem with unequal release date using neurohybrid particle swarm optimization approach. Computational intelligence and neuroscience, 2015, 70.

Clerc, M., 2006. Particle Swarm Optimization.

Crauwels, H.A.J., Potts, C.N. ve Van Wassenhove, L.N., 1998. Local search heuristics for the single machine total weighted tardiness scheduling problem. INFORMS Journal on computing, 10(3), pp.341–350.

Çakar, T., 2011. Single machine scheduling with unequal release date using neuro-dominance rule. Journal of Intelligent Manufacturing, 22(4), pp.481–490.

Der, O., Vural, R.A. ve Yıldırım, T., 2008. Parçacık Sürü Optimizasyonu Tabanlı Evirici Tasarımı (Inverter Design Based on Particle Swarm Optimization). Yıldız Teknik Üniversitesi Elektronik ve Haberleşme Müh. Bölümü.

Eberhart, R. ve Kennedy, J., 1995. A new optimizer using particle swarm theory. Proceedings of the Sixth International Symposium on Micro Machine and Human Science, pp.39–43.

Eren, T., 2009. Minimizing the total weighted completion time on a single machine scheduling with release dates and a learning effect. Applied Mathematics and Computation, 208(2), pp.355–358.

Esmin, A.A.A. ve Lambert-Torres, G., 2006. Fitting fuzzy membership functions using hybrid particle swarm optimization. In Fuzzy Systems, 2006 IEEE International Conference on. IEEE, pp. 2112–2119.

Fry, T.D. v.d., 1989. A heuristic solution procedure to minimize T on a single machine. Journal of the Operational Research Society, 40(3), pp.293–297.

Hassan, R. v.d., 2005. A comparison of particle swarm optimization and the genetic algorithm. In 46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference. p. 1897.

He, S. v.d., 2004. An improved particle swarm optimization for optimal power flow. In Power System Technology, 2004. PowerCon 2004. 2004 International Conference on. IEEE, pp. 1633–1637.

Helwig, S., Neumann, F. ve Wanka, R., 2009. Particle swarm optimization with velocity adaptation. In Adaptive and Intelligent Systems, 2009. ICAIS'09. International Conference on. IEEE, pp. 146–151.

Hu, X., Shi, Y. ve Eberhart, R., 2004. Recent advances in particle swarm. In Evolutionary Computation, 2004. CEC2004. Congress on. IEEE, pp. 90–97.

J.Kennedy ve Eberhart, R.C., 1995. Particle Swarm Optimization,

Juang, C.-F., 2004. A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 34(2), pp.997–1006.

Juang, C.-F. ve Hsu, C.-H., 2005. Temperature control by chip-implemented adaptive recurrent fuzzy controller designed by evolutionary algorithm. IEEE Transactions on Circuits and Systems I: Regular Papers, 52(11), pp.2376–2384.

Karakuzu, C., 2007. Parçacık sürü optimizasyonu İle bulanık-nöral kontrolör eğitimi ve benzetim örnekleri.

Kennedy, J. ve Eberhart, R., 1995. Particle Swarm Optimization. Neural Networks, 4(1), p.1942.

Kooli, A. ve Serairi, M., 2014. A mixed integer programming approach for the single machine problem with unequal release dates. Computers ve Operations Research, 51, pp.323–330.

Laguna, M., Barnes, J.W. ve Glover, F.W., 1991. Tabu search methods for a single machine scheduling problem. Journal of Intelligent Manufacturing, 2(2), pp.63–73.

Li-Ping, Z., Huan-Jun, Y. ve Shang-Xu, H., 2005. Optimal choice of parameters for particle swarm optimization. Journal of Zhejiang University-Science A, 6(6), pp.528–534.

Mahnam, M. ve Moslehi, G., 2009. A branch-and-bound algorithm for minimizing the sum of maximum earliness and tardiness with unequal release times. Engineering Optimization, 41(6), pp.521–536.

Matsuo, H., Suh, C.J. ve Sullivan, R.S., 1989. A controlled search simulated annealing method for the single machine weighted tardiness problem. Annals of Operations Research, 21(1), pp.85–108.

Merkle, D. ve Middendorf, M., 2000. An ant algorithm with a new pheromone evaluation rule for total tardiness problems. In Workshops on Real-World Applications of Evolutionary Computation. Springer, pp. 290–299.

Pan, Q.-K., Tasgetiren, M.F. ve Liang, Y.-C., 2006. A discrete particle swarm optimization algorithm for single machine total earliness and tardiness problem with a common due date. In Evolutionary Computation, 2006. CEC 2006. IEEE Congress on. IEEE, pp. 3281–3288.

Panneerselvam, R., 2006. Simple heuristic to minimize total tardiness in a single machine scheduling problem. The International Journal of Advanced Manufacturing Technology, 30(7–8), pp.722–726.

Pinedo, M., 2005. Planning and scheduling in manufacturing and services, Springer.

Robinson, J. ve Rahmat-Samii, Y., 2004. Particle swarm optimization in electromagnetics. IEEE transactions on antennas and propagation, 52(2), pp.397–407.

Sen, T., Sulek, J.M. ve Dileepan, P., 2003. Static scheduling research to minimize weighted and unweighted tardiness: a state-of-the-art survey. International Journal of Production Economics, 83(1), pp.1–12.

Shi, Y. ve Eberhart, R., 1998. A modified particle swarm optimizer. In Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on. IEEE, pp. 69–73.

Shi, Y. ve Eberhart, R.C., 1999. Empirical study of particle swarm optimization. In Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on. IEEE, pp. 1945–1950.

Shi, Y. ve Eberhart, R.C., 1998. Parameter selection in particle swarm optimization. In International Conference on Evolutionary Programming. Springer, pp. 591–600.

Tamer, S. ve Karakuzu, C., 2006. Parçacık Sürüsü Optimizasyon Algoritması ve Benzetim Örnekleri, ELECO 2006 Elektrik-Elektronik-Bilgisayar Sempozyumu. Elektronik Bildirileri Kitabı,(302-306), Bursa, Türkiye.

Tasgetiren, M.F. v.d., 2004. Particle swarm optimization algorithm for single machine total weighted tardiness problem. In Evolutionary Computation, 2004. CEC2004. Congress on. IEEE, pp. 1412–1419.

Tchomté, S.K. ve Gourgand, M., 2009. Particle swarm optimization: A study of particle displacement for solving continuous and combinatorial optimization problems. International Journal of Production Economics, 121(1), pp.57–67.

Del Valle, Y. v.d., 2008. Particle swarm optimization: basic concepts, variants and applications in power systems. IEEE Transactions on evolutionary computation, 12(2), pp.171–195.

Wu, C.-C., Hsu, P.-H. ve Lai, K., 2011. Simulated-annealing heuristics for the single-machine scheduling problem with learning and unequal job release times. Journal of Manufacturing Systems, 30(1), pp.54–62.

Xie, X.-F., Zhang, W.-J. ve Yang, Z.-L., 2002. Dissipative particle swarm optimization. In Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on. IEEE, pp. 1456–1461.

Yin, Y. v.d., 2012. An investigation on a two-agent single-machine scheduling problem with unequal release dates. Computers ve Operations Research, 39(12), pp.3062–3073.

Yüksel, E. ve Ülker, M., 2008. Malzeme ve geometrik özellikler bakımdan lineer olmayan çok katlı çelik uzay çerçevelerin optimizasyonu. Gazi Üniversitesi Mühendislik-Mimarlık Fakültesi Dergisi, 23(2).

Zhao, F. v.d., 2005. Application of an improved particle swarm optimization algorithm for neural network training. In Neural Networks and Brain, 2005. ICNNveB'05. International Conference on. IEEE, pp. 1693–1698.

ÖZGEÇMİŞ

1983 yılında Edirne'de doğdu. İlk, orta ve lise eğitimini Edirne'de tamamladı. 1999 yılında Edirne 1. Murat Lisesi'nden mezun oldu. 2004 yılında Sakarya Üniversitesi Endüstri Mühendisliği Bölümü'nden mezun oldu. 2010 yılına kadar özel sektörde orta düzey yönetici olarak çalıştı. 2010 yılında Namık Kemal Üniversitesinde Öğretim Görevlisi olarak çalışmaya başladı, aynı yıl Trakya Üniversitesi İktisat A.B.D.' da Yüksek Lisansını tamamladı. 2012 yılında Sakarya Üniversitesi Endüstri Mühendisliği A.B.D.' da ikinci Yüksek Lisansına başladı. 2014 yılından itibaren Trakya Üniversitesi Proje Koordinasyon Uygulama ve Araştırma Merkezinde Uzman olarak çalışma hayatına devam etmektedir. Halen Uygulamalı Girişimcilik Eğitimleri vermekte olup, Kamu Kurumları ve Özel Sektör Kuruluşlarına danışmanlık hizmeti sunmaktadır.