Elektronik Dizgi İşlemlerinin Eniyilenmesi Ve Değişken Maliyetli Seyyar Satıcı Problemi

Proje No: 108M198

Doç. Dr. Ekrem DUMAN
Yrd. Doç. Dr. Ali Fuat ALKAYA

OCAK 2011
İSTANBUL

Önsöz.

108M198 kodlu TÜBİTAK 1001 projemiz Doç. Dr. Ekrem Duman yürütücülüğünde 15 Kasım 2008 – 15 Kasım 2010 tarihleri arasında yürütülmüş ve öngörüldüğü gibi 24 ay içinde tamamlanmıştır. Projede Yrd. Doç. Dr. Ali Fuat Alkaya araştırmacı olarak, yüksek lisans öğrencileri Burcu Müzeyyen Kıyıcığı ve Hüseyin Demirkale ise bursiyer olarak görev almışlardır.

Proje raporunun içerisinde detaylıca açıklandığı üzere proje kapsamında düşünülen tüm çalışmalar başarıyla tamamlanmış ve çok başarılı sonuçlar alınmıştır. İki yüksek lisans tezi ve bir doktora tezi bitirilmiş ve toplam dört konferans bildirisi ve bir SCI indeksli dergi makalesi yayımlanmıştır. İki diğer makale ise SCI indeksli dergilerde değerlendirme aşamasındadır.

Projenin gerçekleştirilmesini verdiği destekle mümkün kılan TÜBİTAK'a ve çalışanlarına, değerli yorumlarıyla bize rehberlik eden anonim proje izleyicimize teşekkürlerimizi bir borç biliriz.

İçindekiler

Tablolar Listesi

Şekiller Listesi

Özet

Bu proje kapsamında baskılı devre kartları üzerine elektronik komponentlerin dizgisi için kullanılan iki tip dizgi makinesinin kullanımında ortaya çıkan en iyileme problemleri ele alınmıştır. Bu makine tipleri son yıllarda yaygınlaşan çip saçıcı makineler ve ona benzeyen ancak daha eski teknoloji olan çip parça yerleştirici makinelerdir. Her iki makine için de parçaların dizgi sırasının eniyilenmesi ve parça tiplerinin besleyici üzerinde nereye yerleştirilmesi gerektiği (besleyici düzeni) problemleri vardır. Bu iki problem birbirinden bağımsız düşünüldüğünde, dizgi sırası problemi, yazında daha önce hiç adı geçmemiş ve bizim Sıraya Dayalı Seyyar Satıcı Problemi (SDSSP) adını verdiğimiz bir SSP genellemesi olarak ortaya çıkmaktadır. Besleyici düzeni problemi ise çip parça yerleştiriciler için çok basittir ancak, çip saçıcılar için Karesel Atama Problemine (KAP) dönüşmektedir. Her iki problemin bir arada ele alınması halinde daha da karmaşık bir problem karşımıza çıkmaktadır.

Proje süresince ilk olarak SDSSP'nin matematiksel modeli geliştirilmiş ve çözümü için basit ancak etkin sezgisel yöntemler geliştirilmiştir. Sonraki aşamalarda KAP'nin çözümü için yazında sıklıkla kullanılan dört farklı metasezgisel yöntem kodlanmış bunların karşılaştırılması sonrasında benzetimli tavlama en iyi KAP çözücü olarak benimsenmiştir. Çip saçıcıların birleşik probleminin çözümü için ise tekrarlamalı yöntem ve problemin tamamının bir metasezgiselle çözülmesi seçenekleri yarıştırılmış ve SDSSP'nin getirdiği özel karmaşıklığın etkisiyle, etkin SDSSP çözümlerini bünyesinde bulunduran tekrarlamalı yöntem daha iyi sonuç vermiştir. Ayrıca, SDSSP ve birleşik problem için CPLEX kullanılarak kesin çözümler de aranmış ancak, küçük problemler dışında çözüm bulunamamıştır.

Projeyle ilgili dört konferans bildirisi yayımlanmış ve üç makale de SCI dergilerde yayımlanmak üzere sunulmuştur. Ocak 2011 itibariyle bu makalelerden biri yayımlanmış diğer ikisi ise değerlendirme sürecindedir.

Anahtar Kelimeler: Seyyar satıcı problemi, sıraya dayalı gezgin satıcı problemi, karesel atama problemi, dizgi optimizasyonu, metasezgiseller

Abstract

Within the scope of this project the optimization problems related with the use of two different placement machines used for the assembly of PCBs are analyzed. These two types of machines are the chip shooters which became popular recently and the chip mounters which have an older technology but are similar to chip shooters. For both machine types there are two inherent problems: determining the placement sequence and determining the places of component types on feeders (feeder configuration). When we approach these problems independently, the placement sequencing problem turns out to be a new variant of the Traveling Salesman Problem (TSP), the Seqeunce Dependent TSP (SDTSP) as we named here. The feeder configuration problem is trivial for the chip mounters but it turns out to be the Quadratic Assignment Problem (QAP) for the chip shooters. When we tackle both problems simultaneously, we face with a more complicated problem.

Throughout the project, we first built the mathematical model of the SDTSP and developed a few simple but effective heuristics to solve it. In the later phases, for the solution of the QAP, we compared the performances of four different metaheuristics and identified the simulated annealing approach as the best among them. For the solution of the combined problem of chip sooters, we compared iterative approach with solving the overall problem by a metaheuristic. Because of the complexity of the sequencing problem, the iterative approach which includes the efficient SDTSP solvers turned out to be better. Also, for both the SDTSP and the combined solution we tried finding the exact solutions with CPLEX but except the small problems we were not successful.

Related with the project, we have published four conference proceedings and submitted three papers to SCI journals. As of January 2011 one of these papers is published and the other two are under review.

Keywords: Traveling salesman problem, sequence dependent traveling salesman problem, quadratic assignment problem, placement optimization, metaheuristics

1. Giriş

Baskılı devre kartlarının dizgi işlemi son 20 senedir dünya ve ülke endüstrisinde çok büyük bütçelerin kullanıldığı bir sektör haline gelmiştir. Çünkü, baskılı devre kartları çoğu elektronik ürün içerisinde önemli parçalar olarak yer almaktadır. Örnek vermek gerekirse bilgisayarlar, hesap makineleri, robotlar, uzaktan kumandalar, cep telefonları ve daha bir çok ürün kategorisi sayılabilir. Bu yüzden baskılı devre kartı dizgi sektörü büyümeye devam edeceği rahatlıkla öngörülebilen ender sektörlerdendir. 2006 yılı itibariyle, dünya üzerinde baskılı devre kartı satış miktarı 973 milyar Amerikan doları olarak hesaplanmış ve 2011 için 1.3 trilyon Amerikan doları olacağı tahmin edilmektedir (Drysdale, 2008).

Baskılı devre kartı değişik tür, şekil ve boyutlarda yüzlerce elektronik parçadan oluştuğu için dizgisi oldukça karmaşık bir iştir. Bu yüzden yıllar içinde baskılı devre kartlarının dizgisinde insan emeğinden ziyade otomatik dizgi makinelerin ağırlıkla kullanıldığı gözlenmektedir. Baskılı devre kartlarının dizgisini yapan bu makineler çok sayıda değişik parçayı, karmaşık araçları kullanarak dahi çok büyük bir hızla ve doğrulukla yapabilmektedir. Ayrıca, geçmiş yıllarda kullanılan Delik Açma (Pin-Through) teknolojisi yerini daha hızlı olan Yüzey Yapıştırma Teknolojisi'ne (YYT) (Surface Mount Technology) büyük ölçüde bırakmıştır (Jeevan et al., 2002). Bu teknoloji, bir kartı eskisine nazaran çok yüksek hızlarda ve hassaslıkta monte edebilir. Örnek vermek gerekirse dakikada 600 parçayı basabilen ve onbinde bir hatayla çalışan makineler sektörde kullanılmaktadır. Bu nedenle YYT çoğu üreticinin tercihi haline gelmiştir. YYT makinelerinin alım maliyeti bütün donanımlarıyla beraber 250.000 ila 1.000.000 Amerikan Doları arasında değişmektedir (Csaszar ve diğ., 2000; Tirpak ve diğ., 2000). Bu makinelerin maliyetlerinin yüksek olmasından dolayı, etkin ve verimli kullanılması sektörde bulunan her kurumun en birincil hedefidir.

Mekanik olarak parça dizgi makineleri kontrol ünitesinin yanında üç ana kısımdan oluşur: Yerleştirme kafası ya da kafaları, taşıyıcı tabla ve besleyici düzeneği. Bu ana kısımların nasıl çalışacağı ve monte edilecek parçaların özellikleri (büyüklük ve ağırlık) dizgi makinesinin hızını belirlemede önemli rol oynamaktadır. Özel ortam gereksinimleri ve makine kabiliyetleri analiz edilmesi ve çözülmesi gereken farklı problemler doğurur. Fakat, bu makinelerde genel olarak gözlemlenen problemler aşağıdaki gibi özetlenebilir (Duman, 1998).

1- Parça tiplerinin makinelere bölüştürülmesi
2- BDK üretim sırasının belirlenmesi
3- Parça tiplerinin besleme hücrelerine bölüştürülmesi (Besleyici düzeni)
4- Parça dizgi sırasının belirlenmesi

Yukarıdaki problemlerin hepsi birbiriyle ilişkilidir ve birinin çözümü diğerlerinin çözümünü etkilemektedir. Bu ilişki ilk iki ve son iki arasında daha açık ve net olarak görülebilmektedir. Eğer eniyilenmiş bir çözüm aranıyorsa, dört problem de aynı anda ele alınmalı ve çözülmelidir. Bu proje çalışmasında, son iki problem hedeflenmiştir. Problemlerin şekli makine mimarileriyle yakından ilgilidir. Dizgi sırasının belirlenmesi problemi tipik olarak Seyyar Satıcı problemi ya da genellemeleri olarak tanımlanabilir (Duman ve Or, 2004; Duman, 2009). Öte yandan besleyici düzeni problemi önemsiz bir problem olabilir ya da Atama Problemi veya Karesel Atama Problemi olarak tanımlanabilir (Duman ve Or, 2007).

Bu proje çalışmasında bu makine mimarilerinden ikisi incelenmiş ve bunların temelini oluşturan Yöneylem Araştırması problemleri tanımlanarak bunlar için iyi sonuç üreten çözümler üretilmiştir. Bunlar "çip parça yerleştirici" ve "çip saçıcı" makineleridir. Her iki makinede üzerinde kafaların bulunduğu döner tarete sahiptir. Bir çip parça yerleştirici örneği olan TDK RX-5A 72 kafaya, çip saçıcılar ise genelde 10 ya da 12 kafaya sahiptir. Her ikisinde de baskılı devre kartı x ve y düzlemlerinde hareket edebilen taşıyıcı tabla üzerinde bulunmaktadır. İki makine arasındaki en önemli fark besleyici düzeneği mimarisindedir.

Bu makinelerin mimarisi gereği, birinci problemin Seyyar Satıcı Problemi (SSP) (Traveling Salesman Problem) olduğu rahatlıkla görülebilir. Fakat kafalarda taşınan parçaların ağırlıklarının farklı olması döner taretin hızını etkileyeceği gerçeğiyle, bu problem taşınan parçanın dizgi sırasına göre tur zamanının değiştiği bir SSP genellemesi içermektedir ve biz bunu projemizde Sıraya Dayalı SSP (SDSSP) olarak tanımladık. Bu problem ilerleyen bölümlerde açıkça tanımlanmıştır. Her ne kadar SSP NP-Zor bir problem olup, çözümüne yönelik algoritma ve sezgisel yöntemler çok fazla sayıda yazında ele alınsa da, bildiğimiz kadarıyla burada tanımladığımız SDSSP yazında tanımlanmamış ve üzerinde çalışılmamıştır.

Konu edilen YYT çip saçıcı makinesinin en iyi dizgi zamanını elde etmeye çalışırken yukarıda tanımlanan problemlerin ayrı ayrı eniyilenmesi yeterli değildir. Bu problemlerin beraberce eniyilenmesi gerekmektedir, çünkü problemler birbiriyle ilişkilidir ve birinin çözümü diğerinin çözümünü etkilemektedir. Bu projede, yukarıda tanımları yapılan iki problemin beraberce eniyilenmesi problemini esas problem olarak adlandırdık. SDSSP ve KAP ise yan problemler olarak isimlendirilmiştir. Esas problem de yazında karşılaşılmamış ve dolayısıyla çözümler üretilmemiş yeni bir problemdir.

İlerleyen bölümlerde ilk olarak bahsedilen yan problemlerden Seyyar Satıcı Problemi genellemesinin matematiksel modeli oluşturulmuştur. SDSSP çözümü için, öncelikle birleşimsel eniyileme problemlerinde kullanılan kesin çözüm yöntemlerini (örneğin dal ve kesi) uygulayan ticari programlar kullanılmış ve sonrasında da bu problem için çeşitli sezgiseller geliştirilmiştir. Ayrıca bu problem için yeni yapısal ve iyileştirme algoritmaları geliştirme üzerinde çalışılmıştır. Sonrasında diğer yan problem olan KAP için yazında ele alınmış en iyi yaklaşımlardan bir tanesi programlanmıştır. Bu yan problem için yeni bir çözüm yaklaşımı sunmak bu proje içinde birincil hedeflerimiz arasında değildir. Son olarak, esas problem için sezgisel yaklaşımlar da kullanılarak eniyilemesi için çalışılmıştır. Geliştirilen algoritma ve yöntemlerin uygulanabilirliğini belirlemek için bilgisayısal deneyler yapılmıştır.

2. Genel Bilgiler

Bu bölümde projede ele alınan makinelerin çalışma prensipleri ve problemlerinin tanımlanması yapılacaktır.

2.1.Çip parça yerleştirici

Makinelerden çip parça yerleştirici (chip mounter) çalışma prensiplerinden ortaya çıkan problemlerin tanımlarının yapılabilmesi için çalışma prensiplerinin detaylıca incelenmesi gerekir. Çip parça yerleştiricinin çalışma prensibi şu şekildedir (Duman, 2007).

Şekil 1'de görüldüğü üzere TDK RX-5A'da besleyici düzen hücreleri 'alım bölgesi' (pickup zone) altındadır. Taret dönerken kafalar uygun olan hücrenin üzerine geldiğinde kafa o hücreden parçayı alır ve taret dönmeye devam eder. Kafa dizgi noktasına geldiğinde taşıdığı parçayı uygun açıyla yerleştirir. Taretin bir adımlık dönmesi esnasında, eş zamanlı olarak taşıyıcı düzeneği BDK'daki bir sonraki parça yerleştirilecek noktayı dizgi noktasına getirir ve dizgi lokasyonu üzerinde bulunan kafadaki parça çakılır. Bu işlemleri daha net bir şekilde özetleyecek olursak:

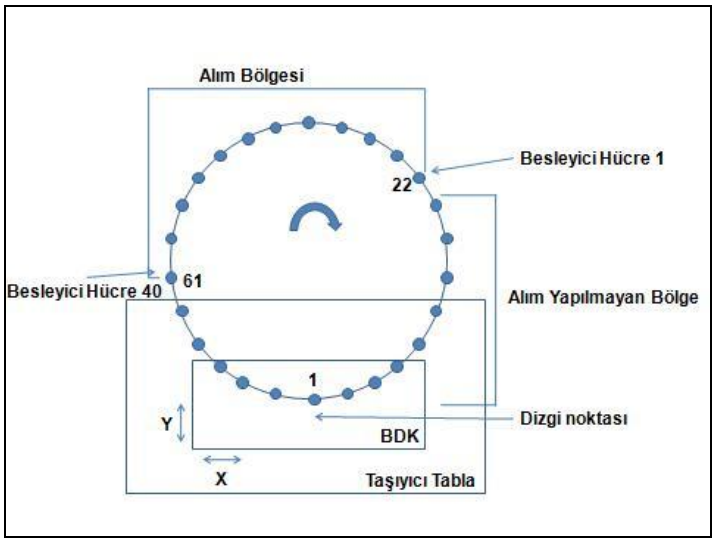
Madde 1:

- Taret döner ve dizgiyi yapacak sıradaki kafa BDK üzerine gelir.
- Taşıyıcı tabla sıradaki yerleştirme noktasını dizgi kafasının altına getirir.
- Gerekiyorsa kafa taşıdığı parçayı yerleştirmek için BDK'yla uygun açıyı oluşturur.

Madde 2:

- Dizgiyi yapacak kafa aşağı hareket eder, dizgiyi yapar ve tekrar yukarı hareket eder.
- Alım bölgesindeki kafalar eğer uygun besleyici hücre üzerinde iseler aşağı hareket eder, parçayı alır ve tekrar yukarı hareket ederler.

Her maddedeki işlemler eşzamanlı olur ve her iki madde birbirini takip ederek dizgi işlemi devam eder. Dolayısıyla bir maddenin süresi o madde içinde yapılan işlerden en uzun süreni kadardır (Chebyshev mesafe ölçümü).



Şekil 1—Çip parça yerleştirici

Bu makine farklı ağırlıklarda parça tiplerini yerleştirebilmektedir. Parça taşıyan kafalardan herhangi biri daha ağır bir parça tipini alırsa taretin dönme hızı yavaşlar. Ele aldığımız

makine için dört farklı hız değeri vardır. Bunlar, hafiften ağıra doğru sıralayacak olursak, beş derecelik dönüş için 0.20, 0.23, 0.33 ve 0.40 saniyedir. Öte yandan taşıyıcı tablanın hızı ise saniyede 120 mm'dir.

Bu makinenin çalışma prensipleri iki problem doğurur. Bunlar; 1. Parça tiplerinin besleme hücrelerine bölüştürülmesi (Besleyici düzeni), 2. Parça dizgi sırasının belirlenmesi problemleridir. Yazında bu makine için parça tiplerinin besleme hücrelerine bölüştürülmesi probleminin ideal olarak çözülebildiği gösterilmiştir (Duman, 2007). Dolayısıyla, bu makine için sadece parça dizgi sırasının belirlenmesi problemi çözülmelidir.

Parça dizgi sırasının belirlenmesi problemini tanımlamak için öncelikle peş peşe yerleştirilen iki parça arasındaki yol masrafını tanımlamamız gerekir. Bunun için ise bazı ön tanımlamalara ihtiyaç vardır. Ele alınan makine Chebyshev mesafe ölçümünü kullandığından iki nokta, *i* ve *j*, arasındaki mesafe *d(i,j)*= max{| i_x - j_x |, | i_y - j_y |} olarak tanımlanır (i_x ve i_y i noktasının *x* ve *y* koordinatlarını belirtmektedir). Yol masrafını zaman cinsinden şu şekilde ifade edebiliriz.

$$CF1(i, j) = \frac{d(i, j)}{v} \tag{1}$$

Buradaki *v* taşıyıcı tablanın hızını ifade etmektedir.

t^0 dizgi zamanını, *tt*'de döner taretin bir adımlık dönme zamanını ifade ederse, o zaman peş peşe yerleştirilen iki parça arasında geçen zaman, t_{ij}, şu şekilde hesaplanabilir.

$$t_{ij} = t^0 + \max\left(CF1(i, j),\ tt\right) \tag{2}$$

Yukarıda belirtilen makine çalışma prensipleri doğrultusunda *tt* sabit değildir ve dört farklı değer alabilmektedir. Zaten eğer *tt* sabit tek bir değere sahip olsaydı o zaman problem klasik SSP'ye dönüşürdü. Bu masraf fonksiyonuyla beraber ortaya çıkan dizgi sırasının belirlenmesi problemini daha açıkça şu şekilde ifade edebiliriz. Makinenin döner tareti dört farklı hız değerinde dönebilmekte ve hangi hız değerinde döneceğini taret üzerinde o anda bulunan en ağır parça tipi belirlemektedir. Dolayısıyla o anda taşıyıcı tabla yerleştirilmesi planlanan *j* parçasının yerleştirilme noktasına hareket ederken, döner taret 1'den 60'a kadar numaralanmış kafalarında taşınan en ağır parça tipine karşılık gelen hız değeriyle bir kademe döner. Yukarıda belirttiğimiz gibi taşıyıcı tabla ve döner taretten hangisi daha çok zamanda işlemini tamamlıyorsa *j* parçasının yerleştirilme süresi o kadar olur. Fakat taretin dönme hızı sonradan yerleştirilecek belli sayıda parçanın seçilmesine bağlıdır. Yani, bu problemde bir noktadan diğerine gitme masrafı oluşturulan gezinme sırasında sonradan gelecek noktaların hangileri olduğuna göre değişiklik göstermektedir. Bu da, problem tanımlanırken klasik SSP formülasyonun geçersiz olduğu anlamına gelir ve yeni bir SSP genellemesine dönüşür.

Problemin matematiksel tanımını ise aşağıdaki gerekli tanımları yaptıktan sonra yapabiliriz.

N: yerleştirilecek toplam parça sayısı,

n: parça tipi sayısı,

K: ağırlık kategori sayısı, (bizim makinemizde *K*=4)

N_k: her bir ağırlık kategorisindeki parça sayısı, *k=1,2,…,K*.

n_k: her bir ağırlık kategorisindeki parça tipi sayısı, *k=1,2,…,K*.

$$\sum_{k=1}^{K} n_k = n$$

R: besleyici düzenekteki hücre sayısı (*R*=60, fakat ilk 20 hücreye hiç bir parça tipi konmaz çünkü orası alım yapılmayan bölgedir.)

H: döner taretteki kafa sayısı (bizim makinemizde *H*=72)

npz: alım yapılmayan bölgedeki kafa sayısı (bizim makinemizde *npz*=20)

ct_{it}: her parçanın parça tipini ifade eden parça tipi matrisi (*N x n*)

$$ct_{it} = \begin{cases} 1 & \text{eger i parcasinin tipi t ise} \\ 0 & \text{aksi takdirde} \end{cases}$$

g_{tk}: her parça tipinin grubunu ifade eden grup matrisi (*n x K*)

$$g_{tk} = \begin{cases} 1 & \text{eger t parca tipi k grubundaysa} \\ 0 & \text{aksi takdirde} \end{cases}$$

tt_k : her bir ağırlık kategorisi k için döner taretin bir adım dönme zamanı

p: yerleştirme sırası (pozisyonu).

$$x_{ip} = \begin{cases} 1, & \text{eger i noktasina p ninci sirada ugranirsa} \\ 0, & \text{aksi takdirde} \end{cases}$$

$$w_{ijp} = \begin{cases} 1, & \text{eger j noktasina p sirasinda ve i noktasina p - 1 sirasinda ugranirsa} \\ 0, & \text{aksi takdirde} \end{cases}$$

Öte yandan besleyici düzenini ifade etmek için *t* parça tipini *r* hücresine atanma durumunu gösteren y_{tr} karar değişkenini tanımlamamız gerekir. Besleyici düzenin ideal bir şekilde çözülmüş olduğunu ve y_{tr} değerlerinin önceden bilindiğini varsayıyoruz.

$$y_{tr} = \begin{cases} 1, & \text{eger t parca tipi r hucresinde saklanirsa} \\ 0, & \text{aksi takdirde} \end{cases}$$

j parçasının *p*'inci sırada yerleştirildiği varsayımıyla *i* parçasından *j* parçasına gidiş masrafını şu şekilde tanımlayabiliriz.

$$C_{ijp} = w_{ijp} maks\left(CF1(i,j), \underset{m=0}{\overset{R-1}{maks}}\left(\sum_{l=1}^{N}\sum_{t=1}^{n}\sum_{k=1}^{K}\sum_{u=m+1}^{R} tt_k g_{tk} ct_{lt} x_{l,p+m} y_{tu} \right)\right) \qquad (3)$$

Bu tanımda dış *maks* fonksiyonundaki ilk terim taşıyıcı tablanın BDK'yı yerleştirme noktası altına getirmesini hesaplar. İkinci terim ise kafalardan herhangi birinde o anda taşınan en ağır parçanın taret zamanını hesaplar.

C_{ijp}'deki doğrusal olmama durumunu D_{ijp} tanımıyla ortadan kaldırabiliriz.

$$\min \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} \sum_{p=1}^{N} D_{ijp} \tag{4}$$

s.t.

$$D_{ijp} - w_{ijp} CF1(i,j) \geq 0 \qquad i,j,p = 1,\ldots,N \tag{5}$$

$$D_{ijp} - w_{ijp} \sum_{l=1}^{N} \sum_{t=1}^{n} \sum_{k=1}^{K} \sum_{u=1}^{R} tt_k g_{tk} ct_{lt} x_{lp} y_{tu} \geq 0 \qquad i,j,p = 1,\ldots,N \tag{6}$$

$$D_{ijp} - w_{ijp} \sum_{l=1}^{N} \sum_{t=1}^{n} \sum_{k=1}^{K} \sum_{u=2}^{R} tt_k g_{tk} ct_{lt} x_{l,p+1} y_{tu} \geq 0 \qquad i,j,p = 1,\ldots,N \tag{7}$$

$$\vdots$$

$$D_{ijp} - w_{ijp} \sum_{l=1}^{N} \sum_{t=1}^{n} \sum_{k=1}^{K} \sum_{u=R}^{R} tt_k g_{tk} ct_{lt} x_{l,p+R-1} y_{tu} \geq 0 \qquad i,j,p = 1,\ldots,N \tag{8}$$

$$\sum_{p=1}^{N} x_{ip} = 1, \qquad i = 1,\ldots,N, \tag{9}$$

$$\sum_{i=1}^{N} x_{ip} = 1, \qquad p = 1,\ldots,N, \tag{10}$$

$$\sum_{i=1}^{N} w_{ijp} = x_{jp}, \qquad j,p = 1,\ldots,N, \tag{11}$$

$$\sum_{j=1}^{N} w_{ijp} = x_{i,p-1}, \qquad i,p = 1,\ldots,N, \tag{12}$$

$$x_{ip} \in \{0,1\}, \qquad i,p = 1,\ldots,N, \tag{13}$$

$$w_{ijp} \in \{0,1\}, \qquad i,j,p = 1,\ldots,N, \tag{14}$$

Kısıt 5 *j* parçasının yerleştirilme zamanının *i* parçasından *j* parçasına gidişten daha kısa olmamasını gerektirir. Kısıtlar 6-8 (*R* tane farklı kısıtı ifade ederek) o anda kafalarda taşınarak gelen parçaların taret zamanlarından daha küçük olamayacağını ifade eder. 9 ve 10 numaralı kısıtlar noktalar ve zaman aralıkları arasında birebir eşleşme yapar. Böylece altturlar da elenmiş olur. 11 ve 12 numaralı kısıtlar her noktadan gelen ve her noktadan çıkan geçişleri eşitler.

Yukarıda tanımladığımız SSP genellemesi yazında tanımlanmamıştır ve biz bu probleme Sıraya Dayalı SSP (SDSSP) (Sequence Dependent TSP, SDTSP) adını vermiş bulunmaktayız.

Duman, çalışmasında parça dizgi sırasının belirlenmesi probleminin formülasyon zorluğundan bahsetmiş ve çözümü için ATMA adını verdiği bir algoritma öne sürmüştür (Duman, 2007). Bu algoritmada öncelikle parçalar ağırlıklarına göre gruplandırılmaktadır (dört grup). Sonrasında her bir gruba ait parçalar için birer rota oluşturulmaktadır. En hafif gruptan başlayarak ağırlık sırasıyla daha ağır olan grubun rotası daha hafif olan rotanın bittiği yerdeki parçaya en yakın parçası başlangıç noktası olacak şekilde rotalar birbirine bağlanır. Bu şekilde parça dizgi sırası elde edilir ve sonrasında bu dizgi sırası kullanılarak yine aynı çalışmada öne sürülen prosedürle besleyici düzeni bulunur (Duman, 2007). Alkaya ve diğerleri grupların yerleştirme sırasını en ağırdan en hafife doğru yapmış ve belli BDK örnekleri için ATMA'dan daha iyi sonuç elde etmiştir (Alkaya ve diğ., 2008). Geliştirdikleri yönteme İATMA adını vermişlerdir.

Örnek verecek olursak varsayalım ki elimizde 100 parçalı bir BDK olsun. Her bir gruptaki parça sayısı N1=80, N2=10, N3=5 ve N4=5 olsun (küçük indisler daha hafif olan grupları göstermektedir). Yine her bir grup için parça tipi sayıları n1=20, n2=10, n3=5 ve n4=5 olsun. Bu örnek için ATMA algoritması uygulandıktan sonra görülür ki döner taret 50 defa 0.20 s hızıyla, 15 defa 0.23 s hızıyla, 10 defa 0.33 s hızıyla ve 25 defa 0.20 s hızıyla döner. Bu hesaplamaların detayları (Duman, 2007)'de mevcuttur.

2.2.Çip saçıcı

Çip saçıcıların çalışma prensipleri ise şu şekildedir. Şekil 2'de görüldüğü üzere besleyici düzeneği doğrusal bir yapıya sahiptir ve sadece x doğrultusunda hareketlidir. Döner taret yine saat yönünde dönmekte iken her adımda kafalardan sadece bir tanesi besleyici düzeneği üzerine gelir ve uygun parçayı alır. Ancak bu, kafanın alacağı parçanın besleyici düzeneğinin hareketleriyle tam kafanın altına getirilmesiyle sağlanır. Şekilde verilen çip saçıcı için alınan her parçanın beş adım sonra monte edileceği rahatlıkla görülebilir. Bu parça son adımda BDK üzerine doğru hareket ederken eş zamanlı olarak taşıyıcı tabla da parçanın monte edileceği BDK'daki noktayı yerleştirme pozisyonuna hizalar ve daha sonra parçayı monte eder. Bu işlemleri de maddeler halinde netleştirecek olursak:

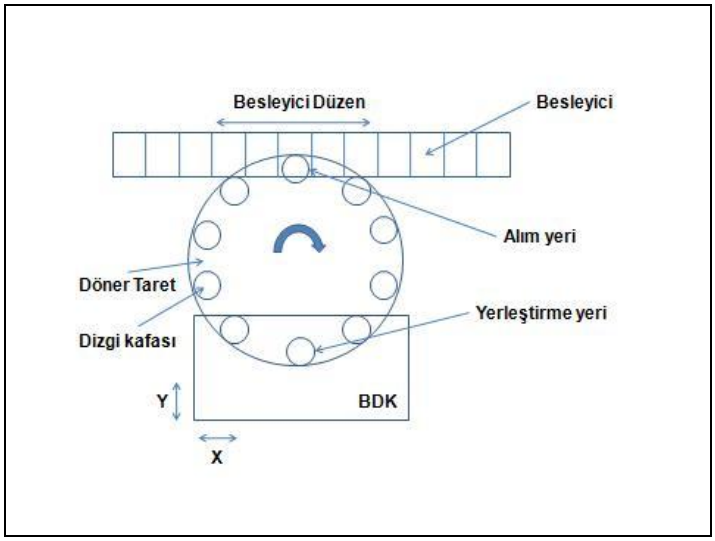
Madde 1:

- Taret döner ve dizgiyi yapacak sıradaki kafa BDK üzerine gelir.
- Taşıyıcı tabla bir sonraki yerleştirme noktasını dizgi kafasının altına getirir.
- Gerekiyorsa kafa taşıdığı parçayı yerleştirmek için BDK'yla uygun açıyı oluşturur.
- Besleyici düzeneği sıradaki alınacak parçayı alım noktasının altına getirir.

Madde 2:

- Dizgiyi yapacak kafa aşağı hareket eder, dizgiyi yapar ve tekrar yukarı hareket eder.

Alım noktasındaki kafa aşağı hareket eder, parçayı alır ve tekrar yukarı hareket eder.



Şekil 2—Çip Saçıcı

Burada da her maddedeki işlemler eşzamanlı olur ve her iki madde de birbirini takip ederek dizgi işlemi devam eder. Dolayısıyla bir maddenin süresi o madde içinde yapılan işlerden en uzun süreni kadardır (Chebyshev mesafe ölçümü).

Bu makine türünde ise üç ayrı problem çözülmeyi beklemektedir. Bunlar besleyici düzeni, parça dizgi sırasının belirlenmesi ve her ikisinin beraberce eniyilemesi.

Çip saçıcı makinesinin döner tareti de taşınan parçanın ağırlığına göre farklı hızlarda hareket eder. Dolayısıyla, parça dizgi sırasının belirlenmesi problemi, yukarıda bahsedilen çip parça yerleştirici makinesinde ortaya çıkan SDSSP problemine dönüşmektedir.

Öte yandan besleyici düzeni problemi tam olarak yazındaki karesel atama problemine dönüşür (Duman ve Or, 2007). Karesel Atama Problemi (KAP) en basit tanımıyla n tane ofisi n tane çalışana şu amaç, bilgi ve kısıtlar altında atamaktır: Bize verilen bir uzaklık matrisi her ofisin birbirine olan uzaklığını içerir (c_{ij}) ve bir ilişki matrisi her çalışanın bir diğeriyle günlük görüşme sayısını içerir (d_{kl}). Her çalışanın sadece bir ofisi olacak şekilde çalışanlarca hergün alınan toplam mesafeyi en küçüklemek bu problemin amacıdır. KAP'nin matematiksel tanımını aşağıdaki şekilde yapabiliriz.

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} c_{ij} d_{kl} x_{ik} x_{jl} \tag{27}$$

öyle ki

$$\sum_{i=1}^{n} x_{ik} = 1, \qquad k = 1,\ldots,n \tag{28}$$

$$\sum_{k=1}^{n} x_{ik} = 1, \qquad i = 1,\ldots,n \tag{29}$$

$$x_{ik} \in \{0,1\}, \qquad i,k = 1,\ldots,n \tag{30}$$

Bu formulasyon çalışan *i*'nin ofis *k*'ya atanması durumunu x_{ik} değişkeniyle ifade etmektedir.

Proje önerisinde belirtildiği gibi KAP'nin çözümü için yazında sunulan yöntemlerden en iyisi seçilerek uygulanmıştır.

Çip saçıcı için bahsi geçen **üçüncü** problem SDSSP ve KAP'nin beraberce en iyilenmesidir.

Çip saçıcılarla ilgili oldukça detaylı bir yazın taraması yapılmış ve PICMET09 konferansında sunulmuştur (Alkaya ve Duman, 2009). Bu makale ekte sunulmuştur. Özetle ifade edecek olursak, bu yazın taramasında ortaya çıkan temel sonuçlar; çoğu çalışmanın parça dizgi sırasının belirlenmesi ve besleyici düzeni üzerinde çalıştığı, iki problemin beraber çözümünde genetik algoritmaların ya da tekrarlamalı çözüm yöntemlerinin kullanıldığı fakat diğer meta sezgisellerin hiç kullanılmadığı, çok az çalışmanın çip saçıcılardaki farklı dönme hızlarını göz önüne alarak iyileştirme yaptığı ve hiçbir çalışmanın farklı dönme hızlarını göz önüne alarak problem formülasyonunu yapmadığıdır.

2.3. Yazın Taraması

Projemizde kesin çözüm yöntemleri ve tanımlanan problemlere benzeyen yazında önceden tanımlanmış problemler için bir yazın taraması yapılması ve bunlar için geliştirilmiş yöntemlerin incelenmesi amaçlanmıştır. Aşağıda bu çalışmaların özetleri alt bölümler halinde sunulmuştur.

2.3.1. Seyyar Satıcı Problemi (SSP) ve Kesin Çözüm Yöntemleri:

Seyyar satıcı problemleri literatürde en iyi bilenen NP-Zor birleşimsel eniyileme problemlerinden birisidir. SSP literatürde çok geniş bir yere sahiptir ve küçük bir araştırma ile birlikte problemin 1940 yılından başlayarak çok sayıda araştırmalara konu olduğu görülebilir. Günümüzde ise SSP ve onun farklı şekilleri üzerine yapılan çalışmalar birçok gerçek problemi çözmede kullanılmaktadır.

SSP'yi tanımlayacak olursak; elimizde bulunan tam grafta, $G=(V,E)$, V noktaların kümesiyle, E kenarların kümesiyle ve son olaraktan, maliyet değeri olan c_{ij} her bir kenar ile ilişkilendirilmiştir. c_{ij} 'nin aldığı değer $i \in V$ köşesinden $j \in V$ köşesine alınan yolu gösterir. V' nin eleman sayısı N ile gösterilmiştir. $|V|=N$. Verilen bu bilgilerle, SSP'nin çözümü bize en kısa Hamiltonian döngüsünü, diğer bir ifadeyle, her bir noktaya sadece bir kez uğrayarak oluşan döngüyü vermek zorundadır.

Problemin matematiksel formülasyonu, hangi kenarlarin turumuza dahil edileceğini belirlemek üzere karar değişkenleri kullanılarak bulunabilir. x_{ij} değişkenine 0-1 değerlerini atamak istersek;

$$x_{ij} = \begin{cases} 1 & \text{eger i ve j arasindaki kenar kullanilmissa} \\ 0 & \text{aksi takdirde} \end{cases}$$

Olarak gösterilebilir.

Sonrasında SSP formülasyonu, i köşesi ile j köşesi arasındaki maliyeti temsil eden c_{ij} değişkeni kullanılarak yazılır.

$$\min \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} c_{ij} x_{ij} \tag{31}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \qquad j = 1,\ldots,n; i \neq j \tag{32}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad i = 1,\ldots,n; i \neq j \tag{33}$$

$$u_i - u_j + n x_{ij} \leq n-1 \qquad i, j = 2,3,\ldots,n; i \neq j \tag{34}$$

$$x_{ij} \in \{0,1\}, \qquad i = 1,\ldots,n, \qquad j = 1,\ldots,n, \tag{35}$$

$$u_i \in Z^+ \qquad i = 1,\ldots,n \tag{36}$$

Yukarıdaki formülasyonda, 32. kısıt satıcının her şehre sadece bir kez girdiğini, 33. kısıt ise her şehirden sadece bir kez çıktığını göstermektedir. Alt turları engellemek ise 34. kısıtta gösterilmiştir (Winston ve Venkataramanan, 2003).

Yukarıda bahsedildiği gibi, SSP her örneği için kolaylıkla çözülebilen bir problem değildir. Bunun sebebi ise SSP ve onun farklı şekilleri olan problemleri çözerken kullandığımız programların sayısal karmaşıklığının problem boyutu arttıkça çok hızlı bir şekilde büyümesidir. Bu problem kategorisi NP-Zor problemler olarak bilinmektedir. (Gary ve Johnson, 1979).

Klasik SSP' de, iki şehir arası maliyet sabittir ve önceden bilinir. Böylece iki nokta arasındaki maliyet fonksiyonu uzaklık ile orantılı artan doğrusal bir fonksiyondur. Uzaklık ölçüsü Öklit veya Chebyshev metrik olabilir. Chebyshev metrik daha çok baskı devre üretim makinelerinde ortaya çıkmaktadır.

SSP' yi çözmek için birçok çözüm yöntemleri geliştirilmiştir. Bunlar içinde ilk olarak kesin çözüm yöntemleri açıklanmalıdır. Kesin çözüm yöntemleri, en iyi çözümü bulmayı garanti eder ve bulunan çözümün eniyiliğini birleşimsel eniyileme problem sınıfının bütün örnekleri için ispatlar. Ancak, koşma zamanı problemin boyutu ile birlikte çarpıcı biçimde artmaktadır. Genelde küçük ve orta boyutlu problemler pratik ve ispatlanabilir bir biçimde çözülebilmektedir. Bütün birleşimsel eniyileme problemlerini başarıyla çözen kesin çözüm yöntemi yoktur. Fakat SSP problemleri ile ilgili kesin çözüm yöntemleri geliştirilmiştir. Geliştirilen bu yöntemleri, dal-sınır, kesen düzlem ve dal-kesi olarak kategorize edilmiştir.

Dal-sınır algoritması en iyi sonucu bulabilmek için bütün çözüm kümesini aramaktadır. Fakat hızlıca büyüyen (exponentially) çözümler dolayısıyla açık sıralandırma imkansızdır. Algoritmada kullanılan sınırlar çözüm uzayının bölümsel olarak aranmasına olanak tanımıştır. (Clausen, 1999).

Kesen düzlem algoritmaları ilk olarak (Gomory, 1958; Gomory, 1963) tarafından önerilmiştir. Gomory kesitleri büyük ölçekli problemlerde zayıf sonuçlar verdiğinden dolayı uzun yıllar ilgi görmemiştir. Yakın geçmişte ise (Balas ve diğ., 1996a, Letchford ve Lodi, 2002) Gomory

kesitlerinin aslında kullanılabilir olduğunu göstermişlerdir. Son zamanlarda kesen düzlem algoritmalarında yeni kavramlara yer verilmiştir. Bunlardan ikisi lift-and-project kesitleri ve Fenchel kesitleridir (Balas ve diğ., 1996b, Boyd, 1994). Ayrıca (Marchand ve diğ., 2002) tarafından ele alınan kesen düzlem çalışması ise birleşimsel eniyileme problemlerini çözen güzel bir araştırmadır. Kesen düzlem yöntemleri ile ilgili başarılı uygulamalardan oluşan araştırma ise (Jünger ve diğ., 1995) tarafından ele alınmıştır.

Kesin çözüm yöntemleri arasında bulunan dal-kesi algoritmaları kesen düzlem ve dal-sınır algoritmalarının kombinasyonundan oluşur. (Padberg ve Rinaldi, 1991) büyük boyutlu SSP' yi çözen dal-kesi algoritmasını geliştirmişlerdir. (Jünger ve diğ., 1995) ise dal-kesi ve kesen düzlem algoritmalarına uygulayıcı gözünden iyi bir referans olmuştur. Yakın geçmişte, (Mitchell, 2002) dal-kesi algoritmalarının özel problemlere nasıl uyarlanacağını ve genel kesen düzlem algoritmalarının nasıl çok çeşitli problemlerde kullanılacağını anlatmıştır.

Kesin çözüm yöntemleri kullanılarak bazı örnekler için kesin çözümler bulunmuştur (Padberg ve Rinaldi,1987 , Grötschel ve Holland, 1991, Padberg ve Rinaldi, 1991, Applegate ve diğ., 2006). Fakat giriş örneği binlerce şehir olan ve kesin bir sonuç sağlayan algoritmalar günlük kullanım için kullanışlı değillerdir. Kesin çözüm yöntemleri üzerine daha geniş ve detaylı bilgi (Winston, 1994)'te bulunabilir.

Günümüzde kesin çözüm yöntemlerini kullanarak problemleri çözmeye çalışan pek çok ticari ve akademik yazılım mevcuttur. Yakın zamanda SSP için geliştirilen önemli bir yazılım Concorde'dur (Applegate ve diğ., 2006). Concorde sadece SSP için geliştirilmiş çeşitli metriklerde çalışabilen ve dijital SSP kütüphanesinde (TSPLIB) yer alan önemli gerçek SSP örneklerinde iyi çalıştığı ispatlanan etkin bir yazılımdır. Fakat maalesef SSP genellemelerini çözecek şekilde geliştirilmemiştir. Dolayısıyla SSP genellemesi olarak tanımlanan birleşimsel eniyileme problemlerin çözümünde, problemin matematiksel formülasyonu çıkartılarak CPLEX, BARON ya da ABACUS gibi ticari yazılımların kullanımı daha iyi olabilir.

Birleşimsel optimizasyon problemlerinin çözümünde kullanılan başka bir yaklaşım sezgisel yöntemdir. Sezgisel yöntemde gereken zamanı önemli ölçüde azaltmak uğruna en iyi çözümü bulma garantisi feda edilir. Böylece birleşimsel eniyileme problemlerinin çözümünde kullanılan sezgisel yöntemler 30 sene içinde daha fazla ilgi görmüştür.

Sezgisel yöntemleri temelde yapısal yöntemler ve bölgesel arama yöntemleri olarak ikiye ayırırız. Yapısal yöntemler sıfırdan başlayarak, başlangıçta boş olan kısmi çözüme, çözüm tamamlanana kadar öğeler ekleyerek sonuca ulaşır. Bunlar genellikle en hızlı sezgisel yöntemlerdir, ancak bölgesel arama algoritmaları ile karşılaştırıldığında çoğunlukla kötü sonuçlar vermektedir. Bölgesel arama algoritmaları ilk başta herhangi bir çözümden başlar ve tekrarlı bir şekilde mevcut çözümü uygun bir şekilde belirtilmiş olan mevcut çözümün komşuluk bölgesiyle değiştirmeye çalışır.

SSP için geliştirilen yapısal yöntemlerin arasında Çevrel Çeper (Convex-Hull) ve En Yakın Komşu (Nearest Neighbor) algoritmaları söylenebilir (Stewart, 1977, Or, 1976, Lawler ve diğ., 1985). SSP için birçok bölgesel araştırma yöntemleri geliştirilmiştir. İlk olarak Croes, 2-opt algoritmasını geliştirmiştir, 10 yıllık sure geçmeden Lin 2-opt kavramını r-opt kavramına genellemiştir (Croes, 1958; Lin 1965). Or geliştirilen or-opt algoritması ile uzunlukları 3, 2 ve 1 olan zincirlerin yerlerini değiştirmiştir.(Or, 1976). Son zamanlarda, Babin ve diğerleri rastgele bir çözümden başlandığı zaman, 2-opt algoritmasının daha hızlı ve Or-opt'tan daha iyi olduğunu göstermiştir. Fakat Or-opt daha kısa zamanda daha iyi sonuçlar sağlaması için

kullanılabilir. Aynı zamanda geliştirilmiş olan Or-opt + 2-opt algoritmasının mükemmel bir kombinasyon olduğunu ve kolay uygulanabileceğini vurgulamışlardır (Babin ve diğ., 2007). Bütün bu bölgesel araştırma yöntemleri makul bir zaman içerisinde en iyi sonuçlara ulaşmada umut verici sonuçlar ortaya koymaktadırlar. SSP üzerine daha geniş ve detaylı bilgi için okuyucu (Gutin and Punnen, 2002) yönlendirilebilir.

2.3.2. Ele aldığımız Problemler

Proje önerisi verilirken ortaya çıkan parça dizgi sırasının belirlenmesinde ortaya çıkan problemin Zamana Dayalı SSP (ZDSSP), Heterojen Araç Rotalama Problemi (HARP) ya da bunların özel koşullu biçimleri olacağı tahmin ediliyordu. Dolayısıyla ZDSSP ve HARP için yapılan yazın özeti çalışmaları aşağıda verilmiştir.

Şunu da belirtmeliyiz ki ilerleyen zaman içinde parça dizgi sırasının belirlenmesi probleminin başlı başına yeni bir SSP genellemesi olduğu ortaya çıkmıştır ve çözülmesi daha zor bir problem olduğu anlaşılmıştır.

Zamana Dayalı SSP (ZDSSP): ZDSSP'nde verilen *i* ve *j* noktaları arasındaki seyahatin masrafı onların düzlemdeki konumlarına değil fakat gezinme listesinde hangi sırada olduklarına göre belirlenir (Time Dependent TSP=TDTSP) (Picard ve Queyranne, 1978). ZDSSP yazında çoğunlukla makine zaman sıralaması problemlerinde çalışılmıştır. Yazında aynı isimle anılan benzer bir problem daha vardır ki bu problemde, bir noktadan diğerine gitme masrafı farklı nedenlerden dolayı günün değişik zamanlarında değişik masraflarla ifade edilmiş ve çözülmeye çalışılmıştır (Schneider, 2002; Malandraki ve Daskin, 1992; Ichoua ve diğ., 2003; Haghani ve Jung, 2005; Albiach ve diğ., 2008). Fakat ZDSSP'nin bu tanımı bizim ilgi alanımız dışındadır.

Yazında, öneminden dolayı ZDSSP için çeşitli formülasyonlar öne sürülmüştür (Picard ve Queyranne, 1978, Wiel ve Sahinidis, 1996, Fox ve diğ., 1980). Bu formülasyonların bir özeti (Bigras ve diğ., 2008) de verilmiştir. Bunlar arasında Wiel ve Sahinidis tarafından verileni biz benimsemiş durumdayız (Wiel ve Sahinidis, 1996).

Yönsüz bir *G(V,E)* grafında *V* noktalar kümesini ve *E*'de kenarlar kümesini ifade etsin. *i* ve *j* noktaları arasında bir kenar varsa, c_{ijt}, *i* noktasından *j* noktasına yapılacak seyahatin *t* zaman aralığında yapılması durumundaki masrafıdır. İki tane değişken kullanarak formulasyonu tanımlayabiliriz.

$$x_{it} = \begin{cases} 1, & \text{eger i noktasina t zamanaraliginda ugranirsa} \\ 0, & \text{aksi takdirde} \end{cases}$$

$$z_{ijt} = \begin{cases} 1, & \text{eger j noktasina t zamaninda ve i noktasina t-1 zamanindaugranirsa} \\ 0, & \text{aksi takdirde} \end{cases}$$

Bu tanımlarla ZDSSP'nin tanımı aşağıdaki gibi yapılabilir.

$$\min \sum_i \sum_j \sum_t c_{ijt} z_{ijt} \qquad (37)$$

öyle ki;

$$\sum_t x_{it} = 1, \qquad i = 1,\ldots,n, \tag{38}$$

$$\sum_i x_{it} = 1, \qquad t = 1,\ldots,n, \tag{39}$$

$$\sum_i z_{ijt} = x_{jt}, \qquad j = 1,\ldots,n, \qquad t = 1,\ldots,n, \tag{40}$$

$$\sum_j z_{ijt} = x_{it-1}, \qquad i = 1,\ldots,n, \qquad t = 1,\ldots,n, \tag{41}$$

$$x_{it} \in \{0,1\}, \qquad i = 1,\ldots,n, \qquad t = 1,\ldots,n, \tag{42}$$

$$z_{ijt} \in \{0,1\}, \qquad i = 1,\ldots,n, \qquad j = 1,\ldots,n, \qquad t = 1,\ldots,n, \tag{43}$$

38 ve 39 numaralı kısıtlar noktalar ve zaman aralıkları arasında birebir eşleşme yapar. Böylece altturlar da elenmiş olur. 40 ve 41 numaralı kısıtlar her şehirden gelen ve her şehirden çıkan geçişleri eşitler. Wiel ve Sahinidis hızlı bir şekilde bir alt sınır elde etmek için Benders analizini bu formulasyona uygulamış ve sonra da bu alt sınır hesaplama yöntemini bir dal ve sınır algoritmasının içine yerleştirmiş ve 18 noktalı örneğe ait kesin çözüm yöntemleri elde edebilmiştir (Wiel ve Sahinidis, 1996). Öte yandan Wiel ve Sahinidis ZDSSP için bir iyileştirme sezgisel yaklaşımını içeren ve eniyisi bilinen ZDSSP örneklerini oluşturmak için de bir çalışma yapmıştır (Wiel ve Sahinidis, 1995).

Yakın zamanda Bigras ve diğerleri (2008) tek bir makine zaman sıralaması problemini bir ZDSSP örneği olarak düşünmüşlerdir. Yukarıdakinden farklı bir formülasyon kullanarak bir dal ve kesi yöntemi ileri sürmüşler ve 50 noktaya kadar olan problemlerin en iyi şekilde çözüleceğini göstermişlerdir.

Heterojen Araç Rotalama Problemi (HARP, Heterogeneous Vehicle Routing Problem=HVRP): Araç Rotalama Problemi (ARP) birleşimsel eniyileme problemleri arasında en zorlu problemlerden biridir. En genel tanımıyla, önceden tanımlanmış müşterileri ziyaret etmesi gereken bir araç filosundaki araçların toplam masraf en az olacak şekilde rotalarının belirlenmesidir. Dantzig tarafından 50 yıl önce tanımlanmıştır (Dantzig, 1959). Heterojen ARP ise araç filosunun heterojen olmasını kabul ederek problemi çözmeye çalışır. HARP'nin formulasyonu şu şekilde yapılabilir.

Yönsüz bir $G=(V',E)$ grafında $V'=\{0,1,\ldots,n\}$ kümesi $n+1$ tane noktayı içersin ve E de kenarlar kümesi olsun. 0 noktası depoyu ve $V=V'/\{0\}$ kümesi n tane müşteriyi ifade etsin. Her i müşterisi depodan q_i miktarınca mal talep etmektedir. Depoda ise heterojen bir araç filosu servis yapmayı beklemektedir. Araç filosu $M=\{1,\ldots,m\}$ kümesini oluşturan m farklı araçtan oluşmaktadır. Her bir k araç tipi için Q_k kapasitesine sahip U_k tane araç mevcuttur ve depoda beklemektedir. Her araç tipi için F_k sabit masrafı vardır. Her $\{i,j\}$ kenarı ver her araç tipi için d_{ij}^k yol masrafı verilmiştir.

Bir *k* araç tipi tarafından gezinilen R={0,i_1,…,i_r,0} rotası *G* içinde depodan çıkan ve {i_1,…,i_r} müşterilerini dolaşan ve dolaştığı müşterilerin toplam talebi Q_k araç kapasitesini aşmayan bir döngüdür. HARP her müşterinin sadece bir defa uğranarak ve her *k* araç tipi için oluşturulan rotaların sayısının en fazla U_k olacağı toplam masrafın en küçüklendiği rotalar belirlemeyi amaçlar.

R^k *k* araç tipi için bütün mümkün rotaları içeren indeks kümesi olsun ve $R = \bigcup_{k \in M} R^k$. Her $l \in R^k$ rotasının bir c_l^k masrafı vardır. $R_i^k \subset R^k$ *i* müşterisini kapsayan *k* araç tipi için belirlenen rotaların indeks altkümesi olsun. Aşağıda R_l^k'yi $l \in R^k$ rotası tarafından ziyaret edilen müşteri altkümesini ifade için kullanacağız. x_l^k ikili değişkeni 1 değerine sadece ve sadece $l \in R^k$ rotası çözüm içerisinde yer alırsa sahip olmaktadır.

$$\min \sum_{k \in M} \sum_{l \in R^k} \left(F_k + c_l^k\right) x_l^k \tag{44}$$

öyle ki;

$$\sum_{k \in M} \sum_{l \in R_i^k} x_l^k = 1, \forall i \in V \tag{45}$$

$$\sum_{l \in R^k} x_l^k \le U_k, \forall k \in M \tag{46}$$

$$x_l^k \in \{0,1\}, \forall l \in R^k, \forall k \in M \tag{47}$$

Kısıt 45 her müşterinin mutlaka sadece bir rota tarafından kapsanmasını gerektirir. Kısıt 46 ise her araç tipinde kullanılabilecek üst limiti ifade eder.

Heterojen araçlardan oluşan araç rotalama problemi ilk olarak Golden ve diğerleri tarafından tanımlanmıştır (Golden ve diğ., 1984). Bu problemdeki amaç verilen heterojen bir araç filosunun rotalarını tayin etmektir.

Bildiğimiz kadarıyla yazında heterojen araç rotalama problemi ile ilgili aşağıdaki çalışmalar mevcuttur.

Taillard, çalışmasında bir kolon üretici yaklaşımla HARP'ı çözmeye çalışmıştır (Taillard, 1999). Araçlar için sabit ve değişken masraflar ve farklı kapasiteler varsaymıştır. Fakat her araç tipindeki araç sayısı sabittir. Algoritmasında, ilk olarak her araç tipi için Adaptive Memory Procedure (AMP) kullanarak ARP çözümü elde eder. Sonra bu turlar bir HARP sonucu elde etmek için birleştirilir. Son işleme (post processing) yaparak algoritmanın performansını iyileştirir. Test çalışmaları Öklit uzayında yapılmış, veri olarak (Golden ve diğ., 1984) çalışmasında verilen verileri kullanmış ve iyi sonuçlar elde etmiştir.

Tarantilis ve diğerleri HARP çözümü için iki farklı çalışma yapmışlardır. Her iki çalışma da farklı kapasitelerde kısıtlı sayıda araç varsayarak Öklit uzayda HARP çözmeye çalışmışlardır. Bu algoritmalar eşik seviye kullanan algoritmalar sınıfına girmektedir. Verilen bir çözümle başlayıp çözüm uzayında determinist bir kontrol parametresi yardımıyla tekrar eden bir şekilde yeni çözümler aranmaktadır. Tarantilis ve diğerleri birinci çalışmalarında List

Based Threshold Accepting (LBTA) ismini verdikleri bir algoritma öne sürmüşler ve karşılaştırma çalışması olarak (Taillard, 1999) ve (Gendreau ve diğ., 1999) çalışmalarında verilen verileri kullanmış ve sekiz problemden beşinde daha iyi sonuçlar bulmuşlardır (Tarantilis ve diğ., 2003). Diğer çalışmalarında Back-Tracking Adaptive Threshold Accepting (BATA) adını verdikleri başka bir algoritma öne sürmüşler ve bu çalışmadaki sonuçlar ilk çalışmalarındaki sonuçları daha da iyileştirmiştir (Tarantilis ve diğ., 2004).

Li ve diğerleri de HARP çözümü için çalışmışlardır. "Record-to-record travel" adı verilen ve önceden bulunan bir metot HRTR ismiyle burada uygulanmıştır (Li ve diğ., 2007). Aslında bu metot benzetimli tavlama metodunun determinist bir versiyonudur. Karşılaştırılan çalışmalardaki altı problem örneğinde daha iyi sonuçlar elde edilmiştir.

3. Gereç ve Yöntem

Bu çalışmada, yukarıda tanımlanan eniyileme probleminin çözümü doğrultusunda araştırmalar yapılmıştır. Bu kapsamda, ilk olarak önceki bölümde bahsedilen yan problemlerden SDSSP'nin matematiksel modeli oluşturulmuştur.

SDSSP çözümü için, öncelikle birleşimsel eniyileme problemlerinde kullanılan kesin çözüm yöntemlerini (örneğin dal ve kesi) uygulayan ticari programlar kullanılmış ve sonrasında da bu problem için çeşitli sezgiseller geliştirilmiştir.

KAP bu projede eniyilenmesi gereken bir başka problemdir. Fakat KAP için yepyeni bir eniyileme yaklaşımı ortaya çıkarmak bu projenin amaçları arasında değildir. Ancak yazında sıkça kullanılan bazı metasezgisel yöntemler kodlanmış, performansları karşılaştırılmış ve en iyisi KAP çözücüsü olarak benimsenmiştir.

Projede ele alınacak bir diğer problem ise SDSSP ve KAP'ın beraberce eniyilenmesi problemidir (çip saçıcı makinede ortaya çıkmaktadır). Birleşimsel problemler çoğunlukla NP-Zor olarak bilindiği için her örnekte en iyi sonucu verecek kesin yöntemler oluşturmak çok zordur. Burada rastladığımız problemde SDSSP ve KAP'ın beraberce eniyilenmesi gerekmektedir. Bu yüzden çözümünün daha zor olduğu açıktır. Bu birleştirilmiş problem için de çözüm yöntemleri geliştirilmiştir.

Geliştirilecek olan çözüm yöntemleri bilgisayar yazılımları olarak hazırlanmış ve performansları rassal olarak üretilen test problemleri ile gerçek hayattan alınan üretim verileri üzerinde bilgisayısal deneylerle ölçülmüştür. Bu bilgisayısal deneyler için yüksek başarımlı bilgisayarlar ve yazılımlar kullanılmıştır.

3.1. Rassal BDK problem üreticisi

Projemizde rassal BDK problemleri üreten bir program geliştirilmiştir. Daha sonra bu programın ürettiği veriler geliştirilen yöntemlerin bilgisayısal olarak denenmesi ve kıyaslanması için kullanılmıştır. Her parçanın üç temel özelliği bulunmaktadır. Parça tipi, parçanın ağırlığıyla belirlenen ait olduğu ağırlık grubu ve yerleştirilme noktası. Bütün bunlar rassal olarak oluşturulmakla beraber bazı kurallar tanımlanmıştır. Öncelikle BDK'nın kaç parçadan (N) oluşacağı üreticiye verilir. N=100, 200, 300 ve 400 olmak üzere biz dört farklı N değerine sahip BDK'lar üretmeyi tercih ettik. Parçaların ait olacağı dört farklı ağırlık grubu belirledik. Bunlara hafiften ağıra doğru grup 1, grup 2, grup 3 ve grup 4 diyelim. Sanayide

üretilen BDK'lar incelendiğinde grup 1' e ait olacak parça sayısı en fazla olurken daha ağır gruplara ait parçaların sayısı az olmaktadır. Her gruptaki parça tipi sayısı bir ile beş arasında düzgün olarak belirlenir ve her parça tipinden rastgele olarak bir, iki ya da üç parça oluşturulur. 200, 300 ve 400 parçaya sahip BDK'larda her gruptaki parça tipi sayısı aynı kalmakta fakat her parça tipine ait parça sayısı *N* ile doğru orantılı olarak artırılmaktadır. Parçaların üzerine yerleştirildiği BDK'nın boyutları 250mmx300mm olarak kabul edilir ve aynı noktaya birden fazla parça yerleştirilmeyecek şekilde noktalar rassal olarak belirlenir.

Sanayide üretilen gerçek BDK'lar incelendiğinde en hafifler hariç aynı ağırlık grubunda olan parçaların birbirine yakın olarak yerleştirildiği gözlemlenmektedir. Bir gruba ait parçaların genelde birbirine yakın olmakla beraber bir iki tanesi gruptan uzak bir yerde olabilmektedir. Bu sayede ağır parçaların dağılımı heterojen bir yapıya sahip olmuştur. Böylece iki BDK modeli oluşturmuş oluyoruz. Birincisinde her parçanın yeri tamamen rassal olarak, ikincisinde ise ağır parçalar birbirine daha yakın olarak oluşturulmaktadır. Birincisine homojen BDK, ikincisine ise yapısal BDK ismini verdik. Böylelikle toplam parça sayısı bakımından dört ve parça dağılımı bakımından iki olmak üzere sekiz farklı BDK tipi oluşturulmuş oldu. BDK üreticisi olarak yazılan program sayesinde her BDK tipinden 100'er tane BDK oluşturularak dosyalar halinde kaydedilmiş durumdadır ve bunlar projenin ilerleyen aşamalarında karşılaştırma çalışması yaparken kullanılacaktır.

3.2.SDSSP için kesin çözüm yöntemleri

SDSSP'nin matematiksel tanımı bölüm 2.1'de verilmişti. Verilen matematiksel tanım doğrusal olmayan tamsayılı programlama problemidir. Değişkenlerimiz ikili (0,1) değişkenlerdir, amaç fonksiyonu doğrusal ancak kısıtlar doğrusal değildir. Bu nedenle, matematiksel tanımı çözecek olan algoritma, tamsayı değişkenleri ve doğrusal olmayan kısıtları ele alabilecek durumda olmalıdır.

Probleme ait örneğe ve GAMS koduna geçmeden önce, çözümde kullandığımız GAMS platformu hakkında bilgi vermek faydalı olacaktır.

"General Algebraic Modeling System (GAMS)" matematiksel tanım ve eniyilemelerde kullanılan yüksek düzeyli modelleme sistemidir. GAMS platformu kendi içerisinde bir derleyici ve bu derleyicinin kullandığı çözücülerden oluşmaktadır. Bünyesinde bulundurduğu bu çözücüler yüksek düzeyli ve geniş çaplı problemlerin çözümünde oldukça başarılıdırlar.

GAMS platformunun en önemli özelliği bir derleyiciye sahip olmasıdır. Bu derleyici ile birlikte matematiksel tanımları yazabileceğimiz programlama dili de mevcuttur. Programlama dili yapı olarak diğer dillerle benzerlik gösterir. GAMS platformunun kullandığı birden çok çözücü bulunmaktadır ve GAMS bu çözücüleri bir platform içerisinde birleştirmektedir. Derleyici üzerinde yazılan matematiksel tanımlar değişik çözücüler ile birlikte, yazılan kodu değiştirmeden çözülebilir. Diğer bir deyişle, matematiksel tanımı içeren kod birden çok algoritma ile çalıştırılabilir. GAMS' in çözücü listesinde doğrusal olmayan programlama (NLP) problemlerinin çözümünde kullanılan BARON, CONOPT3, MINOS, SNOPT gibi çözücüler mevcuttur. Yazılan doğrusal olmayan matematiksel modeller bu çözücüler yardımı ile çözülebilir, performans karşılaştırmaları yapılabilir ve sonuç olarak probleme en uygun çözüm yolu bulunabilir.

Aşağıdaki bölümde matematiksel modeli daha iyi anlamak amacı ile küçük bir örnek ve örneğin matematiksel modeline ait GAMS kodu yer almaktadır.

Yukarıda verilen SDSSP probleminin en iyi çözümünü GAMS platformunda bulabilmeyi küçük problemler için amaçladık. Yazacağımız GAMS kodunun doğru çalışıp çalışmadığını anlayabilmek için en iyi çözümü bilebildiğimiz küçük bir örnek hazırladık. Aşağıda öncelikle bu örneği anlattık, ardından da problemimizin GAMS kodunu verdik.

Örnek problemimizde aşağıdaki matriste belirtildiği gibi, numaralarla tanımlanmış altı tane parça vardır. Bununla birlikte, bu parçaların grupları ve tipleri de verilmiştir. Üç tane parça tipimiz bulunmaktadır ve karmaşıklığı önlemek amacıyla 7'den 9'a numaralandırılmışlardır.

Parçaların ait olduğu iki tane grup vardır, *g1* ve *g2* ve gruplara göre döner taretin dönme süreleri *tt1*=0.20 s ve *tt2*=0.40 s ' dir.

Parça numarası (*i*)	1	2	3	4	5	6
Parça tipi (*t*)	8	8	7	7	9	7
Parça tipi grubu (*k*)	1	1	1	1	2	1

Yukarıdaki matristen yola çıkarak ct_{it} ve g_{tk} matrisleri aşağıdaki gibi tanımlanmıştır;

ct_{it}	7	8	9
1	0	1	0
2	0	1	0
3	1	0	0
4	1	0	0
5	0	0	1
6	1	0	0

g_{tk}	1	2
7	1	0
8	1	0
9	0	1

Aşağıdaki örnek çözümde yerleşme sırası 5,6,4,1,3,2 olarak alınacak olursa x_{ip} matrisi aşağıdaki gibi olur.

x_{ip}	1	2	3	4	5	6
1	0	0	0	1	0	0
2	0	0	0	0	0	1
3	0	0	0	0	1	0
4	0	0	1	0	0	0
5	1	0	0	0	0	0
6	0	1	0	0	0	0

Bu örnekte, beş tane besleme hücremiz bulunmaktadır ve bunlardan ikisi alım yapılan bölge dışarısında yer almaktadır. Böylece aşağıdaki matristen görüldüğü gibi besleme hücrelerinden 1 ve 2'ye hiçbir parça atanmamıştır. Makinenin çalışma sistemiyle ilgili detaylar bir önceki raporda verildiğinden burada ayrıca ifade edilmesine gerek görülmemiştir.

y_{tr}	1	2	3	4	5
7	0	0	1	0	0

8	0	0	0	1	0
9	0	0	0	0	1

Bununla birlikte y_{tr} matrisinde görüldüğü gibi 3,4 ve 5 numaralı hücrelere sırasıyla 7,8 ve 9 numaralı parça tipleri atanmıştır.

Her bir adımda yer alan en ağır bileşeni bulmak amacıyla(bu yöntemle her bir adımda döner taretin hızının bulunması), aşağıda bulunan çalışma simülasyonunu örnek verebiliriz.

Kafa numarası:	1	2	3	4	5	
Besleyici hücre numarası:	1	2	3	4	5	
Parça tipi	-	-	7	8	9	döner taretin bir adım dönme zamanı
p=1	⑤	⑥	❹	❶	3	tt_2
p=2	⑥	④	①	3	2	tt_1
p=3	④	①	❸	❷	❺	tt_2
p=4	①	③	②	⑤	6	tt_2
p=5	③	②	⑤	6	4	tt_2
p=6	②	⑤	❻	4	1	tt_2

Yukarıdaki her bir yerleştirme *p* ile gösterilmiştir. Simülasyonda yer alan içleri dolu çemberler, parçanın üzerinde bulunduğu kafa tarafından alındığını gösterir. Doldurulmayan çemberler parçaların önceki yerleştirmeler sırasında alındığını ve kafa tarafından taşındığını göstermektedir. Çember içine alınmayan sayılarda ise parçanın henüz alınmadığı gösterilmiştir. Örneğin, birinci yerleştirme sırasında, (p=1) 5 numaralı parça karta yerleştirilecek olup 1 numaralı kafa tarafından taşınmaktadır. Ayrıca bu yerleştirmede 6, 4 ve 1 no' lu parçalar sırayla kafa 2, 3 ve 4 numaralı kafalar tarafından taşınmaktadır. Bununla birlikte, 4 ve 1 no' lu parçalar birinci yerleştirme sırasında alınmışlardır. Çünkü 4 no' lu parçanın tipi 3 no' lu hücrede depolanan tip ile 1 no' lu parçanın tipi ise 4 no' lu hücrede depolanan tip ile eşleşmektedir. Bu yerleştirme için, kafaların tümüne bakıldığında 2. Gruba ait parça bulunduğundan döner taretin süresi tt2 olmaktadır. İkinci yerleştirmeye (p=2), kafa 3' ün parça 1' i taşıdığı görülür çünkü önceki yerleştirme sırasında alınmıştır. Bu döner taretin zamanı tt1'e düşmesine sebep olur. Bunun sebebi taşınan bütün parçalar grup 1'in içinde yer almasıdır. Bu yerleştirme sırası altıncı yerleştirmeden sonra tekrar birinciye dönülür.

Turumuzda her bir ardışık yolculuk için verilen *CF1(x, y)* değerleri aşağıdaki gibi farz edilmiştir. *CF1(x, y)* , x köşesinden y köşesine olan yolculuk maliyetini göstermektedir ve $CF1(x,y)=\frac{d(x,y)}{v}$ Şeklindedir. Bu tanımda, V daha önceden tanımlanan hız değerini (Çalışmamızda tablo taşıyıcısının hızı 120mm/s) ve d (x, y) x ve y noktaları arasındaki mesafeyi göstermektedir.

$$2 \xrightarrow[0.4]{} 5 \xrightarrow[0.20]{} 6 \xrightarrow[0.25]{} 4 \xrightarrow[0.20]{} 1 \xrightarrow[0.40]{} 3 \xrightarrow[0.25]{} 2$$

$C_{251}=max(0.40, max(0.4, 0.2, 0.2, 0.2, 0.0)) = 0.4$ s

C_{562}=*max*(0.20, *max*(0.2, 0.2, 0.2, 0.0, 0.0)) = 0.2 s
C_{643}=*max*(0.25, *max*(0.2, 0.2, 0.2, 0.2, 0.4)) = 0.4 s
C_{414}=*max*(0.20, *max*(0.2, 0.2, 0.2, 0.4, 0.0)) = 0.4 s
C_{135}=*max*(0.40, *max*(0.2, 0.2, 0.4, 0.0, 0.0)) = 0.4 s
C_{326}=*max*(0.25, *max*(0.2, 0.4, 0.2, 0.0, 0.0)) = 0.4 s
Sonuç olarak, yolculuk maliyeti 2.2 s olarak hesaplanmıştır.

Aşağıda verilen formülasyonlar yukarıda verilen örnekle ilişkili olarak 6 parça, 3 tip ve 2 gruplu bir örnek için üretilen denklemleri göstermektedir.

Minimum zamanın bulunması amacıyla, doğrusal olmayan tamsayı programlama matematiksel tanımı aşağıdaki gibidir.

$$\min \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} \sum_{p=1}^{N} D_{ijp}$$

öyle ki
Formül (9) ve (10) için;

xip_1(c1).. x(c1,p1) + x(c1,p2) + x(c1,p3) + x(c1,p4) + x(c1,p5) + x(c1,p6) = 1
xip_1(c2).. x(c2,p1) + x(c2,p2) + x(c2,p3) + x(c2,p4) + x(c2,p5) + x(c2,p6) = 1
xip_1(c3).. x(c3,p1) + x(c3,p2) + x(c3,p3) + x(c3,p4) + x(c3,p5) + x(c3,p6) = 1
xip_1(c4).. x(c4,p1) + x(c4,p2) + x(c4,p3) + x(c4,p4) + x(c4,p5) + x(c4,p6) = 1
xip_1(c5).. x(c5,p1) + x(c5,p2) + x(c5,p3) + x(c5,p4) + x(c5,p5) + x(c5,p6) = 1
xip_1(c6).. x(c6,p1) + x(c6,p2) + x(c6,p3) + x(c6,p4) + x(c6,p5) + x(c6,p6) = 1(9)

xip_2(p1).. x(c1,p1) + x(c2,p1) + x(c3,p1) + x(c4,p1) + x(c5,p1) + x(c6,p1) = 1
xip_2(p2).. x(c1,p2) + x(c2,p2) + x(c3,p2) + x(c4,p2) + x(c5,p2) + x(c6,p2) = 1
xip_2(p3).. x(c1,p3) + x(c2,p3) + x(c3,p3) + x(c4,p3) + x(c5,p3) + x(c6,p3) = 1
xip_2(p4).. x(c1,p4) + x(c2,p4) + x(c3,p4) + x(c4,p4) + x(c5,p4) + x(c6,p4) = 1
xip_2(p5).. x(c1,p5) + x(c2,p5) + x(c3,p5) + x(c4,p5) + x(c5,p5) + x(c6,p5) = 1
xip_2(p6).. x(c1,p6) + x(c2,p6) + x(c3,p6) + x(c4,p6) + x(c5,p6) + x(c6,p6) = 1(10)

Formül (11) ve (12) için;

w(c2,c1,p1) + w(c3,c1,p1) + w(c4,c1,p1) + w(c5,c1,p1)+ w(c6,c1,p1) = x(c1,p1)
w(c2,c1,p2) + w(c3,c1,p2) + w(c4,c1,p2) + w(c5,c1,p2)+ w(c6,c1,p2) = x(c1,p2)
w(c2,c1,p3) + w(c3,c1,p3) + w(c4,c1,p3) + w(c5,c1,p3)+ w(c6,c1,p3) = x(c1,p3)
w(c2,c1,p4) + w(c3,c1,p4) + w(c4,c1,p4) + w(c5,c1,p4)+ w(c6,c1,p4) = x(c1,p4)
w(c2,c1,p5) + w(c3,c1,p5) + w(c4,c1,p5) + w(c5,c1,p5)+ w(c6,c1,p5)= x(c1,p5)
w(c2,c1,p6) + w(c3,c1,p6) + w(c4,c1,p6) + w(c5,c1,p6)+ w(c6,c1,p6) = x(c1,p6)
w(c1,c2,p1) + w(c3,c2,p1) + w(c4,c2,p1) + w(c5,c2,p1)+ w(c6,c2,p1) = x(c2,p1)
w(c1,c2,p2) + w(c3,c2,p2) + w(c4,c2,p2) + w(c5,c2,p2)+ w(c6,c2,p2) = x(c2,p2)
w(c1,c2,p3) + w(c3,c2,p3) + w(c4,c2,p3) + w(c5,c2,p3)+ w(c6,c2,p3) = x(c2,p3)
w(c1,c2,p4) + w(c3,c2,p4) + w(c4,c2,p4) + w(c5,c2,p4)+ w(c6,c2,p4) = x(c2,p4)
w(c1,c2,p5) + w(c3,c2,p5) + w(c4,c2,p5) + w(c5,c2,p5)+ w(c6,c2,p5) = x(c2,p5)
.
.

. (11)

Kalan 25 denklem atlanmıştır

w(c1,c2,p1) + w(c1,c3,p1) + w(c1,c4,p1) + w(c1,c5,p1)+ w(c1,c6,p1) = x(c1,p6)
w(c1,c2,p2) + w(c1,c3,p2) + w(c1,c4,p2) + w(c1,c5,p2)+ w(c1,c6,p2) = x(c1,p1)
w(c1,c2,p3) + w(c1,c3,p3) + w(c1,c4,p3) + w(c1,c5,p3)+ w(c1,c6,p3) = x(c1,p2)
w(c1,c2,p4) + w(c1,c3,p4) + w(c1,c4,p4) + w(c1,c5,p4)+ w(c1,c6,p4) = x(c1,p3)
w(c1,c2,p5) + w(c1,c3,p5) + w(c1,c4,p5) + w(c1,c5,p5)+ w(c1,c6,p5) = x(c1,p4)
w(c1,c2,p6) + w(c1,c3,p6) + w(c1,c4,p6) + w(c1,c5,p6)+ w(c1,c6,p6) = x(c1,p5)
w(c2,c1,p1) + w(c2,c3,p1) + w(c2,c4,p1) + w(c2,c5,p1)+ w(c2,c6,p1) = x(c2,p6)
w(c2,c1,p2) + w(c2,c3,p2) + w(c2,c4,p2) + w(c2,c5,p2)+ w(c2,c6,p2) = x(c2,p1)
w(c2,c1,p3) + w(c2,c3,p3) + w(c2,c4,p3) + w(c2,c5,p3)+ w(c2,c6,p3) = x(c2,p2)
w(c2,c1,p4)+w(c2,c3,p4)+w(c2,c4,p4)+w(c2,c5,p4)+w(c2,c6,p4)=x(c2,p3) w(c2,c1,p5) +
w(c2,c3,p5) + w(c2,c4,p5) + w(c2,c5,p5)+ w(c2,c6,p5) = x(c2,p4)
.
.
. (12)

Kalan 25 denklem atlanmıştır

Formül (5) için:
dijp_wijp(c1,c2,p1).. - 10*w(c1,c2,p1) + d(c1,c2,p1) ≥ 0
dijp_wijp(c1,c2,p2).. - 10*w(c1,c2,p2) + d(c1,c2,p2) ≥ 0
dijp_wijp(c1,c2,p3).. - 10*w(c1,c2,p3) + d(c1,c2,p3) ≥ 0
dijp_wijp(c1,c2,p4).. - 10*w(c1,c2,p4) + d(c1,c2,p4) ≥ 0
dijp_wijp(c1,c2,p5).. - 10*w(c1,c2,p5) + d(c1,c2,p5) ≥0
dijp_wijp(c1,c2,p6).. - 10*w(c1,c2,p6) + d(c1,c2,p6) ≥ 0

dijp_wijp(c1,c3,p1).. - 0.4*w(c1,c3,p1) + d(c1,c3,p1) ≥ 0
dijp_wijp(c1,c3,p2).. - 0.4*w(c1,c3,p2) + d(c1,c3,p2) ≥ 0
dijp_wijp(c1,c3,p3).. - 0.4*w(c1,c3,p3) + d(c1,c3,p3) ≥ 0
dijp_wijp(c1,c3,p4).. - 0.4*w(c1,c3,p4) + d(c1,c3,p4) ≥ 0
dijp_wijp(c1,c3,p5).. - 0.4*w(c1,c3,p5) + d(c1,c3,p5) ≥ 0
dijp_wijp(c1,c3,p6).. - 0.4*w(c1,c3,p6) + d(c1,c3,p6) ≥ 0
dijp_wijp(c1,c4,p1).. - 10*w(c1,c4,p1) + d(c1,c4,p1) ≥ 0
dijp_wijp(c1,c4,p2).. - 10*w(c1,c4,p2) + d(c1,c4,p2) ≥ 0
dijp_wijp(c1,c4,p3).. - 10*w(c1,c4,p3) + d(c1,c4,p3) ≥ 0
dijp_wijp(c1,c4,p4).. - 10*w(c1,c4,p4) + d(c1,c4,p4) ≥ 0
dijp_wijp(c1,c4,p5).. - 10*w(c1,c4,p5) + d(c1,c4,p5) ≥ 0
dijp_wijp(c1,c4,p6).. - 10*w(c1,c4,p6) + d(c1,c4,p6) ≥ 0
dijp_wijp(c1,c5,p1).. - 10*w(c1,c5,p1) + d(c1,c5,p1) ≥ 0
dijp_wijp(c1,c5,p2).. - 10*w(c1,c5,p2) + d(c1,c5,p2) ≥ 0
dijp_wijp(c1,c5,p3).. - 10*w(c1,c5,p3) + d(c1,c5,p3) ≥ 0
dijp_wijp(c1,c5,p4).. - 10*w(c1,c5,p4) + d(c1,c5,p4) ≥ 0
dijp_wijp(c1,c5,p5).. - 10*w(c1,c5,p5) + d(c1,c5,p5) ≥ 0

dijp_wijp(c1,c5,p6).. - 10*w(c1,c5,p6) + d(c1,c5,p6) ≥ 0
dijp_wijp(c1,c6,p1).. - 10*w(c1,c6,p1) + d(c1,c6,p1) ≥ 0(5)
.
.
.
KALAN 155 DENKLEM ATLANMIŞTIR.

Matematiksel modelde 4'ten 6'ya kadar bulunan denklemler bize parçaların yerleştirme zamanını kontrol etmemizi sağlar. Döner taretin dönme süresi her zaman kafalarda bulunan en ağır parça tarafından belirlenir ve herhangi bir adımda yerleştirilen parçanın yerleştirme süresi hiçbir zaman bu süreden daha az değildir. Bu yüzden matematiksel modelde bulunan 4 ile 6 arasındaki denklemler besleme hücrelerinin sayısı ile aynıdır ve her bir denklem için 180 tane kısıt üretilir.

equ1(c1,c2,p1) ..(0)*w(c1,c2,p1) + d(c1,c2,p1) + (0)*x(c1,p1) + (0)*x(c2,p1) + (0)*x(c3,p1) + (0)*x(c4,p1) + (0)*x(c5,p1) ≥ 0
equ2(c1,c2,p1)..(0)*w(c1,c2,p1) + d(c1,c2,p1) + (0)*x(c1,p2) + (0)*x(c2,p2) + (0)*x(c3,p2) + (0)*x(c4,p2) + (0)*x(c5,p2) ≥ 0
equ3(c1,c2,p1)..(0)*w(c1,c2,p1) + d(c1,c2,p1) + (0)*x(c1,p3) + (0)*x(c2,p3) + (0)*x(c3,p3) + (0)*x(c4,p3) + (0)*x(c5,p3) ≥ 0
equ4(c1,c2,p1)..(0)*w(c1,c2,p1) + d(c1,c2,p1) + (0)*x(c1,p4) + (0)*x(c2,p4) + (0)*x(c5,p4) ≥ 0
equ5(c1,c2,p1)..(0)*w(c1,c2,p1) + d(c1,c2,p1) + (0)*x(c5,p5) =G= 0 ; (LHS = 0)

Kalan 895 denklem atlanmıştır.
Yukarda bulunan 5 denklemden her biri bir besleme hücresini temsil etmektedir.
Problemin çözümü için yapılan bilgisayısal çalışmalar Bulgular kısmında detaylıca verilmiştir.

3.3.SDSSP için geliştirilen sezgisel çözüm yaklaşımları

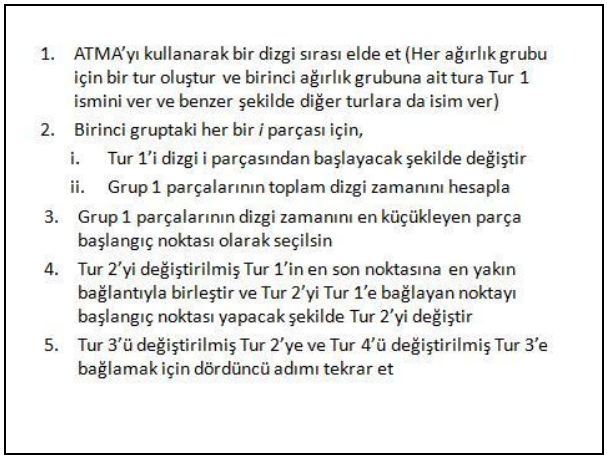
Projemizde SDSSP için etkin sezgisel çözüm yaklaşımlarının geliştirilmesi ve programlanmasının yapılması da hedeflenmiştir. Aşağıda bu çalışmaların detayları verilmektedir.

3.3.1. İlk Noktayı Değiştir Yöntemi

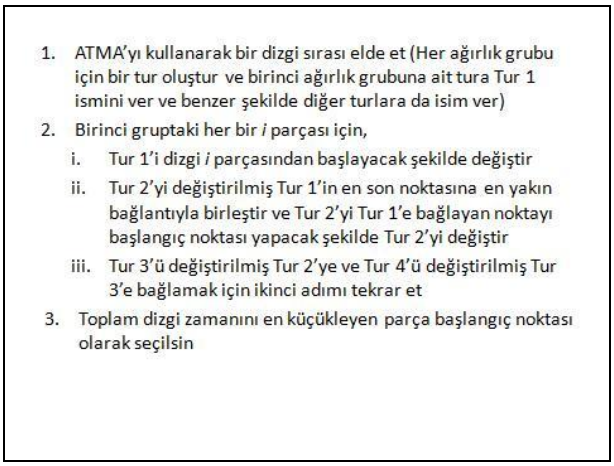
SDSSP'nin çözümü için geliştirdiğimiz bir yöntem İlk Noktayı Değiştir Yöntemi'dir (Adjust First Point Procedure). Bu yöntem tura başlangıç noktasının fark etmeyeceği ve herhangi bir nokta olabileceği problemlerde geçerlidir ve BDK üretimi böyle bir problemdir. Bu yöntemde ilk olarak Convex-Hull ve Or-Opt algoritmalarını kullanarak bir SSP turu elde ederiz. Sonrasında, her *i* noktası için, sanki dizgi bu *i* noktasından başlıyormuş gibi toplam dizgi zamanını hesaplarız. Toplam dizgi zamanını en küçükleyen başlangıç noktası dizgiye başlama noktası olarak seçilir.

İlk Noktayı Değiştir Yöntemi (İNDY) ve İlk Noktayı Toplam Masrafa göre Değiştir Yöntemi (İNTMDY) bu anlayış doğrultusunda geliştirilmiştir. İNDY'de sadece grup 1 parçalarının oluşturduğu tura ait dizgi masrafı esas alınmaktadır (Şekil 3). İNTMDY'de ise toplam dizgi zamanı göz önüne alınarak başlangıç noktası belirlenmektedir. Hatırlayalım ki grup 1 parçalarının dizgisi bittikten sonra grup 2, grup 3 ve grup 4 parçaları yerleştirilir ve her bir sonraki turun başlangıç noktası önceki turun bitiş noktasına en yakın nokta olarak belirlenir.

Fakat eğer grup 1'in başlangıç noktasını değiştirirsek bitiş noktası da değişir ve bunun sonucu olarak sırasıyla grup 2'nin, grup 3'ün ve grup 4'ün başlangıç ve bitiş noktaları da değişir ve toplam dizgi zamanı da tekrar hesaplanmalıdır. İNTMDY'de bu değişikliğe uygun olarak toplam dizgi masrafı hesaplanır (Şekil 4).



Şekil 3—İlk Noktayı Değiştir Yöntemi



Şekil 4— İlk Noktayı Toplam Masrafa göre Değiştir Yöntemi

3.3.2. Sapak Olanları Ertele Yöntemi

SDSSP'nin çözümü için önerdiğimiz bir başka yöntem ise Sapak olanları Ertele (SE) yöntemidir. Parçaları ağırlıklarına göre gruplandırma ve her bir grup için bir tur oluşturma ve bu turları sırasıyla gezinme yazında rastlanan bir yöntemdir. Ancak bu yöntem bazen verimsiz dizgi sıralarına neden olabilir. Eğer bir parça kendi grubuna ait diğer parçalardan daha uzak bir noktaya yerleştirilecekse bu parça sapak parça diyebiliriz. Böyle sapak bir parçayı başka bir grubun parçalarıyla beraber yerleştirmek toplam dizgi zamanını iyileştirebilir. Başka bir örnek olarak, bir parça dizgi sırasından çıkarıldığında toplam dizgi zamanının değişmediğini düşünelim (bu durumda bu parçanın öncesinde ve sonrasında yerleştirilmesi planlanan parçalar arasındaki uzaklık taret zamanı içinde alınıyor demektir). Böyle bir parçayı başka bir grup içerisinde yerleştirmek belki de iki uzak parçayı birleştirerek toplam dizgi zamanını azaltabilir. Bu yöntemde dikkat edilmelidir ki hafif olan parçaları ağırlar arasında yerleştirmek iyileştirme getirebilir ama ağır parçaları hafifler arasında yerleştirmek

kesinlikle iyileştirme getirmeyecektir. Bunun nedeni ağır olan bir parçanın hafiflerin arasına karıştırılması önceden yerleştirilen belli sayıda parça için taret zamanını artıracaktır. Bu yüzden nispeten hafif olan parçalar ağır olan parçalar arasında yerleştirilmeye çalışılmalıdır (Şekil 5).

1. Grupları hafiften ağıra doğru sırala
2. Her bir *m* grubu için
   - *m* grubundaki her bir *i* parçası için
     - *m*'den ağır her bir *n* grubu için
       - *n* grubundaki her bir *j* parçası için
         - Geçici olarak *i* noktasını *j*'den sonraya ekle ve toplam masraf azalmışsa bu değişikliği kalıcı olarak sakla

Şekil 5—Sapak olanları Ertele Yöntemi

SE'nin biraz daha gelişmiş versiyonu olan Gelişmiş SE'de (GSE) SE yöntemi toplam dizgi zamanında iyileştirme elde edilmeyene dek tekrar tekrar çağrılır (Şekil 6). Kolaylıkla tahmin edilebilir ki, SE'yle kıyaslandığında, GSE daha çok zaman almakta fakat kesinlikle daha iyi sonuçlar vermektedir.

Yap

1. dizgiZamanı = mevcut dizgi sırasının toplam masrafı
2. SE'yi uygula
3. yeniDizgiZamanı = mevcut dizgi sırasının toplam masrafı

yeniDizgiZamanı ≠ dizgiZamanı sürece

Şekil 6—Gelişmiş Sapak olanları Erteleme Yöntemi

3.3.3. RRTLEM

HARP yazınında rastladığımız önemli bir iyileştirme metodu RRT'dir (Li ve diğ., 2007). Bu algoritmayı modifiye ederek ve başka yöntemlerle birleştirerek RRTLEM algoritmasını geliştirdik.

RRTLEM (Record-to-Record Travel with Local Exchange Moves) bizim geliştirdiğimiz RRT'yi yerel değiştirme hareketleriyle birleştiren bir yerel arama algoritmasıdır. RRTLEM'den bahsetmeden önce RRT'den bahsetmekte fayda vardır. RRT, Benzetimli Tavlama BT (Simulated Annealing) yönteminin deterministik bir versiyonudur ve RRT'nin BT'den daha iyi sonuçlar verebildiği gösterilmiştir (Dueck, 1993).

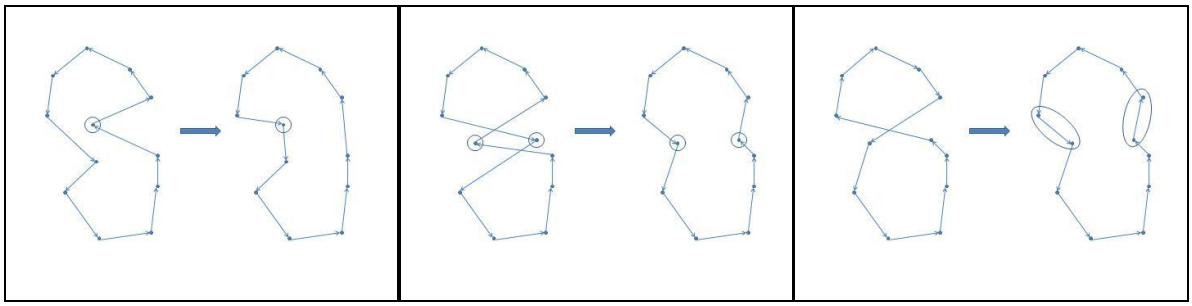
RRT bir ilk çözümle başlar. Bizim çalışmamızda bu ilk çözüm ATMA uygulanarak elde edilir. Elde edilen ilk çözüm mevcut çözümü gösteren *s*'ye atanır. *bs* değişkeni de algoritma boyunca elde edilen en iyi çözümü tutar ve o da ilk değer olarak *s*'nin değeri *bs*'ye atanır. Bir çözümün masrafını *f(s)* fonksiyonu versin. Bizim çalışmamızda *f(s)*, verilen dizgi sırasının (*s*) toplam dizgi zamanını verir. *Record* değişkeni bulunan en iyi çözümün masrafını saklı tutar. *Deviation* değişkeni *Record*'un önceden tanımlı belli bir yüzdesidir (*rate*). *N(s)*, *s* çözümünün

komşuları olan çözümleri içeren kümeyi ifade eder ve *PickNextNeighbor()*, *n(s)* içinden bir sonraki komşu çözümü belli bir kural veya sıraya göre verir. RRT deterministik bir algoritmadır çünkü komşu bir çözüm olan s', sadece masrafı *Record+Deviation*'dan düşük ise *s*'nin yerini alır (Şekil 7).

```
s := GenerateInitialSolution()
bs := s
Record := f(bs)
Deviation := Record x p
While termination conditions not met
    s' := PickNextNeighbor(N(s))
    If f(s') < Record + Deviation
        s := s'
        If f(s') < Record
                bs := s'
                Record := f(s')
                Deviation := Record x p
        Endif
    Endif
Endwhile
```

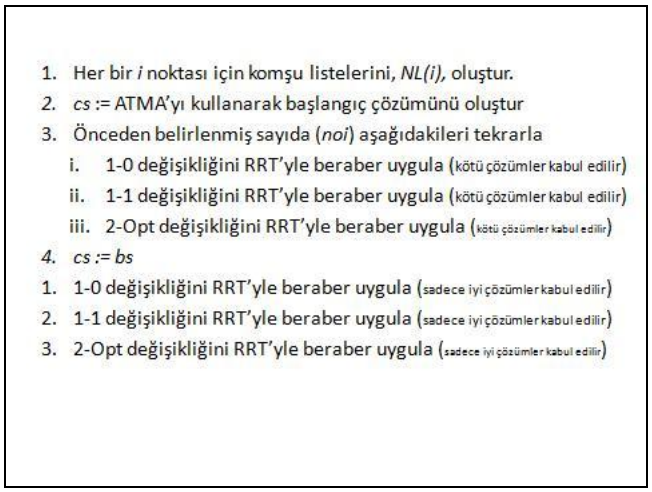
Şekil 7—RRT

Mevcut çözümden yeni çözümler elde etmek için yerel aramalar yapmak gerekir. SSP ve genellemelerinde çokça uygulanan bir yöntem noktaların ve kenarların diziliş sırasını değiştirmektir (Waters, 1987; Tarantilis ve diğ., 2004). Noktaları değiştirmek için iki seçenek mevcuttur; ya bir noktayı tur üzerinde başka bir yere alabilirsiniz (1-0 Değişikliği) ya da iki noktanın yerini karşılıklı olarak değiştirirsiniz (1-1 Değişikliği). İki kenarın değiştirilmesine ise 2-Opt değişikliği diyoruz (Şekil 8).

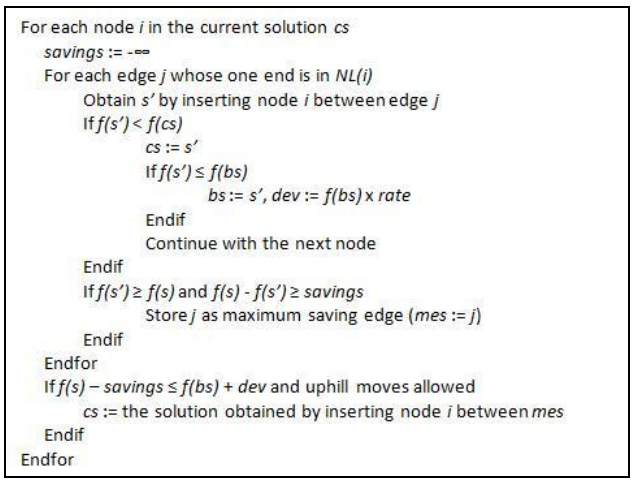


Şekil 8—Sırasıyla 1-0 Değişikliği, 1-1 Değişikliği ve 2-Opt değişikliği

Geliştirdiğimiz RRTLEM algoritması RRT'yi yerel değiştirme hareketleriyle birleştirmektedir (Şekil 9). Aslında bu algoritma basit tekrarlamalı ifadelerden oluşmakta ve RRT'nin yerel değiştirme hareketlerinin içine gömülmesi detaylı olarak sonraki şekilde ifade edilmektedir (Şekil 10). RRTLEM grup 1 parçaları için uygulanmaktadır.



Şekil 9—RRTLEM Algoritması



Şekil 10—RRT'yle beraber 1-0 Değişiklik hareketi

RRTLEM algoritmasının performansını belirleyen iki önemli parametresi tekrar sayısı (*noi*) ve *Deviation* değişkeninin hesaplanmasında kullanılan yüzde oranıdır (*rate*). Büyük bir *noi* değeri kullanmak arama uzayını genişletir fakat aynı zamanda çalışma zamanını artırır. Öte yandan düşük bir *rate* değeri yerel minimum noktalarından kurtulma şansını azaltır ve yüksek bir *rate* değeri algoritmanın gereksiz yere zirve noktaları arasında dolaşmasına neden olur. Dolayısıyla, RRTLEM algoritmasında bu parametrelerin mantıklı değerleri kullanılmalıdır ve bu değerlerin elde edilmesi için geniş bir ön çalışma yapmak gerekir.

3.4. KAP İçin Yapılan Çalışmalar

KAP literatürü incelendiğinde çok sayıda çalışmaya rastlanmaktadır. KAP ilk olarak Koopmans ve Beckman (1957) tarafından önerilmiştir. Bu konuda meta-sezgisel tekniklerin kullanılmasına son zamanlarda daha sık rastlanır olmuştur (Chwif vd.,1998). KAP için meta-sezgisellerden faydalanılan bazı çalışmalar: Cung ve diğ.(1997), Chwif ve diğ. (1998), Ramkumar ve diğ. (2008) ve Duman ve Or (2007) tarafından hazırlanmıştır. Duman ve Or'un çalışması, söz konusu literatürde en çok kullanılan üç meta-sezgiselin Tabu Araması,

Benzetim Tavlaması ve Genetik Algoritma bir karşılaştırmasını sunması açısından önemli bir çalışmadır.

Karesel Atama Problemi (KAP) en basit tanımıyla *n* tane ofisi *n* tane çalışana şu amaç, bilgi ve kısıtlar altında atamaktır: Bize verilen bir uzaklık matrisi her ofisin birbirine olan uzaklığını içerir (c_{ij}) ve bir ilişki matrisi her çalışanın bir diğeriyle günlük görüşme sayısını içerir (d_{kl}). Her çalışanın sadece bir ofisi olacak şekilde çalışanlarca hergün alınan toplam mesafeyi en küçüklemek bu problemin amacıdır.

Bu projede Benzetimli tavlama, Genetik algoritma, Scatter arama ve Grasp metotları ile programlama yapılarak karşılaştırmaları yapılmış ve en iyi meta-sezgisel çözüm metodu sunulmuştur. Grasp metodu özellikle KAP için geliştirilmiştir. Kullanacağımız diğer metotlar ise KAP uygulamalarında kullanılan en meşhur metotlardır. Duman ve Or (2007) ile de birleştirildiğinde tüm önemli meta sezgisellerin performansları değerlendirilebilmiş olacaktır.

JAVA ortamında CM (Comparison of Methods – Metotların karşılaştırılması) isimli bir yazılım geliştirilmiştir. Bu yazılımda elde edilen sonuçlar, Duman ve Or (2007)'den alınan literatür problemleri test edilerek elde edilmiştir. Her problem için bilinen en iyi çözüm metodu sunulmuştur.

3.4.1. Benzetimli Tavlama

Benzetimli Tavlama algoritması, Burkard ve Rendl tarafından KAP için 1984 yılında geliştirilmiştir.

Benzetimli Tavlama'da kötü çözümler çözüm kümesine dahil edilebilir. Araştırma boyunca da kötü çözümler azalma gösterir. Başlangıçta T (sıcaklık) değeri yüksektir. T sıfır değerinden büyük ve iterasyon da maximum iterasyon sayısından (Noi) küçük olana kadar, işlem devam eder. T sıfır değerinden küçük veya iterasyon da maximum iterasyon sayısından(Noi) büyük olma durumda, araştırma sonlandırılır.

Bizim yazdığımız Benzetim Tavlama Algoritması aşağıdaki gibidir (Şekil 11):

```
Algoritma BT
while(T>0&&totalR<noi){
   for(j=0;j<R;j++){
   ns=cs.neighbour();
   delta=ns.getCost()-bs.getCost();
   if(ns.getCost()<cs.getCost()){cs=ns;}
   else if(Math.random()<Math.exp(-Math.abs(delta)/T)){cs=ns;}
   if(ns.getCost()<bs.getCost()){bs=ns;}}
                    T/=a;
                    R*=b;
                    totalR+=R;
 }
```

Şekil 11— Benzetim Tavlama Algoritması

İlk olarak başlangıç derecesi (T), derecenin azalma oranı (a), maximum iterasyon sayısı (Noi), her bir derece için iterasyon sayısı (R) ve iterasyonun artma oranı (b) değerleri verilir.

Rastgele seçilmiş ‘cs’ çözümünden oluşan rastgele komşular ‘ns’ değerine atanır. ‘bs’ de en iyi çözümü göstermektedir. ΔE_{ns} ve ΔE_{bs}, iki başarılı enerji durumudur. ΔE_{ns} ve ΔE_{bs} değerlerinin farkı alınır ve ΔE değerine atama yapılır.

Eğer ΔE değeri sıfırdan küçük ise, enerji azalması oluşmuştur ve işlem devam eder. Diğer anlamda, uygunluk değerinin (cost function) düştüğünü göstermektedir. Eğer ΔE değeri sıfıra eşit ise, aynı durumu koruduğunu göstermektedir.Yani enerji durumunda değişiklik yoktur. ΔE değeri sıfırdan büyük olmuş ise enerjinin arttığını göstermektedir. Diğer deyişle, uygunluk değerinin (cost function) arttığını göstermektedir.

Eğer ‘ns’ değerinin uygunluk değeri, ‘cs’ değerinin uygunluk değerinden küçük ise ‘ns’ değerini ‘cs’ değerine ataması yapılır.
Eğer random(0, 1) ≤ exp(−δf/T) ise ‘ns’ deki değeri ‘cs’ değerine atama yap.
Eğer ‘ns’ değerinin uygunluk değeri ‘cs’değerinin uygunluk değerinde küçük ise ‘ns’ değerini ‘bs’ değerine ataması yapılır.
En son olarak;
T değerini T/a kadar azaltılır.
R değerini R*b kadar arttırılır.
R değerini Total R değerine eşitlenir.
T’nin sıfır değerinden küçük veya iterasyon sayısının maximum iterasyon sayısından büyük olma durumda araştırma sonlandırılır.

3.4.2. Genetik Algoritma

Genetik Algoritma, Holland tarafından 1975 yılında geliştirilmiştir (Holland, 1975). 1989 yılında Goldberg tarafından oluşturulan genetik algoritmalar rastgele başlangıç çözümlerinin popülasyonuyla başlar. Popülasyondaki her bir ferdin (problemin yapısına göre önceden belirlenen) bir kuralla ya da formülle uygunluk değeri hesaplanır. Yine bir kurala göre popülasyon içinden çiftler seçilir ve çaprazlama ve mutasyon işlemleri uygulanarak yeni bireyler elde edilir ve böylece yeni bir nesil elde edilir. Sayısı artan popülasyondan bir kurala uygun olarak bazı fertler çıkartılır. Yeni bireyler ve nesiller oluşturma bu şekilde devam eder.

Başlangıç popülasyonu oluşurken, popülasyonun boyutu kadar birbirinden farklı bireyler oluşturulur. Popülasyondaki her bir bireyin temsil ettiği çözüm değerlendirilir ve her bir bireye taşıdığı çözümün kalitesine göre bir uygunluk değeri (fitness) verilir.

3.4.2.1. Seçme yöntemleri

Yazında rastladığımız kadarıyla, KAP için 2 tane seçme yöntemi uygulanmaktadır.

i. Rulet-Çemberi Seçim Tekniği: Seçim yöntemleri içinde en basit ve en kolay uygulanabilir nitelikteki yöntem, rulet-çemberi yöntemidir: tüm fertler bir çizgi üzerinde birbirine bitişik bölümler şeklinde dizilirler. Bununla birlikte her bir ferde ilişkin bölümün uzunluğu, onun uygunluk değeri kadar olur. Rastgele sayı üretilir ve rastgele sayı hangi bölüm içine denk gelirse, o bölümün ait olduğu fert seçilir. İşlem eşleşecek popülasyonun gerekli adedine ulaşılıncaya dek sürdürülür. Bu teknikte, ebeveynler uygunluk değerlerine göre seçilirler. Kromozomlar ne kadar iyiyse, seçilme şansları o kadar yüksektir. Bir kromozom birden fazla seçilebilir. (Mahdavi ve diğ., 2009).

ii. Turnuva Seçim Tekniği: Rastgele n adet birey seçilir. n adet içindeki en iyi uygunluk değerine sahip kromozom, yeni nesile aktarılır. Popülasyon büyüklüğüne ulaşıncaya kadar bu işlem devam edecektir (Dikos ve diğ,1997).

Rulet-Çemberi seçim tekniğinin, Turnuva seçim tekniğine göre daha iyi sonuç verdiği gözlenmektedir (Dikos ve diğ.,1997).

3.4.2.2. Çaprazlama yöntemleri

Yazında rastladığımız kadarıyla, KAP için 6 tane çaprazlama yöntemi uygulanmıştır.

i. PBX: Birinci ebeveynden rassal olarak (%50 olasılıkla) seçilen genler, pozisyonlarını koruyarak oluşturulacak yavru kromozoma aktarılırlar. Kalan genler ise ikinci ebeveynden sıraları bozulmadan soldan sağa doğru yavru kromozoma aktarılırlar (Misevicius ve diğ., 2005).

ii. PMX: Rastgele olarak kromozomda bir aralık belirlenir. Belirlenen aralık ebeveynler arasında karşılıklı olarak yer değiştirir. Aynı kromozomda aynı genin birden fazla bulunması söz konusu olabilir. Bu durumda aralık dışında yer alan aynı değere sahip olan genler diğer kromozomda aralık dışında tekrar eden genlerin yerine yerleştirilir (Misevicius ve diğ., 2005).

iii. OX: Rastgele 2 kromozom ve 2 kesim noktası seçilir. Kesim noktaları arasındaki kromozomlar karşılıklı olarak yer değiştirir. Kesim noktaları dışında yer alan genler içerisinde tekrarlı genler oluşursa bunlar yerine sıra ile soldan sağa doğru kromozomda bulunmayan genler yazılır (Misevicius ve diğ., 2005).

iv. CX: İlk kromozomdan en baştaki gen seçilir ve bu gen yeni diziye yerleştirilir. Bu gene karşılık gelen ikinci kromozomdaki gen belirlenir bu değer de yeni kromozom üzerine yerleştirilerek dairesel bir şekilde bütün genler belirlenir. (Mawdeslev ve diğ., 2003).

v. ULX : Rastgele iki tane ebeveyn seçilir. Aynı pozisyondaki gen değerleri aynı ise, yeni kromozomun üzerine yerleştirilir. Kalan genler rastgele seçilerek, yeni kromozomun üzerine soldan sağa doğru yerleştirilir (Mahdavi ve diğ., 2009).

vi. MPX: Rastgele ikiden fazla ebeveyn seç. Aynı pozisyondaki gen değerleri aynı ve sayısı fazla ise, geni kromozomun üzerine yerleştirilir. Diğer pozisyonlar içinde aynı yöntem uygulanır (Misevicius ve diğ., 2005).

MPX çaprazlama yöntemi, tabu arama ve genetik algoritma birleşiminden oluşan algoritma ile çözüldüğünde, diğer çaprazlama yöntemlerine göre en başarılı seçilmiştir (Misevicius ve diğ., 2005).

PMX çaprazlama yöntemi, genetik algoritmada Rulet çemberi seçim yöntemi ile kullanılmış ve Turnuva seçim yöntemine göre daha iyi sonuç vermiştir (Dikos ve diğ.,1997). KAP, TSP gibi problemler için PMX uygundur (Chan ve diğ., 2006).

OX çaprazlama yöntemi ve OPX çaprazlama yöntemi, KAP çözümleri için en kötü yöntem olarak belirtilmiştir (Mahdavi ve diğ., 2009; Ravindra ve diğ., 2000).

3.4.2.3. Mutasyon Yöntemleri:

Yazında rastladığımız kadarıyla, KAP için kullanılan 3 tane mutasyon yöntemi vardır.

i. Yanındakiyle yer değiştir: Rastgele bir sayı oluştur (R), [1,n] doğrultusunda, R değerinin pozisyonunu belirle ve (R+1) pozisyonunla yer değiştir (Ramkumar, ve diğ., 2009).

M(2 1 3 4) = 2 3 1 4

ii. Atlama (Swap): Rastgele iki sayı oluştur (R_1,R_2)., [1,n] doğrultusunda, (R_1 ve R_2) sayıların yerlerini değiştir (Ramkumar ve diğ., 2009).

M(2 1 3 4) = 4 1 3 2

iii. Ekleme (Insert): Rastgele iki sayı oluştur (R_1,R_2)., [1,n] doğrultusunda, R_2 değerini R_1 değerinin bulunduğu yere ekle ve diğerlerini 1 kaydır (Ramkumar ve diğ., 2009).

M(2 1 3 4) = 2 4 1 3

Mutasyon yöntemleri karşılaştırıldığında, Ekleme (Insert) mutasyonun daha iyi bir yöntem olduğu gözükmektedir (Ramkumar ve diğ., 2009; Ravindra ve diğ., 2000).

Yanındakiyle yer değiştir yöntemi, KAP için uygun bir mutasyon yöntemi değildir (Chan ve diğ., 2006).

Çarprazlama ve Mutasyon Oranları:

i. P (population size): 100, G (generation size):40, PC (the probability of crossover): 0.90 ,PM(the probability of mutation): 0.10 , R(a number of runs): 10 kullanılmıştır. (El-Baz.M.A.,2004).

ii. Benchmark problem çözümü için: P: 50- 200, G: Değişiklik gösterir, PC:0.70-0.80, PM:0.01-0.10 kulanılmıştır. (Mahdavi, I. ve diğ, 2009).

iii. P: 40, G:1200,PC:0.40,PM:0.20 kullanılmıştır. Memnuniyet verici bir sonuç elde edilmemiştir. (Gong,D.ve diğ,1999).

iv. P:100, G:3000, PC: %55 ,PM:% 5 ,PR (selection): %45, çarpazlama: PMX, seçme yöntemi: Rulet çemberi kullanılmıştır (Dikos,A. ve diğ,1997).

Durma Kriteri:

i. Maximum popülasyon büyüklüğüne geldiği zaman, işlem sonlanır (El-Baz, 2004).

ii. Bir önceki bulunan en iyi değerde değişme olmuyorsa, işlem sonlanır (Mantawy ve diğ.,1999).

3.4.2.4. Gerçekleştirilen Genetik Algoritma

Projemizde KAP'ı çözmek için gerçekleştirilen Genetik Algoritma Şekil 12'deki gibidir.

Algoritma GA
1. Populasyon (pop) = yarat.random()

```
2.  bs = pop.Min().getCost()
3.  While ( gs != G ||  bs != newbs)
    {
4.      newbs = bs
5.      x  = pop.secRulet Çemberi Tekniği()
6.      y  = pop.secRulet Çemberi Tekniği()
7.      (c , n)  = Çarpr. PMX (x,y)
8.      (c , n)  = Mutasyon. Ekleme(c, n)
9.       pop = pop + (c, n)
10.     10.a  newpop = SF (pop) veya  10.b newpop = SS(pop)
11.     pop.sil()
12.     pop = newpop
13.     bs = pop.Min().getCost()
14.  }
10. a SF  { for (int k=0; k < P ; k++) z = pop.getFitness()
            pop.ekle(z)  }
10.b  SS { for (int k=0; k < P ; k++) z = pop.secRulet Çemberi Tekniği()
            pop.ekle(z)  }
```

Şekil 12— Genetik Algoritma

Proje kapsamında, Genetik Algoritma ile ilgili iki tane yazılım çalışması yapılmıştır. İki yazılım çalışması ile ilgili tek ayrım en iyi çözümü bulurken 10.a'da SF (En iyi çözümleri seç) diğerinde ise SS (Rulet Çemberi tekniğini kullanarak seç) metotlarının kullanılmasıdır.

İlk olarak çözümlerin sayısı (P), nesil sayısı (G) değerleri verilir. (1) Rastgele seçilmiş başlangıç çözümleri ile populasyon oluşturulur. (2) Populasyondaki her bir çözümün uygunluk değeri (cost function) hesaplanır. En düşük uygunluk değerine sahip olan çözüm seçilir ve en iyi çözüm 'bs' olarak atanır. (3) Bu işlem G değerine ulaşıncaya kadar veya en iyi çözüm değeri değişmediği sürece devam edecektir. (4) 'bs' değişkeni 'newbs' değişkenine atanır. (5 ve 6) Rulet Çemberi tekniği ile 'x' ve 'y' bireyleri seçilir. (7) Seçilen 'x' ve 'y' bireyleri PMX metodu kullanılarak çaprazlamaya uğratılır. (8) Çaprazlama sonucunda oluşan 'c' ve 'n' bireylerine Ekleme (Insertion)- Mutasyon yöntemi uygulanır. (9) 'c' ve 'n' bireyleri populasyona eklenir. (10.a) 1. yazılım çalışması kapsamında: SF metodu kullanılır. Sonraki nesli oluşturacak bireyler için uygunluk değerinin en iyi olması durumu ele alınarak oluşan yeni nesil bireyler 'newpop' değişkenine atama yapılır). (10.b) 2. yazılım çalışması kapsamında: SS metodu kullanılır. Sonraki nesli oluşturacak bireyler için Rulet Çemberi tekniği kullanılır. Yeni nesil bireyler 'newpop' değişkenine atama yapılır). (11) 'pop' silinir. (12) 'newpop' değişkeni 'pop' değişkenine atanır. (13) Populasyonun (pop) uygunluk değeri (cost function) hesaplanır. En düşük uygunluk değerine sahip olan çözüm seçilir ve en iyi çözüm 'bs' olarak atanır.

3.4.3. Dağınık Arama

Dağınık Arama Algoritması, Glover tarafından 1977 yılında geliştirilmiştir. Dağınık Arama metodunun KAP üzerindeki ilk uygulamasını 1997 yılında Cung ve arkadaşları gerçekleştirmiştir. Dağık arama algoritmasının en önemli özelliği populasyonu oluştururken tekrarlanmayan çözümlerden oluşmasıdır. Bundan dolayı, farklılık metodunu kullanmak

gerekmektedir. Aynı zamanda, mutasyon ve çaprazlama da geliştirme metodu olarak kullanılmaktadır. Dağınık arama ile ilgili yayınlarının sayısı oldukça azdır (Loiola ve diğ., 2007).

3.4.3.1. RefSet Oluşturma Metodu (Generation Method):

RefSet, b1 tane en iyi çözümden (high quality) ve b2 tane en farklı (high diverse) çözümden oluşmaktadır (Marti ve diğ., 2006).

3.4.3.2. Farklılık Metodu (Diversification Method):

Referans kümenin aynı değerleri içermemesi gerekmektedir. Bundan dolayı, farklılık metodu çok önemlidir ve tabu listesi kullanarak farklılık yaratılmaktadır. Her bir çözüm populasyona eklenmeden önce tabu listesinde olup/olmadığı kontrol edilir. Eğer çözüm tabu listesinin elemanı değilse, çözüm tabu listesine eklenir. Eğer çözüm tabu listesinin elemanı ise, bu çözüm elenir. Böylelikle, farklı çözümler içeren bir populasyon ve Referans küme oluşturulur.

3.4.3.3. Mutasyon Yöntemleri:

İki tane mutasyon yöntemi vardır. Atlama (Swap) ve Ekleme (Insert) mutasyon yöntemleri Genetik Algoritma altbölümünde detaylıca açıklanmıştır.

3.4.3.4. En Farklı Çözümleri Bulma Yöntemi (High Diverse Solution Method):

İki çözüm arasındaki en büyük uzaklık farkı, en farklı çözümü bulmamızı sağlamaktadır. İki çözüm arasındaki uzaklığın hesaplanması aşağıdaki formülde sunulmuştur (Yuan ve diğ., 2007).

$d(p,q) = \Sigma^{n-1}_{i=1}$ uzaklık p_{i+1} ve p_i arasında q bazında

3.4.3.5. Rulet-Çemberi Seçim Tekniği:

Seçim yöntemleri içinde en basit ve en kolay uygulanabilir nitelikteki yöntem, rulet-çemberi yöntemidir. Rulet-çemberi yöntemi: tüm fertler bir çizgi üzerinde birbirine bitişik bölümler şeklinde dizilirler. Bununla birlikte her bir ferde ilişkin bölümün uzunluğu, amacımız en küçükleme olduğu için onun uygunluk değeriyle ters orantılı olarak belirlenir. Böylelikle en iyi olan çözüme en büyük uzunluk verilmiş olur. Rastgele sayı üretilir ve rastgele sayı hangi bölüm içine denk gelirse, o bölümün ait olduğu fert seçilir. İşlem eşleşecek popülasyonun gerekli adedine ulaşılıncaya dek sürdürülür. Bu teknikte, ebeveynler uygunluk değerlerine göre seçilirler. Bireyler ne kadar iyiyse, seçilme şansları o kadar yüksektir. Bir birey birden fazla kez seçilebilir. (Mahdavi ve diğ., 2009).

3.4.3.6. Çaprazlama Yöntemleri:

i. PMX: Rastgele olarak kromozomda bir aralık belirlenir. Belirlenen aralık ebeveynler arasında karşılıklı olarak yer değiştirir. Aynı kromozomda aynı genin birden fazla bulunması söz konusu olabilir. Bu durumda aralık dışında yer alan aynı değere sahip olan genler diğer kromozomda aralık dışında tekrar eden genlerin yerine yerleştirilir. (Misevicius ve diğ., 2005).

ii. Birleştirme Metodu (Combine Method): Birleştirme metodu ile sadece bir tane çözümün üretilmesi sağlanır. Birleştirme metodu üç bölümden oluşmaktadır. İlk olarak, başlangıç yeri seçilir ve iki çözüm sol tarafa doğru kaydırılır ve soldaki değerleri ise sağ tarafın en sonundan başlatılır ve ikinci olarak, p1 ve p2 arasındaki uzaklıklar karşılaştırılır, en kısa mesafeye sahip olan çözüm sol tarafa doğru kaydırılır aynı değere eşit ise c değerine atılır ve aynı değerler p1 ve p2 çözümlerinden çıkartılır, bu işlem p1 ve p2 değerleri bitinceye kadar devam eder. (Yuan ve diğ., 2007).

3.4.3.7. RefSet Güncelleme Metodu:

İlk olarak, tabu listesindeki çözümler silinir. RefSet'deki çözümler ile yeni çözümler tabu listesine eklenir. Tabu listesinde birbirinden farklı çözümler bulunmaktadır. Tabu listesinden b1 tane en iyi çözümden (high quality) ve b2 tane en farklı (high diverse) çözümden seçilir ve Refset güncellenir (Marti.R ve diğ., 2006).

3.4.3.8. Durdurma Kriteri:

RefSet'de yeni çözümler bulunmaması durumunda, işlem sonlanır (Marti ve diğ., 2006).

Dağınık Arama algoritması ile ilgili iki tane yazılım çalışması yapılmıştır. Birinci yazılım kapsamında (SPR), seçme yöntemi olarak Rulet Çemberi Tekniği, çaprazlama yöntemi olarak PMX ve mutasyon olarak Ekleme yöntemleri kullanılmıştır. İkinci yazılım kapsamında (SLC), seçme yöntemi olarak Sıralı çaprazlama için Birleştirme metodu ve mutasyon için Atlama yöntemleri kullanılmıştır.

Projemizde KAP'ı çözmek için gerçekleştirilen Dağınık Arama Algoritması Şekil 13'teki gibidir.

```
Algoritma SS
p = Olustur_random()
p = Farklılık_metod(p)
p = Mutasyon_Atlama (p)
p = Kontrol_tabu_list(p)
Refset = tabulist (b1) + tabulist(b2)
While ( RefSet'de yeni çözüm yoksa )
{
 /***1.yazılım için (SPR)***/

    SPRx= RefSet. sec_Rulet_Cemberi()
    SPRy= RefSet. sec_Rulet_Cemberi()

    PMX (SPRx,SPRy) → z, t
     z =Mutasyon.Ekleme (z)
     t = Mutasyon.Ekleme (t)
     Eğer z.getCost() < t.getCost() && z ¢ Tabu_list ,  Tabu_ list = Refset + z
    Eğer z.getCost() > t.getCost() && t ¢ Tabu_list ,   Tabu_list = Refset + t
   RefSet.guncelle (tabu_list)
```

```
/***2.yazılım için (SLC)***/
    SLCx= RefSet. sec_sıralı()
    SLCy= RefSet. sec_sıralı()

    SLCz = Birlestirme (SLCx,  SLCy)
   SLCz = Mutasyon.Atlama(SLCz)
   Tabu_list = Refset + SLCz
   RefSet.guncelle (tabu_list)
}
```

Şekil 13— Dağınık Arama Algoritması

İlk olarak tabu liste uzunluğu (Tsize), en iyi çözüm sayısı (b1), en farklı çözüm sayısı (b2) değerleri verilir. Başta, Tabu listesinde hiçbir çözüm bulunmamaktadır. Rastgele seçilmiş başlangıç çözümleri ile populasyon oluşturulur. Farklılık metodu kullanılarak populasyondaki her bir çözüm için tabu listesi kontrolü yapılır. Eğer çözüm tabu listesinde yok ise, tabu listesine eklenir. Eğer çözüm tabu listesinde var ise, tabu listesine eklenmez. Bu işlem tabu listesinin uzunluğuna (Tsize) erişine kadar devam edecektir. Böylelikle, Tabu listesinde birbirinden faklı çözümler bulunmaktadır. Dağınık arama algoritmasının en temel özelliği, birbirinden faklı bireyler ile işlem yapılmasının sağlanmasıdır. Tabu listesindeki çözümler mutasyona uğratılır. Bu aşamada, Atlama mutasyon yöntemi kullanılmıştır. Mutasyona uğratılan çözümler içinde tabu liste kontrolü yapılır. Eğer çözüm tabu listesinde yok ise, tabu listesine eklenir. Eğer çözüm tabu listesinde var ise, tabu listesine eklenmez. Tabu listesinden b1 tane en iyi çözüm ve b2 tane en farklı çözüm alınarak Refset'e eklenir. Tabu listesindeki çözümler silinir.

Birinci yazılım kapsamında (SPR) : RefSet'den iki tane çözüm Rulet Çemberi tekniği ile seçilir (SPRx,SPRy) ve PMX yöntemi ile çaprazlamaya uğratılır. Çaprazlama sonucunda oluşan z ve t çözümleri Ekleme yöntemi ile mutasyona uğratılır. Eğer z çözümünün uygunluk değeri t çözümünün uygunluk değerinden küçük ise ve z çözümü tabu listesinin elemanı değilse, z çözümü tabu listesine eklenir. Eğer z çözümünün uygunluk değeri t çözümünün uygunluk değerinden büyük ise ve t çözümü tabu listesinin elemanı değilse, t çözümü tabu listesine eklenir. Tabu listesine RefSet'deki çözümler ile z veya t çözümleri eklenir. Tabu listesi, birbirinden farklı çözümlerden oluşmaktadır. Tabu listesinden b1 tane en iyi çözüm ve b2 tane en farklı çözüm alınarak Refset'e eklenir. Bu işlemler, RefSet'de yeni çözüm olmayıncaya kadar devam edecektir.

İkinci yazılım kapsamında (SLC): RefSet'den iki tane çözüm sıralı seçim tekniği ile seçilir (SLCx,SLCy) ve Birleştirme yöntemi ile çaprazlama'ya uğratılır. Çaprazlama sonucunda oluşan z çözümü Atlama yöntemi ile mutasyona uğratılır. Eğer z çözümü tabu listesinin elemanı değilse, z çözümü tabu listesine eklenir. Tabu listesine RefSet'deki çözümler ile z çözümü eklenir. Tabu listesi, birbirinden farklı çözümlerden oluşmaktadır. Tabu listesinden b1 tane en iyi çözüm ve b2 tane en farklı çözüm alınarak Refset'e eklenir. Bu işlemler, RefSet'de yeni çözüm olmayıncaya kadar devam edecektir.

3.4.4. GRASP

GRASP algoritmasında kullanılan teknikler aşağıda sunulmuştur.

3.4.4.1. Rulet-Çemberi Seçim Tekniği:

Tanımı önceki bölümlerde yapılmıştır.

3.4.4.2. Mutasyon Yöntemleri:

Kullanılan iki tane mutasyon yöntemi vardır. Bunlar, atlama (swap) ve ekleme (insert)'dür. Tanımları önceki bölümlerde yapılmıştır.

3.4.4.3. RCL (Kısıtlı aday) Listesi:

Kısıtlı aday çözümlerin işlem görmeleri için kullanılan bir listedir.

Projemizde KAP'ı çözmek için gerçekleştirilen GRASP Algoritması Şekil 14'deki gibidir.

```
Algoritma GRASP
Baslangıs_populasyonu_yarat(   )
oldv ← pop.getMin.getFitness()
for ( i < maxi)
{
     RCL_list_olustur(for(i =0;i<pop.size/2 ;i++) { s = sec_Rulet_Cemberi()
                                                    RCL.ekle(s)
                                                   pop.sil (s)  } )
    Cozumleri_guncelle( for(int k=0;k<RCL.size();k++){    uprs.ekle()
                                                          upra = uprs;
                                                         uprm = uprs;
                                                         uprs.Mutasyon_Atlama(mp)
                                                          uprm.Mutasyon_Ekleme(mp)

                                      Eğer (uprs.getFitness() < uprm.getFitness()) {gbest =
uprs;
                                                                    değilse gbest =
uprm}
                                       Eğer (gbestv < upra.getFitness()){gestv = gestv;
                                                                 değilse  gestv = upra ve
                                                                       pop.ekle(upra); }
                                      Eğer (gestv == uprs.getFitness( pop.ekle(uprs);}
                                      Eğer (gestv == uprm.getFitness()) { pop.ekle(uprm);}

                                        return bestv=pop.getMin.getFitness();
   }
   Eğer (oldv < bestv) , en iyi çözüm oldv, değilse en iyi çözüm bestv
}
```

Şekil 14— GRASP algoritması

Maximum iterasyon sayısı (maxi) ve çözüm sayısı (seed) değerleri verildikten sonra ilk olarak, başlangıç populasyonu yaratılır, bunun için çözüm sayısı (seed)'in iki katı kadar populasyona çözümler eklenir. Populayondaki bireylerin uygunluk değeri hesaplanır ve en düşük uygunluk değerine sahip olan çözüm 'oldv' olarak atanır. Maximum iterasyon sayısına ulaşıncaya kadar aşağıdaki işlemlere devam edilir. Populasyon uzunluğunun yarısı kadar çözüm Rulet-Çemberi tekniği ile seçilir ve çözümler (upra) *RCL* list'e eklenir. Seçilen

çözümler populasyondan çıkartılır. *RCL* list uzunluğu kadar, çözümler hem Ekleme (uprm) hem de Atlama (uprs) mutasyon yöntemi uygulanır. Her çözümün kendisi, Ekleme ve Atlama yöntemiyle elde edilen değerler karşılaştırılır ve bu üçünden en iyisi populasyona eklenir. Populasyondaki çözümlerin uygunluk değeri hesaplanır ve en düşük uygunluk değerine sahip olan çözüm 'bestv' olarak atanır. Eğer oldv, bestv'den küçük ise, en iyi çözüm oldv, değil ise en iyi çözüm bestv olarak atanır.

3.5. Esas problem için etkin sezgisel yaklaşımların geliştirilmesi

Hatırlatmak gerekirse proje önerisinde çip saçıcı adı verilen dizgi makinelerinde de ortaya çıkan problemlerin eniyilenmesi için çalışmalar yapılacağı ifade edilmişti. Çip saçıcılarda üç ayrı problem çözülmeyi beklemektedir. Bunlar besleyici düzeni, parça dizgi sırasının belirlenmesi ve her ikisinin beraberce eniyilenmesi. Bunların nasıl ortaya çıktığı proje önerisinde detaylıca ifade edildiğinden tekrar değinilmeyecektir.

Parça dizgi sırasının belirlenmesi problemi projemizin önceki dönemlerinde incelenen çip yerleştirici makinesinde ortaya çıkan ve projemizde formülasyonunu verdiğimiz SDSSP problemine dönüşmektedir. Dolayısıyla SDSSP için geliştirilen yöntemler çip saçıcıların işlemlerinin eniyilenmesi için de uygulanabilir.

Öte yandan besleyici düzeni problemi tam olarak karesel atama problemine (KAP) dönüşür. Bölüm 3.4'de belirtildiği gibi KAP için yazındaki meta sezgisellerden dört tanesi uygulanarak eniyilemesine çalışılmıştır.

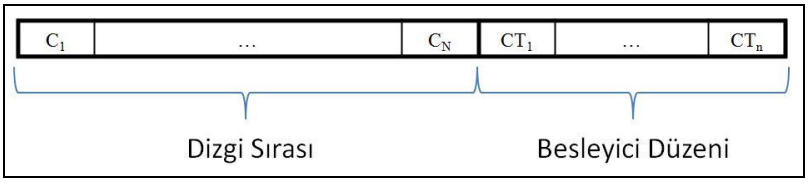
Üçüncü olarak her iki problemin beraberce eniyilemesi problemi öncekilerden daha da önemli bir problemdir. Bunun nedeni, bir problemin çözümünün diğer problemin çözümüne olan çok büyük etkisidir.

Bu etkiyi basitçe açıklayacak olursak, örneğin, verilen bir BDK dizgi işleminde, parça dizgi sırasının belirlenmesi problemi eniyiye yakın bir şekilde çözülmüş olsun. Taretin bir adım dönmesinden daha kısa zaman içinde taşıyıcı tabla BDK'yı bir sonraki parçanın dizgi yeri için hizalar. Hatırlayalım ki, bu işlerle beraber yapılması gereken bir diğer iş, besleyici düzeneğinin bir sonraki alınacak parça türüne ait hücreyi alım yerindeki kafanın altına getirmesidir. Bu iş için en kötü senaryo besleyici düzeneğinin bir uçtan diğer uca hareketi olabilir. Bu ise, yan problemlerin ayrı ayrı enyilenmesi durumunda kolaylıkla ortaya çıkabilecek bir durumdur. Bu durumda parça dizgi sırasının belirlenmesi problemi ne kadar eniyilenmiş olursa olsun, besleyici düzen probleminin çözümü bunu kötü hale getirmiştir.

Benzer şekilde besleyici düzeni problemi eniyiye çok yakın şekilde çözülmüş olsun. Yani besleyici düzeneğinin hareketleri toplamda en az hareketi vermekte olsun. Fakat, besleyici düzeneği bir hücreden diğerine hareket ederken yine aynı zamanda yapılması gereken işlerden biri de taşıyıcı tablanın BDK'yı bir sonraki dizgi işlemi için hizalamasıdır. Bu ise, BDK'nın bir uçtan diğer uca hareket ettirilmesini gerektiriyor olabilir. Bu durumda da besleyici düzen problemi ne kadar eniyilenmiş olursa olsun, parça dizgi sırasının belirlenmesi probleminin çözümü bunu kötü hale getirmiştir.

Bu örneklerle, iki yan problemin beraberce eniyilenmesi dediğimiz esas problem çözümünün gerekliliği rahatlıkla görülebilir.

Esas problem için ileri sürülecek bir çözüm önerisi aslında iki parçadan oluşmaktadır. Bunlar dizgi sırasını ve besleyici düzeni ifade eden iki ayrı yapıdır. Bu iki ayrı yapı aslında sırasıyla SDSSP'nin ve KAP'nin çözümlerini ifade etmektedir (Şekil 15).



Şekil 15—Esas problem için çözüm ifadesi

Esas problemin küçük örneklerde bile kesin çözümünü bulmak çok zaman almaktadır. Bu sonuç SDSSP için yaptığımız kesin çözüm yöntemleri çalışmalarında rahatlıkla görülebilir. Kaldı ki esas problemin çözümünde SDSSP'ne ilaveten KAP'nin çözümü de bulunmalıdır. Dolayısıyla esas problemin çözümünde etkin sezgisel yöntemler kullanılmalıdır.

Esas problemin eniyilenmesi için tasarladığımız bazı yöntemler aşağıdaki gibidir.

1. Tekrarlamalı bir yöntemle çözülmeye çalışılması
2. Birleşik bir çözüm olarak ifade edilerek çözülmeye çalışılması

Tekrarlamalı yöntemi şu şekilde açıklayabiliriz: KAP için rastgele bir çözümle başla ve bunu kullanarak toplam dizgi zamanını enazlayacak şekilde SDSSP için bir çözüm önerisi bul. Sonrasında SDSSP için bulunan çözümü kullanarak toplam dizgi zamanını enazlayacak şekilde KAP için yeni bir çözüm üret. Bu tekrarlamalı süreç belli bir sayı kadar ya da toplam dizgi zamanında iyileştirme olmayana dek devam edebilir.

Tekrarlamalı yöntem uygulanırken SDSSP ve KAP için uygulayacağımız yöntemler projenin bu safhasına kadar geliştirdiğimiz ve belirlediğimiz yöntemler olacaktır.

Esas problemin eniyilenmesinde uygulanabilecek ikinci bir yöntem de çözümlerin birleşik olarak ifade edilerek ikisinin aynı anda eniyilenmesini sağlamaktır. Böyle zor bir problemin eniyilenmesi için meta-sezgisellerden yararlanılabilir. Bu kapsamda belirlenen metasezgiseller Benzetimli Tavlama, Genetik Algoritma ve Yapay Arı Kolonisidir.

3.5.1. Esas Problem için uygulanan metasezgiseller

Benzetimli Tavlama metasezgiselinin KAP'indeki performansından dolayı esas problem için uygulanmasını da tercih ettik.

Benzetimli Tavlama algoritması yeni bir çözüme (komşu çözüm) mevcut çözümdeki ikili yer değiştirmelerle ulaşır. Benzetimli tavlamada komşu çözüm sadece bir iyileştirme durumunda değil aynı zamanda (gittikçe düşen bir ihtimalle) kötüleşme durumunda da yeni mevcut çözüm olarak kabul edilebilir. Böylelikle yerel optimumlardan kaçma özelliğine sahiptir. Kötü bir çözümün kabul edilme ihtimali $e^{-\Delta/T}$ ile hesaplanır. Burada Δ mevcut çözümle yeni çözümün uygunluk değerlerinin farkı ve T ise demir tavlama olayından esinlenerek tanımlanan zaman ilerledikçe azalan sıcaklığı ifade eder.

Benzetimli Tavlama metasezgiselin tasarlama şeklimiz 2. rapor döneminde açıkça belirtilmişti. Yeni komşu üretme dışında bütün özellikleriyle aynı algoritma kullanılmıştır.

Hatırlamak gerekirse bu tasarımda algoritmamızın dört parametresi vardı ve bunların tanımı şu şekildeydi.

T: başlangıç derecesi,
R: her bir derece için iterasyon sayısı,
noi: maximum iterasyon sayısı,
a: derecenin azalma oranı ve
b: iterasyonun artma oranı
Bu parametreler için kullandığımız değer kümeleri ise şu şekildedir.
T={100,1000}
R={5,20}
a={1.1,1.5}
b={1.1,1.5}
N bir BDK'daki parça sayısını, n ise parça tipi sayısını ifade etmektedir.

Yüksek bir *T* değeri daha fazla sayıda kötü çözümlerin tercih edilmesini sağlar. Yüksek bir *R* değeri daha geç bir soğumaya neden olacağından daha fazla sayıda kötü çözümlerin tercihine sebep olur. Yüksek bir *a* değeri hızlı bir soğumaya sebep olacak ve kötü çözümlerin tercih ihtimalini azaltacaktır. Yüksek bir *b* değeri ise *R*'nin artış hızını artıracak ve daha geç bir soğumaya neden olarak kötü çözümlerin tercih ihtimalini artıracaktır.

Genetik algoritma tasarımında kullandığımız yöntemler ve parametreler KAP için tasarladığımızla aynıdır. Hatırlamak gerekirse seçim yöntemi olarak rulet çemberi, çaprazlama yöntemi olarak PMX, mutasyon yöntemi olarak ekleme mutasyonu kullanılmıştır. Sonraki neslin oluşturulması ise en iyi olanların seçimi kullanılmıştır. Kullanılan değerler ise iki bireyin çaprazlamaya uğrama ihtimali 0.75, mutasyona uğrama ihtimali 0.05, nesil sayısı (*nog*) 40 ve 400, çözüm sayısı (*noi*) 50, 100 ve 200 olarak belirlenmiştir.

Esas problemin çözümünde kullanılan bir diğer algoritma ise yapay arı kolonisi algoritmasıdır. Yapay arı kolonisi algoritması Karaboğa tarafından öne sürülen nispeten daha yeni bir metasezgiseldir (Karaboga ve Basturk, 2007). Bildiğimiz kadarıyla yapay arı kolonisi algoritması ayrık (discrete) problemlerde hiç uygulanmamıştır.

Yapay arı kolonisi algoritması arıların doğal ortamda yemek kaynaklarını bulma yöntemlerinden esinlenerek oluşturulmuştur. Bir arı kolonisinde, arılar daha zengin yemek kaynaklarını bulmak için birbiriyle işbirliği yaparak arama yaparlar. Bir arı kolonisinde üç tip arı vardır; işçi arı, gözcü arı ve keşif arıları. İşçi arılar bulunan yemek kaynaklarının kalitesini gözcü arılara bildirmekle sorumludur. Bunu kaynaktan döndükten sonra gözcü arılar önünde çeşitli danslar ederek ifade ederler. Kovanda bekleyen gözcü arılar izledikleri dansa göre bir arının peşinden giderek belirtilen kaynaktan yemek getirirler. Belirtilen kaynakta yemek sonlandıktan sonra kaşif arılar devreye girer ve yeni yemek kaynakları bulurlar.

Tasarladığımız yapay arı kolonisi algoritmasında *cs* tane arı oluşturulur. Bunların yarısı işçi diğer yarısı ise gözcü arılardır. Algoritma *cs*/2 tane rassal çözüm bularak başlar. İşçi arıları temsil eden bu çözümlerin kalitesi ölçüsünde ve aynı zamanda ihtimale bağlı olarak bu çözümlerin komşu çözümleri gözcü arılar tarafından elde edilir. Her elde edilen yeni çözüm eğer mevcut en iyi çözümden daha iyiyse yeni en iyi çözüm olarak saklanır. Mevcut bir çözümün gözcü arılar tarafından komşuları elde edilirken eğer peşpeşe belli bir sayı defa (*limit*) daha iyisi elde edilemiyorsa tamamen rassal olarak oluşturulan yeni bir çözümden

tekrar başlanır. Bu işlem önceden belirlenen bir sayı (*döngü*) kadar tekrar edilir. Adım adım ifade edecek olursak, yapay arı kolonisi aşağıdaki gibidir.

1.İlk çözümler rassal olarak oluşturulur.
2. Mevcut çözümlerin kalitesine göre her bir çözümün komşu çözümleri araştırılır
3. Belli bir sayı (*limit*) defa komşu çözümü aranmasına rağmen daha iyi bir komşu bulunamamışsa bu çözüm yeni rassal bir çözümle değiştirilir.
4. En iyi çözüm saklanır.
5. 2-4 adımları belli bir sayı (*döngü*) kadar tekrar edilir.

Yapay arı kolonisi algoritmasında kullanılan parameteler için belirlediğimiz değerler şu şekildedir: *cs*=10,20,50; *limit*=100,1000,10000; *döngü*=100,1000,10000.

Yukarıda bahsi geçen metasezgisellerin esas probleme uygulanmasında dikkat edilmesi gereken en önemli nokta komşu çözümlerin nasıl oluşturulacağıdır. Hatırlanmalıdır ki esas problemin çözümleri dizgi sırası ve besleyici düzeni altçözümlerinden oluşmaktadır. Mevcut çözümün komşu bir çözümünü elde etmek için ya sadece dizgi sırasında bir değişiklik, ya sadece besleyici düzende bir değişiklik, ya sırasıyla önce biri sonra diğerinde değişiklik ya da aynı anda bir değişiklik yapılabilir. Bu bağlamda aşağıdaki 5 beş farklı yöntemi geliştirdik.

Yeni bir çözümü elde etmek için;

1. Her ikisini de eşit şans vererek mevcut çözümün hangi alt çözümünde değişiklik yapılacağına karar vermek.
2. Her ikisini de ağırlıklı şans vererek (parça sayısı ve parça tipi sayılarının kareleriyle doğru orantılı olarak) mevcut çözümün hangi alt çözümünde değişiklik yapılacağına karar vermek.
3. Alt çözümlerde sırayla değişiklik yapmak.
4. Her iki altçözümde de aynı anda değişiklik yapmak.
5. İlk olarak parça sayısının karesi kadar sayıda yeni çözümü dizgi sırasında değişiklik yaparak, sonra da parça tipi sayısının karesi kadar yeni çözümü besleyici düzeninde değişiklik yaparak elde etmek ve bu işlemi algoritma sonlanana kadar tekrarlamak.

3.5.2. Esas Problem için gerçekleştirilen tekrarlamalı yöntem

Tekrarlamalı yöntemi şu şekilde açıklayabiliriz: KAP için rastgele bir çözümle başla ve bunu kullanarak toplam dizgi zamanını enazlayacak şekilde SDSSP için bir çözüm önerisi bul. Sonrasında SDSSP için bulunan çözümü kullanarak ve bir önceki adımdaki besleyici düzeni çözümünden başlayarak yeni bir KAP çözümü üret. Bu tekrarlamalı süreci belli bir sayı kadar devam ettir.

Tekrarlamalı yöntem uygulanırken SDSSP ve KAP için uygulayacağımız yöntemler projenin bu safhasına kadar geliştirdiğimiz ve belirlediğimiz yöntemler olacaktır.

SDSSP için tasarladığımız İlk Noktayı Toplam Masrafa göre Değistir Yöntemi (İNTMDY), Gelişmiş Sapakları Ertele (GSE) ve RRTLEM yöntemleri bu aşamada kullanılabilir. Bu yöntemler iyileştirme yöntemleriydi ve ilk çözümü ATMA algoritmasıyla elde ediyorduk. Öte yandan ikinci dönemde yaptığımız çalışmalarda BDK dizgisi konusunda ortaya çıkan gerçek KAP örnekleri için bulunan en iyi metasezgiselin benzetimli tavlama olduğunu görmüştük.

Tekrarlamalı yöntemde, rassal oluşturulan bir besleyici düzeni için ATMA, İNTMDY, RRTLEM ve GSE yöntemleri kullanılarak dizgi sırası elde edilir. Elde edilen dizgi sırası kullanılarak benzetimli tavlama yöntemiyle besleyici düzeni iyileştirilmeye çalışılır. Yeni besleyici düzeni kullanılarak dizgi sırası yukarıda adı geçen yöntemlerle yeniden kurularak en iyilenir. Bu işlem belli bir sayı kadar devam eder.

4. Bulgular ve Tartışma

Bu bölümde tanımlanan problemler için geliştirilen çözüm yöntemlerinin bilgisayısal çalışmaları detaylıca verilmiştir.

4.1.SDSSP için kesin çözüm yöntemleri

4.1.1. Çözüm

Problem ile ilgili istatistiksel veri GAMS platformunda DICOPT çözücüsü ile üretilmiştir. Tablo 1 bize bir önceki adımda verilen formülasyon ile üretilen bilgileri göstermektedir. Burada doğrusal olmayan N-Z bize modelde doğrusal olmayan matris girişlerini göstermektedir. Bu bilgiler ile birlikte problemin büyüklüğü ve bu büyüklüğün etkileri analiz edilebilir. Bu verilere ek olarak, problemlere ait kod uzunlukları (code length), diğer bir değişle Doğrusal olmayan problem karmaşıklıkları tablolarda gösterilmiştir. Doğrusal olmayan problemlerde problemin karmaşıklığı probleme etki eden matris girişleri ile aynı olmayabilir. Örnek vermek gerekirse, x*y probleminin doğrusal olmayan problem karmaşıklığı, üssel(x*y)'den (exp x*y) daha azdır. Ve doğrusal olmayan matris girişleri ise 1'dir. GAMS platformu karmaşıklığı daha iyi ifade etmek ve örnekte ifade edilen problemlerin karmaşıklıkları arasındaki farkı göstermek amacıyla istatistiksel bilgileri arasında, kod uzunluğu (code length), diğer bir deyişle Doğrusal olmayan problem karmaşıklığını da göstermektedir. Bu nedenle doğrusal olmayan problem karmaşıklığı, problemin karmaşıklığı ile ilgili bilgiyi daha net vermektedir.

Tablo 1— GAMS Platformunda üretilen istatistiksel bilgiler

Matematiksel Model İstatistiği			
Denklem Bloğu	11	Denklem Sayısı	1,165
Değişken Bloğu	4	Değişken Sayısı	397
Sıfır Olmayan Değişken	6, 265	Doğrusal olmayan Matris	4,320
Tam Sayılı Değişken	216		

GAMS'de problemin çözüm özeti aşağıdaki gibidir:

```
                S O L V E      S U M M A R Y

     MODEL   sdtsp                OBJECTIVE  obj
     TYPE    MINLP                DIRECTION  MINIMIZE
     SOLVER  DICOPT               FROM LINE  225

**** SOLVER STATUS     1 Normal Completion
**** MODEL STATUS      8 Integer Solution
**** OBJECTIVE VALUE                2.2000

 RESOURCE USAGE, LIMIT          0.812      2222.000
 ITERATION COUNT, LIMIT       493    2000000000
 EVALUATION ERRORS              0             0
```

Yukarıdaki çözüm özeti bize 6 parça, 2 parça grup ve 3 parça tipi bulunan bir baskı devrenin 5 besleme hücresi ile birlikte çözüldüğü zamanki bilgileri vermektedir. Buna göre11 denklem bloğu, 1165 denklem ve 397 değişken üretildiği görülür. 0.812 saniyede çözüm olan 2.2'ye ulaşıldığı görülür. Bir sonraki bölümde matematiksel modelin sayısal analizi problem boyutuna bağlı olarak anlatılmaktadır.

4.1.2. Sayısal Analiz

Bu bölümde parça dizgi sırasına ait matematiksel modele ait sayısal veriler incelenmiştir. Parça sayısı, parça tipi ve besleme hücre sayıları listelenmiş ve buna bağlı olarak problemi çözmek için gereken zaman, problemin karmaşıklı hesaplanmış ve çözmek için gerekli donanım yapısı incelenmiştir.

Parça dizgi sırası probleminin büyüklüğü, parça sayısı, parça tipi ve grubunun sayısı olarak düşünüldüğü zaman, problemin matematiksel karmaşıklığı problemin büyüklüğüne bağlı olarak değişiklik göstermektedir. Diğer bir değişle, probleme ait kısıt kümesi ve değişken sayısı karmaşıklığı ifade etmektedir. Parça tipi, parça grubu, parça sayısı arttığı zaman problemin çözümü zorlaşmakta ve çözüme ulaşmak için gereken zaman da artmaktadır. Bu nedenle modelin sayısal karmaşıklığı problemin büyüklüğü ile bağlantılıdır. Bunun yanında problemde verilen besleme hücresi sayısı modelin büyüklüğü ile ilişkilidir (modelde bulunan 4-5 arasındaki kısıt kümesindeki kısıt sayısı artmakta veya azalmaktadır).

4.1.2.1. Sayısal Karmaşıklık

Matematiksel tanımın karmaşıklığını incelemek için, ilk olarak parça sayısının model üzerindeki etkisi incelenmektedir.

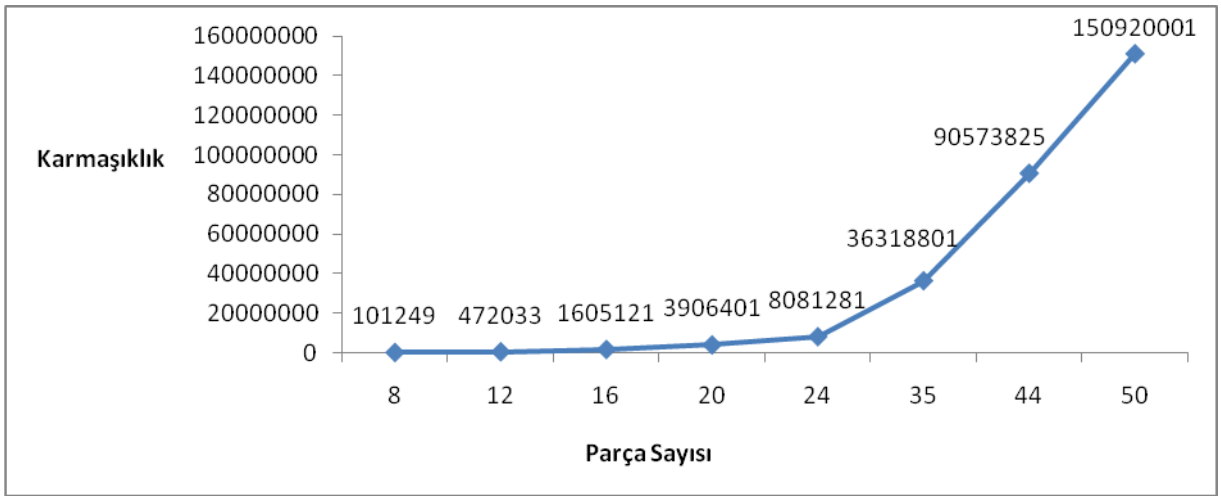
Tablo 2'de görüldüğü gibi problemin karmaşıklığı ve üretilen denklem sayısı parça sayısına göre gittikçe artmaktadır (üstel).

Tablo 2— Parça sayısına göre model istatistiği (Besleme hücresi sayısı = 5, Alım yapılmayan bölgedeki hücre sayısı (npz) =2)

Parça Sayısı	8	12	16	20	24	35	44	50
Denklem Sayısı	2,833	9,817	23,585	46,441	80,689	252,421	503,449	740,101
Doğrusal olmayan matris	16,128	76,032	261,120	638,400	1,324,800	5,997,600	14,984,640	24,990,000

Değişken	961	3,313	7,937	15,601	27,073	84,526	168,433	740,101
Tam Sayılı Değişken	512	1,728	4,096	8,000	13,824	42,875	85,184	125,000
Doğrusal olmayan problem karmaşıklığı	101,249	472,033	1,605,121	3,906,401	8,081,281	36,318,801	90,573,825	150,920,001

Doğrusal olmayan karmaşıklık seviyesi Şekil 16'da gösterilmektedir ve parça sayısına göre gittikçe artan (üstel) durum daha belirgin bir şekilde anlaşılmaktadır.



Şekil 16— Parça sayısına bağlı karmaşıklık seviyesi

Tablo 2'de görüldüğü gibi parça sayısını arttırdığımız zaman probleme ait denklem sayısı da artmaktadır. Bu da daha fazla bellek tüketimine sebep olmaktadır. Tablo 3'te modelin parça sayısına bağlı olarak ihtiyaç duyduğu bellek miktarı yaklaşık olarak gösterilmektedir. Problem çözülürken gözlenen maksimum bellek miktarı aşağıda gösterilmiştir. Parça sayısı 50'ye çıktığı zaman, GAMS platformunun ihtiyaç duyduğu bellek miktarı 16 GB'a kadar çıkmaktadır

Tablo 3— Parça sayındaki değişime göre ihtiyaç duyulan bellek miktarı(yaklaşık değerler alınmıştır)

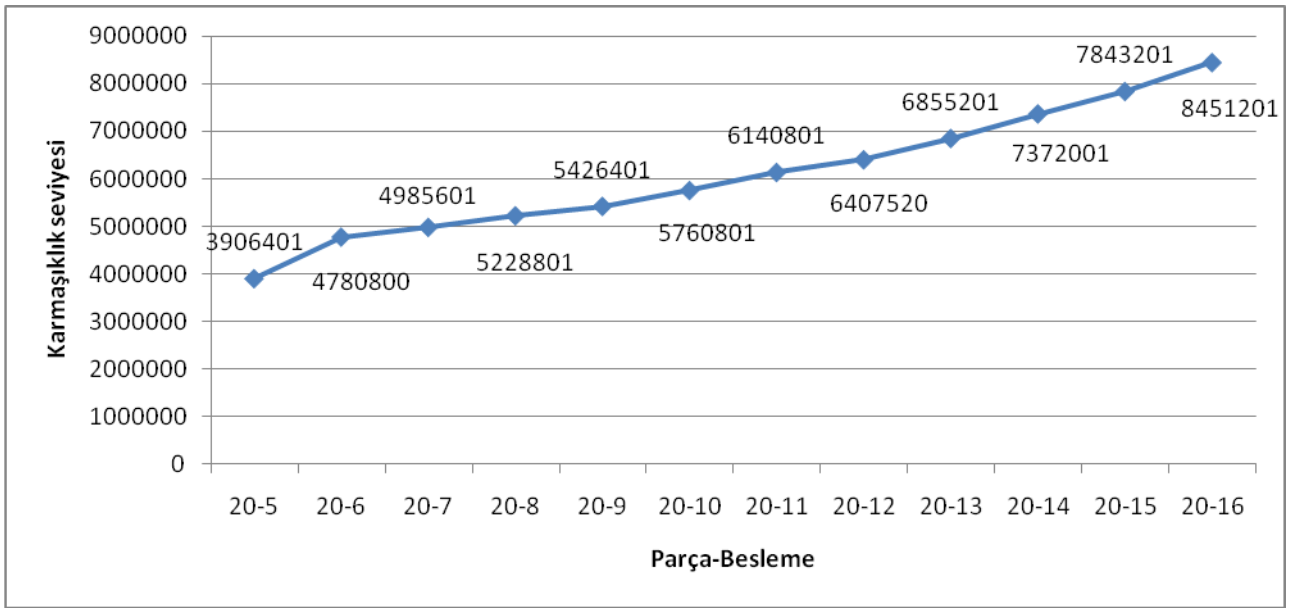
Parça Sayısı	15	20	25	35	40
Kullanılan Bellek (mb)	700	1,000	1,600	3,200	4,300

Bir sonraki adımda besleme hücresi sayısı arttırılmış olup modelimizin karmaşıklığı üzerindeki etkisi incelenmiştir. Tablo 4'te artan besleme hücresi ve 20 parçalı problemlerin karmaşıklık istatistikleri gösterilmektedir. Şekil 17'ye bakılarak karmaşıklıkta meydana gelen artış eğiminin parça sayısı arttırıldığı zamanki eğimden daha küçük olduğu görülür. Tablo 4 ve Şekil 17 besleme hücresine göre modelin karmaşıklığını göstermektedir.

Tablo 4— Besleme hücresi sayısına göre matematiksel tanım istatistiği (npz = 2)

Parça - Besleme hücresi	20-5	20-6	20-7	20-8	20-9	20-10	20-16

Denklem Sayısı	46,441	54,041	61,641	69,241	76,841	84,441	130,041
Doğrusal olmayan matris	638,400	782,800	813,200	851,200	881,600	934,800	1,368,000
Değişken	15,601	15,601	15,601	15,601	15,601	15,601	15,601
Tam Sayılı Değişken	8,000	8,000	8,000	8,000	8,000	8,000	8,000
Doğrusal olmayan problem karmaşıklığı	3,906,401	4,788,000	4,985,601	5,228,801	5,426,401	5,760,801	8,451,201



Şekil 17— Besleme hücre sayısına bağlı karmaşıklık seviyesi

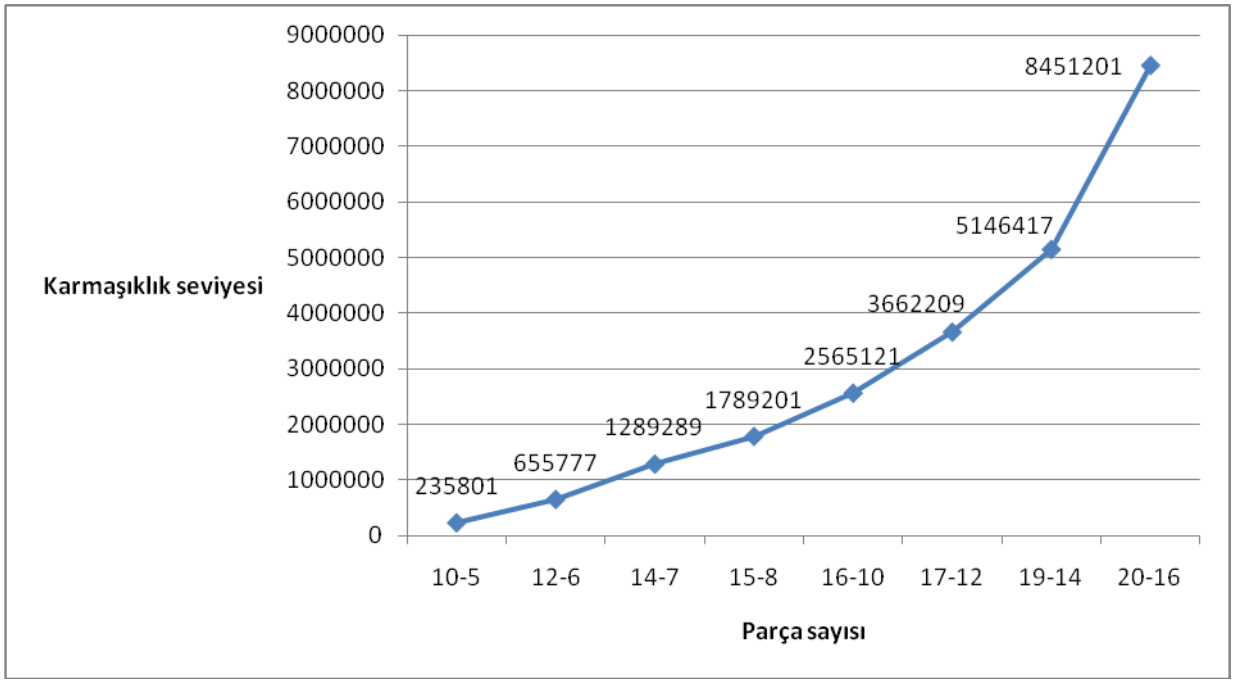
Önceki adımlarda parça ve besleme hücresi sayısı birbirinden bağımsız olarak arttırılmış olup model üzerindeki etkisi incelenmiştir. Bu adımda ise parça ve besleme hücresi sayısı aynı anda arttırılmıştır. Tablo 5'deki sonuçlara bakıldığı zaman karmaşıklık seviyesindeki artışın daha belirgin olduğu gözlenmektedir.

Tablo 5— Parça Sayısı ve Besleme hücresin aynı anda artışı ile oluşan sayısal bilgi (npz=2)

Parça-Besleme Hücresi Sayısı	10-5	12-6	14-7	15-8	16-10	17-12	19-14	20-16
Denklem Sayısı	5,621	11,401	20,805	28,831	42,785	60,725	98,231	130,041
Doğrusal olmayan matris	37,80	106,1	20,129	289,80	414,72	591,87	831,74	1,368,0

	0	28		0	0	2	4	00
Değişken	1,901	3,313	5,293	6,526	7,937	9,538	6,859	15,601
Tam Sayılı Değişken	1,000	1,728	2,744	3,375	4,096	4,913	6,859	8,000
Doğrusal olmayan problem karmaşıklığı	235,801	655,777	1,289,289	1,789,201	2,565,121	3,662,209	5,146,417	8,451,201

Şekil 18'te parça sayısı ve besleme hücresi sayısını aynı anda arttırdığımızdaki sonuçları göstermektedir.



Şekil 18—Parça sayısı ve besleme hücresinin eş zamanlı artışıyla meydana gelen karmaşıklık

4.1.2.2. Koşma Zamanı Sonuçları

Yukarıdaki matematiksel tanım doğrusal olmayan tamsayılı programlama problemidir. Problem içerisinde tam sayılı değişkenler model içerisinde doğrusal olmayan ikili değişkenler şeklindedir. Bu nedenle matematiksel problemi çözecek olan algoritmalar ve çözücüler bu iki durumu da karşılamak zorundadır.

Problemi çözmek için DICOPT çözücüsü kullanılmıştır çünkü modelimizi çözen diğer çözücüler DICOPT kadar başarılı olmamıştır.

Problemin çözümünde DICOPT ile karşılaştırılan diğer bir çözücü BONMIN (Basic Open-source Nonlinear Mixed INteger programming)'dir. BONMIN MINLP problemlerin çözümünde kullanılan C++ ile yazılmış açık kaynak kodlu bir çözücüdür. Bu çözücüde bulunan algoritmalar arasında dal-sınır, dışsal yaklaşım algoritmaları bulunur. Problem için tanımlanan matematiksel tanım, COIN-OR BONMIN çözücüsü ile çözüldüğü zaman,

Zamansal sonuçlarda büyük oranda artış olmuştur. DICOPT ile karşılaştırıldığı zaman sonuçlar Tablo 6'da gösterilmiştir.

Tablo 6— Dicopt ile BONMIN çözücülerinin zamansal karşılaştırılması (Besleme hücresi=5, npz=2)

Parça Sayısı	5	10	15
DICOPT Çözme Zamanı (Saniye)	1.062	5.406	39.547
BONMIN Çözme Zamanı(Saniye)	81.766	1,014.672	1,561.047

Matematiksel modelle birlikte optimum çözümün bulunmasına rağmen problemi çözmek için gereken zaman üstel bir grafik çizmektedir. Buna bağlı olarak problemlerin ihtiyaç duyduğu bellek miktarı da arttığından, GAMS platformu belirli bir sayının üstündeki problemleri başarılı bir şekilde çözememektedir. Sonuç olarak matematiksel tanımı çözerken iki engelle karşılaşılmaktadır. Bunlar;

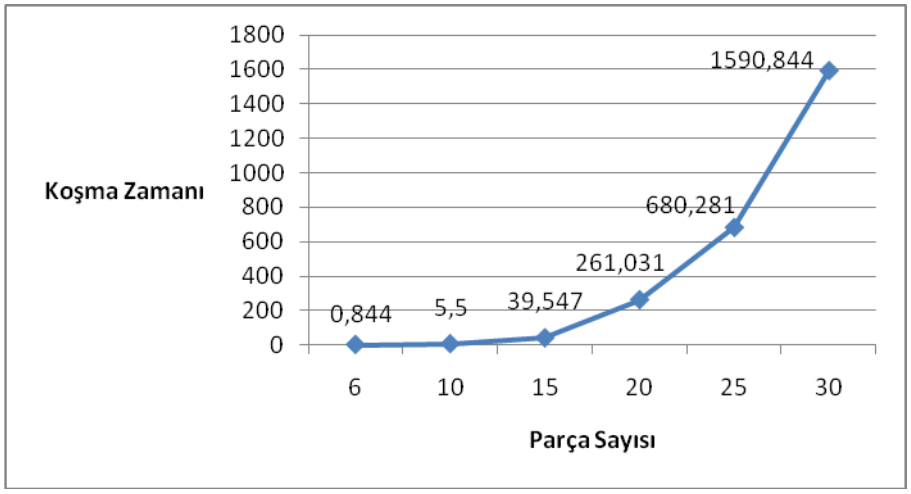
- Gittikçe artan hesaplama zamanı
- Gittikçe artan bellek miktarı ihtiyacı

Değişen parça sayısına göre problemin çözümünde ihtiyaç duyulan zaman Şekil 19'da gösterilmektedir. Ayrıca bu zaman matematiksel problemin zorluğuna göre artıp azalmaktadır.

GAMS'de verilen problemin zorluk derecesi kolay olduğu zaman, çözmek için gerekli olan zaman kısalmaktadır. Fakat rastgele oluşturulan ve zorluk derecesi yüksek olan problemlerde bu zaman çok yükselmektedir. Tablo 7 kolay problemlerde GAMS tarafından kullanılan zamanı göstermektedir. Kolay problemin tanımını yapacak olursak, "aynı gruptaki ve tipteki parçaların bir sıra şeklinde birbiri ardına koyulması" şeklinde söylenebilir. Diğer bir değişle, parçalarımızı, c1, c2, c3, c4, c5, c6 olarak düşünecek olursak ve bu parçaları birbiri peşine sıralarsak kolay problemler oluşturmuş oluruz.

Tablo 7— Parça sayısına göre, kolay problemlerde çalışma zamanı(Besleme hücresi=5, npz=2)

Parça Sayısı	6	10	15	20	25	30
Besleme Hücresi	5	5	5	5	5	5
Parça Tipi Sayısı	3	3	3	3	3	3
Çalışma süresi(saniye)	0.844	5.500	39.547	261.031	680.281	1,590.844



Şekil 19— Parça sayısına göre, kolay problemlerde çalışma zamanı

Konumları rastgele oluşturulan problemlerde ise GAMS tarafından kullanılan zaman Tablo 8'de gösterilmektedir. Problemi çözerken tekrarlama sayısı arttığından, çözmek için gerekli olan zaman da artan bir şekilde artmaktadır.

Tablo 8— Rastgele oluşturulan problemlerde çözme zamanı ve optimum sonuçlar (Besleme hücresi sayısı=5 , npz=2)

Parça Sayısı	6	10	15	20	25	30
Besleme Hücresi	5	5	5	5	5	5
Parça Tipi Sayısı	3	3	3	3	3	3
Çalışma süresi(saniye)	0.859	13.000	101.122	1,527.281	52,627.320	93,895.064
En iyi Çözüm	4.06	5.52	7.17	8.16	9.740	10.740

Yukarıdaki tabloya bakılarak çözüm zamanının 30 parçalık bir baskı devre için 26 saate kadar çıktığı görülebilir.

GAMS Platformunda, problemin boyutunu 46 parçaya kadar arttırdığımızda, çözücüleri başlatmak için gereken bellek miktarı yetersiz olduğundan problemin çözümüne ulaşılamamaktadır. Tablo 9'a bakıldığı zaman, doğrusal olmayan karmaşıklık, denklem ve değişken sayısı küçük örneklere göre büyük bir artış göstermektedir. Bu da gereken bellek miktarının ve zamanın artışını olumsuz yönde etkilemektedir.

Tablo 9— Parça sayısının arttırılmasıyla oluşan model istatistiği (Besleme hücresi sayısı = 5, npz =2)

Parça Sayısı	40	44	50-Yetersiz Bellek Miktarı
Denklem	377,681	503,449	740,101
Doğrusal olmayan matris	10,233,600	14,984,640	24,990,000
Tam Sayılı Değişken	126,401	168,433	740,101
Doğrusal olmayan problem karmaşıklığı	62,025,601	90,573,825	150,920,001

4.1.2.3. Sonuç

Sonuç olarak çip parça yerleştirici makinelerde baskı sırası problemi GAMS platformu üzerinde kodlanmış olup değişik büyüklükteki problemler üzerinde matematiksel incelemeler yapılmıştır.

GAMS ile birlikte küçük boyuttaki problemlerde en iyi çözümün bulunmasına rağmen problem boyutu arttıkça üstel şekilde artan sayısal karmaşıklık ve hesaplama zamanının oluştuğu görülmektedir.

Artan karmaşıklık ile birlikte problemin çözümünde ihtiyaç duyulan bellek ihtiyacı da artmaktadır. Bu nedenle büyük boyuttaki problemler başarılı bir şekilde çözülememektedir.

4.2.SDSSP için geliştirilen sezgisel çözüm yaklaşımları

Bu alt bölümde SDSSP için geliştirdiğimiz sezgisellerin bilgisayısal olarak performans analizini sunmaktayız. Önceki bölümlerde belirtildiği üzere SDSSP tanımı çip yerleştirici makinelerin işlem özellikleri incelenirken ortaya çıkmıştı. Bu makinenin dizgi zamanının eniyilenmesi ile ilgili olarak yazında iki eser bulunmaktadır (Duman, 2007; Alkaya ve diğ., 2008). Dolayısıyla, projemizde geliştirdiğimiz SDSSP için sezgisel çözüm yöntemleri adı geçen çalışmalardaki çözüm yöntemleriyle kıyaslanmıştır.

İNDY ve İNTMDY'nin ATMA'ya kıyasla performansları aşağıdaki tabloda verilmiştir (Tablo 10). Sonuçlar göstermektedir ki İNTMDY ATMA'yı geçmektedir. İNDY ise yapısal BDK'lar için ATMA'dan daha kötü sonuçlar vermektedir. Bunun nedeni İNDY'nin kurgusunda yer alan, grup 1 parçalarının dizgi zamanına göre karar vermesidir. Halbuki grup 1 parçalarının dizgi sırasının değişmesi sonraki gruplarda yer alan parçaların dizi zamanlarını kötüleştirmiş olabilir.

Tablo 10—İNDY ve İNTMDY'nin 100 noktalı BDK'lar üzerinde performansı

	Homojen BDK			Yapısal BDK		
Sezgisel	Dizgi Zamanı	İyileştirme	Çalışma Zamanı	Dizgi Zamanı	İyileştirme	Çalışma Zamanı
ATMA	40.55	-	0.53	35.83	-	0.51
ATMA+ İNDY	39.92	1.55%	0.54	36.12	-0.79%	0.55
ATMA+ İNTMDY	38.62	4.77%	0.54	34.38	4.06%	0.55
iATMA	40.20	0.85%	0.50	34.58	3.50%	0.52
iATMA+ İNDY	40.44	0.27%	0.51	35.89	-0.15%	0.51
iATMA+ İNTMDY	39.21	3.32%	0.51	34.34	4.17%	0.50

SE ve GSE'nin ATMA'ya kıyasla performansları aşağıdaki tabloda verilmiştir (Tablo 11). Sonuçlar göstermektedir ki SE ve GSE ATMA'yı iyileştirmekte ve oldukça iyi sonuçlar vermektedirler. Çalışma zamanı olarak GSE, SE'den yaklaşık 2,5 kat daha fazla zaman almaktadır ve buna karşılık yaptığı iyileştirme SE'den çok ötede değildir.

Tablo 11—SE ve GSE'nin 100 noktalı BDK'lar üzerinde performansı

	Homojen BDK			Yapısal BDK		
Sezgisel	Dizgi Zamanı	İyileştirme	Çalışma Zamanı	Dizgi Zamanı	İyileştirme	Çalışma Zamanı
ATMA+SE	38.73	4.50%	3.88	35.01	2.30%	4.03

ATMA+GSE	38.51	5.05%	10.85	34.94	2.49%	9.58
iATMA+SE	38.76	4.42%	4.02	34.38	4.04%	3.99
iATMA+GSE	38.63	4.75%	10.35	34.36	4.11%	8.10

RRTLEM algoritmasında iki parametrenin değerlerinin belirlenmesi gerektiğini vurgulamıştık. Yaptığımız ön çalışma sonucunda *noi* parametresi için 400 ve *rate* parametresi için %1.4 değerlerinin en iyi sonuçları verdiğini tespit ettik. Bu değerlerle çalıştırdığımız RRTLEM algoritmasının ATMA'yla performans kıyaslaması Tablo 12'de verilmiştir. Bu tablodan gözlenen odur ki RRTLEM büyük iyileştirmeler sağlamakta fakat çalışma zamanı ATMA'ya kıyasla daha büyük olmaktadır.

Tablo 12—RRTLEM'nin 100 noktalı BDK'lar üzerinde performansı

	Homojen BDK			Yapısal BDK		
Sezgisel	Dizgi Zamanı	İyileştirme	Çalışma Zamanı	Dizgi Zamanı	İyileştirme	Çalışma Zamanı
ATMA+RRTLEM	38.63	4.73%	11.74	34.15	4.69%	11.37
iATMA+RRTLEM	38.95	3.96%	10.83	33.75	2.40%	10.44

4.3.KAP İçin Yapılan Çalışmalar

Bu alt bölümde KAP için yapılan bilgisayısal çalışmalar sunulmuştur.

4.3.1. Benzetimli Tavlama

(Duman ve Or, 2007)'de yer alan literatür problemlerinden sekiz tanesine ulaşılmış, üçüne ulaşılamamıştır. Bundan dolayı testler sekiz problem üzerinde yapılmıştır.

Kullanılan parametreler sırasıyla:

i. Başlangıç Derecesi (T): 100 veya 1000 değerlerinden oluşmaktadır. Test problemlerimiz göz önüne alındığında T'nin 100 olması durumunda %10 daha kötü çözümlerin kabul edilmesi olasılığı yaklaşık olarak 0.75'tir. Eğer T 1000 ise bu olasılık yaklaşık 0.05 civarında olmaktadır.
ii. Derece düşürme oranı(a): 1.1 veya 1.5 değerlerinden oluşmaktadır. T/a değeri kadar azalma işlemi gerçekleştirilir.
iii. Her Derecede iterasyon sayısı (R) : 5 veya 25 değerlerinden oluşmaktadır.
iv. Iterasyon sayısını arttırma oranı (b): 1.1 veya 1.5 değerlerinden oluşmaktadır. O sıcaklıkta R*b değeri kadar deneme işlemi gerçekleştirilir.
v. noi (iterasyon sayısı): (N^3/3) veya (N^3) değerlerinden oluşmaktadır.

Parametrelerin farklı kombinasyonlarıyla aşağıdaki tabloda görüldüğü gibi 12 farklı benzetimli tavlama sezgiseli elde ettik (Tablo 13).

Tablo 13—Benzetimli Tavlama algoritma parametreleri

İşlem Adı	Tür	Noi	T	R	a	b
H5	*BT*	N^3/ 3	100	5	1.5	1.1

H6	*BT*	N^3	100	5	1.5	1.1
H7	*BT*	N^3/ 3	100	20	1.5	1.1
H8	*BT*	N^3	100	20	1.5	1.1
H9	*BT*	N^3/ 3	100	20	1.1	1.5
H10	*BT*	N^3	100	20	1.1	1.5
H11	*BT*	N^3/ 3	1000	5	1.5	1.1
H12	*BT*	N^3	1000	5	1.5	1.1
H13	*BT*	N^3/ 3	1000	20	1.5	1.1
H14	*BT*	N^3	1000	20	1.5	1.1
H15	*BT*	N^3/ 3	1000	20	1.1	1.5
H16	*BT*	N^3	1000	20	1.1	1.5

12 sezgisel sekiz tane test probleminde test edilmiş ve her test 10 defa çalıştırılmıştır. Test sonuçları ile ilgili analizler aşağıda sunulmaktadır.

Tablo 14'te sekiz tane probleme göre En İyi Ortalama ve En İyi Çözüm sunulmaktadır.

Her test problemi için her sezgisel 10 defa çalıştırılmıştır. Dolayısıyla her test problemi için toplam 120 (12x10) koşturulma sonucu vardır. Bu 120 koşturulma arasında en iyi sonucun hangi sezgiselle başarıldığı not edilmiştir (En İyi çözüm sütunu). Ayrıca her sezgiselin her test problemindeki (10 çalışma için) ortalama performansı hesaplanmıştır. Ortalama performans açısından en iyi sonucu veren sezgisel ise En İyi Ortalama sütununda verilmiştir.

H8, H9, H15 ve H16 dışındakiler iyi performans göstermiştir. (noi) = N^3, R=20 ve T=1000 değerlerinin performansı, (noi) = N^3/3, R=5 ve T=100 değerlerinden daha iyidir.

Tablo 14— Problemlerin en iyi ortalama ve en iyi uygunluk değeri

Problem	En iyi Ortalama (değer, sezgisel)	En iyi Çözüm (değer, sezgisel)
PS11AK08-9	799,00 H5	740,00 H14
PS11AK1011	806,60 H14	750,00 H13
PS11AK12-7	818,60 H13	774,00 H13
PS11AK15-4	746,80 H7	714,00 H6,H11
PS11AK16-3	1395,60 H14	1348,00 H12
PS11AK16-4	1441,60 H10	1392,00 H6,H14
PS11AK16-5	1502,20 H10	1454,00 H6
PS11AK17N3	1156,20 H7	1076,00 H14

Tablo 15'da her sezgiselin parametreleri ve testlerde elde ettikleri başarı sayıları ifade edilmiştir. Bu sayılar Tablo 14'ten elde edilmiştir. Örneğin, H5'e En İyi Ortalama sütununda bir defa rastlanmakta ve En İyi Çözüm sütununda hiç rastlanmamaktadır. Dolayısıyla Tablo 15'da H5 için "En iyi Ortalama sayısı" sütununda 1 ve "En İyi Çözüm sayısı" sütununda 0 değeri verilmektedir. Bu değerler sezgisellerin performanslarını karşılaştırmak için önemli değerlerdir.

Bizce sezgisellerin kıyaslanmasında "En İyi Çözüm" sayısı yüksek olan, en iyi sezgiseldir. Bunun nedeni BDK planlamalarının çok sık yapılmaması ve dolayısıyla eniyileme

algoritmalarının koşum zamanlarının çok kritik olmamasıdır. Eğer koşum zamanı kritik olsaydı, ortalama performansı iyi olan yöntemler diğerlerine tercih edilebilirdi.

Tablo 15— Test sonuçlar ve kullandıkları parametreler

İşlem Adı	En İyi Ortalama sayısı	En İyi Çözüm sayısı	noi	T	R	A	b
H5	*1*	0	N^3/ 3	100	5	1.5	1.1
H6	0	3	N^3	100	5	1.5	1.1
H7	*2*	0	N^3/ 3	100	20	1.5	1.1
H8	*0*	0	N^3	100	20	1.5	1.1
H9	*0*	0	N^3/ 3	100	20	1.1	1.5
H10	*2*	0	N^3	100	20	1.1	1.5
H11	*0*	1	N^3/ 3	1000	5	1.5	1.1
H12	*0*	1	N^3	1000	5	1.5	1.1
H13	*1*	2	N^3/ 3	1000	20	1.5	1.1
H14	*2*	3	N^3	1000	20	1.5	1.1
H15	*0*	0	N^3/ 3	1000	20	1.1	1.5
H16	*0*	0	N^3	1000	20	1.1	1.5

H14 değeri ortalamada 2 kez ve uygunluk değerinde 3 kez başarılı olduğu için en iyi sezgiseldir. Bu sonuçla birlikte (noi) = N^3 , R=20,a=1.5, b=1.1 ve T=1000 değerlerinin performansı diğerlerinden daha iyidir denilebilir. Bu sonuç Duman ve Or(2007) ile de paraleldir.

4.3.2. Genetik Algoritma

Genetik Algoritma için kullanılan parametreler sırasıyla:

i. Seçim Yöntimi (S) : Birinci yazılım çalışması kapsamında: SF metodu kullanılır. İkinci yazılım çalışması kapsamında: SS metodu kullanılır.
ii. Mutasyon (MP) : 0.05 değerinden oluşmaktadır.
iii. Çaprazlama (CXP) : 0.75 değerinden oluşmaktadır.
iv. Maximum nesil sayısı (G): 40 veya 400 değerlerinden oluşmaktadır.
v. Çözüm sayısı (P) : 50, 100 veya 200 değerlerinden oluşmaktadır.

Parametrelerin farklı kombinasyonlarıyla Tablo 16'da görüldüğü gibi 12 farklı Genetik Algoritma sezgiseli elde ettik.

Tablo 16— Genetik Algoritma parametreleri

İşlem Adı	S	G	P
G1	SF	40	50
G2	SF	40	100
G3	SF	40	200
G4	SF	400	50
G5	SF	400	100

G6	SF	400	200
G7	SS	40	50
G8	SS	40	100
G9	SS	40	200
G10	SS	400	50
G11	SS	400	100
G12	SS	400	200

12 sezgisel 8 tane test probleminde test edilmiştir. Her test 10 defa çalıştırılmıştır. Test sonuçları ve ile ilgili analizler aşağıda sunulmaktadır.

Testler sonucunda 8 tane problemin hangi sezgisel algoritmada en iyi ortalama (10 koşum için) ve en iyi çözüm ile sonuçlandığı Tablo 17'de sunulmaktadır. G5 ve G6 iyi performans göstermişlerdir.

Tablo 17— Problemlerin en iyi ortalama ve en iyi uygunluk değeri (G1 – G12)

Problem	En iyi Ortalama	En iyi Çözüm
PS11AK08-9	856,20 G6	806,00 G6
PS11AK1011	849,20 G6	802,00 G5
PS11AK12-7	917,20 G6	854,00 G5
PS11AK15-4	762,80 G6	716,00 G6
PS11AK16-3	1407,60 G6	1378,00 G6
PS11AK16-4	1490,60 G6	1458,00 G5
PS11AK16-5	1593,80 G5	1514,00 G5
PS11AK17N3	1230,40 G6	1154,00 G6

12 tane sezgisel 8 tane problem için test edilmiştir. Her bir sezgiselin 8 tane problemde almış olduğu en iyi ortalama ve en iyi uygunluk değerlerinin sayısı Tablo 18'de sunulmuştur.

Tablo 18— Test sonuçlar ve kullandıkları parametreler

Algoritma Adı	En İyi Ortalamaya Sahip Olma Sayısı	En İyi Çözüme Sahip Olma Sayısı	S	G	P
G1	*0*	0	SF	40	50
G2	0	0	SF	40	100
G3	*0*	0	SF	40	200
G4	*0*	0	SF	400	50
G5	*1*	4	SF	400	100
G6	*7*	4	SF	400	200
G7	*0*	0	SS	40	50
G8	*0*	0	SS	40	100
G9	*0*	0	SS	40	200
G10	*0*	0	SS	400	50
G11	*0*	0	SS	400	100
G12	*0*	0	SS	400	200

SS yöntemini kullanan G7-G12 arasında en iyi performansı G12 göstermiştir. Fakat G1 ve G12 aralığında, G12 değerinin kötü sonuçlandığı gözlenmektedir. G6 sezgisel çözüm yötemi 7 kez en iyi ortalamaya sahip olmuş ve 4 defa da en iyi çözümü vermiştir. Dolayısıyla en iyi sezgisel çözüm yötemi G6'dır. Bir diğer ifadeyle (S) = SF, G:400 ve P:200 değerlerinin performansı diğerlerinden daha iyidir. Nesil ve populasyon sayısı ne kadar büyük olursa algoritma o kadar iyi performans göstermektedir. G12 değerinin parametreleri de (S) = SS, G:400 ve P:200'dür. G6 ile G12 değerlerine baktığımızda seçme metodları fark göstermektedir. SF seçme metodu en iyi bireyleri seçerek yeni nesil oluşturmaktadır. Kötü bireylere şans vermemektedir. SS metodu ise kötü bireylerin seçilmesine de şans verir.

4.3.3. Dağınık Arama

Dağınık Arama için kullanılan parametreler sırasıyla:

i. Tabu Listesi Uzunluğu (Tsize) : 30, 50 veya 100 değerlerinden oluşmaktadır. Tsize uzunluğu b1+b2 uzunluğundan büyük olmak zorundadır.
ii. En iyi çözümlerin sayısı (b1): 10 veya 20 değerlerinden oluşmaktadır. (Dai,G. 2008).
iii. En farklı çözümlerin sayısı (b2): 10 veya 20 değerlerinden oluşmaktadır. (Dai,G. 2008).
vi. Mutasyon (MP) : 0.05 değerinden oluşmaktadır.
vii. Çaprazlama (CXP) : 0.75 değerinden oluşmaktadır.

Parametrelerin farklı kombinasyonlarıyla Tablo 19'da görüldüğü gibi 14 farklı Dağınık arama sezgiseli elde ettik.

Tablo 19—Dağınık Arama parametreleri

Procedure	Type	Tsize	B1	b2
S1	SPR	30	10	10
S2	SPR	50	10	10
S3	SPR	50	20	10
S4	SPR	50	10	20
S5	SPR	50	20	20
S6	SPR	100	10	10
S7	SPR	100	20	20
S8	SPR	100	20	10
S9	SPR	100	10	20
S10	SCL	30	10	10
S11	SCL	50	10	10
S12	SCL	100	10	10
S13	SCL	50	20	20
S14	SCL	100	20	20

14 sezgisel 8 tane test probleminde test edildi. Her test 10 defa çalıştırılmıştır. Test sonuçları ile ilgili analizler aşağıda sunulmaktadır. Testler sonucunda 8 tane problemin hangi sezgisel algoritmada en iyi ortalama ve en iyi çözüm ile sonuçlandığı Tablo 20'de sunulmaktadır. S3, S5 ve S7 dışındakiler kötü performans göstermişlerdir.

Tablo 20—Problemlerin en iyi ortalama ve en iyi uygunluk değeri (S1 – S14)

Problem	En iyi ortalama		En iyi çözüm	
PS11AK08-9	993,2	S5	870	S5
PS11AK1011	976,6	S5	858	S5
PS11AK12-7	1028,8	S5	886	S5
PS11AK15-4	835,8	S5	772	S5
PS11AK16-3	1614,6	S5	1284	S3
PS11AK16-4	1599,2	S7	1450	S7
PS11AK16-5	1802,6	S5	1540	S5
PS11AK17N3	1494	S5	1248	S5

14 tane sezgisel 8 tane problem için test edilmiştir. Her bir sezgiselin 8 tane problemde almış olduğu en iyi ortalama ve en iyi uygunluk değerlerinin sayısı Tablo 21'de sunulmuştur.

Tablo 21—Test sonuçlar ve kullandıkları parametreler

İşlem Adı	En İyi Ortalamaya Sahip Olma Sayısı	En İy Çözüme Sahip Olma Sayısı	Tsize	b1	b2
S1	0	0	30	10	10
S2	0	0	50	10	10
S3	0	1	50	20	10
S4	0	0	50	10	20
S5	7	6	50	20	20
S6	*0*	0	100	10	10
S7	*1*	1	100	20	20
S8	*0*	0	100	20	10
S9	*0*	0	100	10	20
S10	*0*	0	30	10	10
S11	*0*	0	50	10	10
S12	*0*	0	100	10	10
S13	*0*	0	50	20	20
S14	*0*	0	100	20	20

S5 sezgiseli ortalamada 7 kez ve uygunluk değerinde 6 kez başarılı olduğu için en iyi Dağınık Arama sezgiselidir. Diğer bir deyişle Tsize = 50 , b1=20 ve b2=20 değerlerinin performansı diğerlerinden daha iyidir. Tsize değeri b1 ve b2 toplamından ne kadar büyük olursa, performans o kadar kötüleşmiştir. Tsize'ın büyük olması, b2 değerinin en kötü bireylere sahip olma şansının artmasına neden olmaktadır. SPR yönteminin SCL yöntemini kullanan işlemlerden daha iyi sonuç verdiği saptanmıştır. Birinci yazılım kapsamında (SPR – S1-S9), seçme yöntemi olarak Rulet Çemberi Tekniği, çaprazlama yöntemi olarak PMX ve mutasyon: Ekleme yöntemleri kullanılmıştır. İkinci yazılım kapsamında (SLC –S10 – S14), seçme yöntemi olarak Sıralı, çaprazlama: Birleştirme metodu ve mutasyon için Atlama yöntemleri kullanılmıştır. SPR yöntemine göre, rulet çemberi tekniği ile kötü bireylerin seçilme şansı da vardır bundan dolayı kötü bireylerin de seçilerek çaprazlamaya uğratılır ve iki tane birey oluşmaktadır. Toplamda 3 tane bireyden hangisi en küçük uygunluk değerine sahip ise o birey populasyona eklenir. Böylelikle, kötü bireylerin iyi birey olma şansı bir önceki iterasyonlara göre artmaktadır. SCL yönteminde ise sıralı seçme yöntemi kullanıldığından herhangi iki birey çaprazlamaya uğratılarak yeni bir birey oluşur ve bu yeni

birey de mutasyona uğratılıp populasyona eklenir. Böylelikle, iyi bireylerin populasyonda kalma ihtimali giderek azalma gösterebilir.

4.3.4. GRASP

GRASP için kullanılan parametreler sırasıyla:

i. Maximum iterasyon (maxi) : 100,1000, 2000 ve 5000 değerlerinden oluşmaktadır.
ii. Çözümlerin sayısı (seed) : 50, 500, 1000 ve 2500 değerlerinden oluşmaktadır.
iii. Mutasyon (MP) : 0.05 değerinden oluşmaktadır.

Parametrelerin farklı kombinasyonlarıyla aşağıdaki tabloda (Tablo 22) görüldüğü gibi 10 farklı GRASP sezgiseli elde ettik.

Tablo 22— GRASP algoritma parametreleri

Procedure	MI	Seed
R1	100	50
R2	1000	500
R3	5000	500
R4	5000	2500
R5	5000	1000
R6	2000	500
R7	2000	1000
R8	2000	50
R9	2000	2500
R10	5000	50

10 sezgisel 8 tane test probleminde test edilmiştir, her test 10 defa çalıştırılmıştır. Test sonuçları ile ilgili analizler Tablo 23'te sunulmaktadır.

Tablo 23— Problemlerin en iyi ortalama ve en iyi uygunluk değeri

Problem	En iyi Ortalama		En iyi Çözüm	
PS11AK08-9	939,0	R10	852	R10
PS11AK1011	936,6	R3	836	R10
PS11AK12-7	982,2	R4	884	R3
PS11AK15-4	804,8	R5	742	R4
PS11AK16-3	1527,8	R4	1444	R3
PS11AK16-4	1571,8	R5	1486	R5
PS11AK16-5	1687,2	R4	1556	R5
PS11AK17N3	1398,4	R3	1314	R3

10 tane sezgisel 8 tane problem için test edilmiştir. Her bir sezgiselin 8 problemde almış olduğu en iyi ortalama ve en iyi uygunluk değerlerinin sayısı Tablo 24'te sunulmuştur.

Tablo 24— Test sonuçlar ve kullandıkları parametreler

İşlem Adı	En İyi Ortalamaya Sahip Olma Sayısı	En İyi Çözüme Sahip Olma Sayısı	Maxi	Seed
R1	0	0	100	50
R2	0	0	1000	500
R3	2	3	5000	500
R4	3	1	5000	2500
R5	2	2	5000	1000
R6	*0*	0	2000	500
R7	*0*	0	2000	1000
R8	*0*	0	2000	50
R9	*0*	0	2000	2500
R10	*1*	2	5000	50

R4 değeri ortalamada 3 kez ve uygunluk değerinde 1 kez başarılı olduğu için en iyi çözümdür. En iyi sezgisel çözüm R4'dür. (maxi) = 5000, (seed)=2500 değerlerinin performansı diğerlerinden daha iyidir. İterasyon sayısı ve çözümlerin sayısı ne kadar büyük olursa algoritma o kadar iyi performans göstermektedir.

4.3.5. Meta-sezgisellerin karşılaştırılması

Benzetimli Tavlama, Genetik Algoritma, Dağınık Arama ve GRASP metotları aynı ortamda test edilmiştir. Bu kapsamda metotların en iyi işlem adı ile her bir problemin en iyi ortalama ve en iyi çözümleri karşılaştırılacaktır.

Benzetimli tavlama algoritmasında en iyi sonucu H14, Genetik algoritmada en iyi sonucu G6, Dağınık arama'da en iyi sonucu S5 ve GRASP algoritmasında en iyi sonucu R4 vermiştir.

Tablo 25'te her bir algoritmanın sunduğu en iyi sezgiselin vermiş olduğu en iyi ortalama değerleri sunulmaktadır.

Tablo 25— En iyi Ortalama

Problem	En İyi Ortalama				En İyi Ortalama Sonucu
	H14	G6	S5	R4	
PS11AK08-9	820	856,2	993,2	967,4	H14
PS11AK1011	806,6	849,2	976,6	939,4	H14
PS11AK12-7	842,2	917,2	1028,8	982,8	H14
PS11AK15-4	752	762,8	835,8	806,6	H14
PS11AK16-3	1395,6	1407,6	1614,6	1527,8	H14
PS11AK16-4	1461,2	1490,6	1684,2	1582,2	H14
PS11AK16-5	1542,6	1603,8	1802,2	1687,2	H14
PS11AK17N3	1165,4	1230,4	1494,0	1402,8	H14

H14 değeri ortalamada 8 kez başarılı olduğu için en iyi çözümdür. Diğer ifadeyle noi = N^3 , R=20,a=1.5, b=1.1 ve T=1000 parametre değerleriyle çalıştırılan Benzetimli Tavlama'nın performansı diğerlerinden daha iyidir. Diğer algoritmalara baktığımızda, ikinci sırada Genetik algoritma, üçüncü sırada GRASP ve sonuncu sırada Dağınık arama algoritmasıdır.

Tablo 26'da ise her bir algoritmanın sunduğu en iyi sezgiselin vermiş olduğu en iyi çözüm değerleri sunulmaktadır.

Tablo 26—En iyi çözüm

Problem	En İyi Çözüm				En İyi Çözüm sonucu
	H14	G6	S5	R4	
PS11AK08-9	740	806	870	904	H14
PS11AK1011	756	808	858	872	H14
PS11AK12-7	800	858	886	928	H14
PS11AK15-4	718	716	772	742	G6
PS11AK16-3	1370	1378	1398	1450	H14
PS11AK16-4	1392	1458	1502	1510	H14
PS11AK16-5	1474	1546	1540	1632	H14
PS11AK17N3	1076	1154	1248	1320	H14

H14 değeri ortalamada 7 kez başarılı olduğu için en iyi çözümdür. PS11AK15-4 problem çözümünde Benzetimli Tavlama algoritmasına göre iyi sonuç veren Genetik algoritmadır. Üçüncü sırada Dağınık arama, ve sonuncu ise GRASP algoritmasıdır.

Tablo 27'de, problem bazında, her bir algoritmanın sunduğu en iyi ortalama değerini göstermektedir.

Tablo 27— En iyi Ortalama

Problem	En İyi Ortalama				En İyi Ortalama Sonucu
	SA	GA	SS	GRASP	
PS11AK08-9	799 H5	856,2 G6	993,2 S5	939,0 R10	H5
PS11AK1011	806,6 H14	849,2 G6	976,6 S5	936,6 R3	H14
PS11AK12-7	818,6 H13	917,2 G6	1028,8 S5	982,8 R4	H13
PS11AK15-4	746,8 H7	762,8 G6	835,8 S5	804,8 R5	H7
PS11AK16-3	1395,6 H14	1407,6 G6	1614,6 S5	1527,8 R4	H14
PS11AK16-4	1441,6 H10	1490,6 G6	1599,2 S7	1571,8 R5	H10
PS11AK16-5	1502,2 H10	1593,8 G5	1802,6 S5	1687,2 R4	H10
PS11AK17N3	1156,2 H7	1230,4 G6	1494 S5	1398,4 R3	H7

H14 sezgiseli ortalamada 2 kez, H7 sezgiseli ortalamada 2 kez, H10 sezgiseli ortalamada 2 kez, H5 ve H13 sezgiselleri ortalamada 1 kez başarılı olmuşlardır. Benzetimli Tavlama algoritması her bir problem için en iyi ortalama değeri sağlamıştır. İkinci sırada Genetik Algoritma, üçüncü sırada GRASP ve son sırada ise Dağınık arama gelmektedir.

Her bir problem için sunulan en iyi çözümler aşağıdaki gibidir:

Tablo 28'de, problem bazında, her bir algoritmanın sunduğu en iyi çözüm değerini göstermektedir.

Tablo 28— En iyi çözüm

Problem	En İyi Çözüm				En İyi Çözüm Sonucu
	SA	GA	SS	GX	
PS11AK08-9	740 H14	806 G6	870 S5	852 R10	H14
PS11AK1011	750 H13	802 G5	858 S5	836 R10	H13
PS11AK12-7	774 H13	854 G5	886 S5	884 R3	H13
PS11AK15-4	714 H6, H11	716 G6	772 S5	742 R4	H6, H11
PS11AK16-3	1348 H12	1378 G6	1284 S3	1444 R3	S3
PS11AK16-4	1392 H6,H14	1458 G5	1450 S7	1486 R5	H6, H14
PS11AK16-5	1454 H6	1514 G5	1540 S5	1556 R5	H6
PS11AK17N3	1076 H14	1154 G6	1248 S5	1314 R3	H14

H14 değeri ortalamada 3 kez, H13 değeri ortalamada 2 kez, H6 değeri ortalamada 2 kez, H11 ve S3 değerleri ortalamada 1 kez başarılı olmuşlardır. Benzetimli Tavlama algoritması her bir problem için en iyi çözüm değeri sağlamıştır. İkinci sırada Genetik Algoritma, üçüncü sırada GRASP ve son sırada ise Dağınık Arama vardır.

4.3.6. Sonuç

52 sezgisel algoritma 8 tane gerçek problem üzerinde test edilmiştir. KAP çözümlerinde en iyi çözümü ve performansı sağlayan Benzetimli Tavlama algoritmasıdır. En iyi sezgisel çözüm H14'dür. noi=N^3 , R=20,a=1.5, b=1.1 ve T=1000 değerlerinin performansı diğerlerinden daha iyidir.

Algoritmaların en iyi çözümü sundukları (H14, G6, S5, R4) değerlerinde ise, H14, 8 kez başarılı olduğu için en iyi ortalamaya sahip çözümdür. Diğerlerin performansı G6, R4 ve S5 şeklindedir.

Algoritmaların en iyi çözümü sundukları (H14, G6, S5, R4) değerlerinde ise, H14, 7 kez başarılı olduğu için en iyi çözüme sahip çözümdür. G6 çözümü ise 1 kez başarılı olmuştur.

En iyi ortalamada 8 ve en iyi çözümde 7 kez başarılı olduğu için H14 en iyi sezgisel çözümdür.

Algoritmaların her bir problem için farklı çözümler sundukları değerlerin karşılaştırılması yapıldığında Benzetimli Tavlama algoritmasının en iyi meta sezgisel algoritma olduğu saptanmıştır. Bu sonuç Duman ve Or (2007) ile de paraleldir. Diğer algoritmaların sıralaması, Genetik algoritma, GRASP ve Dağınık Arama şeklindedir.

4.4.Esas problem için bilgisayısal sonuçlar

Algoritmalar tasarlandıktan sonra bilgisayısal test çalışmaları yapılmıştır. Bu çalışmalarda kullanılan dizgi makinesi parametreleri Ellis ve diğerlerinin (2001) çalışmasında kullanılan makine parametrelerine çok yakındır. Oradaki parametreler ya yuvarlanmış ya da doğrusal hale getirilmiş olarak kullanılmıştır. Bu, konu için yazında kolaylıkla yararlanılabilecek tek kaynak olduğu için tercih edilmiştir (Alkaya ve Duman, 2009). Makine özelliklerini açıkça verecek olursak taşıyıcı masa hızı 280 mm/saniye, dört döner taret değerleri 0.15, 0.19, 0.24 ve 0.29 saniye, besleyici düzeneğinin bir slotluk hareketi 0.18 saniyede ve her ilave hareketi 0.045 saniyedir. Kullanılan BDK'lar ise iki grup olup birinci grup gerçek veriler kullanılan Duman ve Or (2007)'den alınmıştır. İkinci grup veriler ise projenin birinci aşamasında tasarlanan rassal BDK üreticisinin verileridir. Hatırlamak gerekirse rassal BDK'lar homojen ve yapısal olarak iki çeşit tasarlanmıştır.

Tekrarlamalı yöntemde kullanılan çözüm tekniklerinin en iyi parametre değerleri önceki dönem çalışmalarında ortaya çıkarılmıştır. Ancak esas probleme uygulanacak metasezgisellerin parametre değerlerini tespit etmek gerekmektedir.

Her bir metasezgiselde hangi parametre değerlerinin en iyi performansı sergilediğinin belirlenmesi için bütün parametre kombinasyonlarının denenmesi gerekir. Yeni komşu belirleme yöntemlerini de dahil edersek benzetimli tavlama için 80, genetik algoritma için 30, yapay arı kolonisi için 135 farklı kombinasyon oluşmaktadır. Her bir metasezgisel için oluşan farklı parametre değer kombinasyonlarının en iyisini bulmak için testler gerçek BDK'lar üzerinde yapılmıştır. Her bir parametre değer kümesi için algoritmalar 10'ar defa koşturulmuş ve her 10 koşmanın ortalama ve minimum değerleri kaydedilmiştir.

Parametreleri analiz ettiğimiz yöntem tek bir parametrenin bütün değerlerinde diğer bütün koşmaların ortalamasının alınmasıdır. Böylelikle diğer parametrelerin etkileri de göz önüne alınarak bir parametrenin aldığı değerler kıyaslanmış olacaktır. Örneğin, benzetimli tavlama için T=100 için diğer bütün parametreler, alabileceği bütün değerleri alarak testler yapılır ve elde edilen en iyi sonuç aynı işlemin T=1000 için yapılmasıyla kıyaslanır.

4.4.1. Benzetimli Tavlama sonuçları

Bu çalışmanın sonuçları Tablo 29'de verilmiştir. Tablodaki her bir değer bulunabilen en iyi çözümden uzaklığı göreceli olarak vermektedir.

Tablo 29—Benzetimli Tavlama parametre performans analizi

Parametre	Değerler	PS11AK08-9	PS11AK1011	PS11AK12-7	PS11AK15-4	PS11AK16-3	PS11AK16-4	PS11AK16-5	PS11AK17N3	Ortalama
T	100	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
	1000	8,5%	4,6%	10,5%	0,7%	1,5%	2,8%	5,3%	5,2%	4,9%

R	5	2,7%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,3%
	20	0,0%	2,9%	1,6%	0,7%	1,0%	2,8%	5,3%	2,1%	2,0%
a	1,1	2,7%	0,0%	0,0%	5,4%	0,0%	0,0%	0,0%	0,0%	1,01%
	1,5	0,0%	1,1%	1,6%	0,0%	0,5%	0,6%	2,0%	2,1%	0,98%
b	1,1	2,7%	0,0%	0,0%	0,7%	0,0%	0,0%	0,0%	0,0%	0,4%
	1,5	0,0%	1,1%	1,6%	0,0%	0,5%	0,6%	2,0%	2,1%	1,0%
nm	1	0,6%	3,1%	2,9%	0,7%	1,2%	2,0%	4,4%	3,0%	2,2%
	2	0,0%	0,0%	2,6%	1,3%	0,0%	0,0%	0,0%	0,0%	0,5%
	3	1,1%	3,4%	0,0%	2,7%	1,5%	2,6%	2,8%	2,8%	2,1%
	4	20,7%	17,8%	20,5%	17,5%	20,6%	25,3%	24,2%	21,4%	21,0%
	5	5,2%	3,2%	3,7%	0,0%	1,3%	3,0%	3,6%	4,0%	3,0%

Tablo incelendiğinde şu sonuçlara ulaşabiliriz. T=100, R=5 ve b=1.1'in diğer değer alternatiflerine göre en iyi sonucu elde etmede daha başarılı olduğu görülmektedir. Öte yandan, a=1.1 değeri en iyi sonucu altı defa elde etme etmesine karşılık ortalama olarak çok az da olsa a=1.5 değerine nispeten daha kötü performans sergilemiştir. Yeni komşu seçme teknikleri kıyaslandığında en iyi sonucu ikinci yöntem ve en kötü sonucu dördüncü yöntem vermiştir.

Dolayısıyla, çalıştığımız problem tiplerinde en iyi performans bu değer kümesiyle elde edilmiştir.

Bu değer kümesinin ilk dört parametre değerinden çıkacak en önemli sonuç benzetimli tavlama algoritması gerçek BDK örneklerinde kötü çözümlere daha fazla şans verdikçe daha iyi çözümlere geçiş yapamamaktadır. Dolayısıyla, yerel optimumlara takılmadan yeni çözümlere geçme işlemi daha çok kötü çözümlerde gezinmeyi beraberinde getirmektedir. Yeni komşu bulma yöntemlerinden ise en başarılısı ikinci olarak isimlendirdiğimiz mevcut çözümün hangi alt çözümünde değişiklik yapılacağına karar vermek için her birine (parça sayısı ve parça tipi sayılarının kareleriyle doğru orantılı olarak) ağırlıklı şans verme yöntemidir. Bu şekilde örneğin 50 parça tipi ve 250 parça sayısı olan bir BDK'da dizgi sırasının değiştirilmesiyle elde edilen yeni çözüme 1'e 25 oranında şans vermiş oluyoruz. Bu şekilde amaçlanan, alt çözümlere farklı kombinasyonlarla elde edilecek çözüm sayıları kadar şans vermektir. Bu yöntemin eşit şans verme ya da yeni çözümün alt çözümlerin sırayla değiştirilerek elde edilmesi yönteminden daha iyi olduğu görülmektedir. Yeni komşu elde etme yöntemlerinden en başarısızı her iki alt çözümün aynı anda değiştirilmesiyle elde edilme yöntemidir. Aslında bu da öngörülebilir bir sonuçtur çünkü başarısı iyi bir dizgi sırası ya da besleyici düzenine bağlı bir çözüm bir anda her ikisinde yapılan değişiklikle başarısız bir çözüme dönüşebilir.

Sonuç olarak benzetimli tavlama metasezgiselinin BDK problemleri için en uygun parametre değerleri T=100, R=5 a=1.5 ve b=1.1, nm=2 olarak belirlenmiştir ve bu değerler benzetimli tavlama metasezgiseli rassal BDK'lar üzerinde koşturulurken kullanılacaktır.

4.4.2. Genetik Algoritma sonuçları

Genetik algoritmanın gerçek BDK'lar üzerinde koşturulmasıyla elde edilen sonuçlar Tablo 30'da verilmiştir. Tablodaki her bir değer genetik algoritmanın bulabildiği en iyi çözümden uzaklığı göreceli olarak vermektedir.

Tablo 30—Genetik algoritma parametre performans analizi

Parameter	Değerler	PS11AK08-9	PS11AK1011	PS11AK12-7	PS11AK15-4	PS11AK16-3	PS11AK16-4	PS11AK16-5	PS11AK17N3	Ortalama
nog	40	22%	26%	30%	23%	20%	23%	21%	31%	24%
	400	12%	17%	20%	15%	11%	12%	11%	18%	15%
noi	50	3%	26%	31%	24%	19%	21%	20%	30%	22%
	100	2%	22%	25%	19%	15%	17%	16%	24%	18%
	200	0%	17%	19%	14%	12%	13%	12%	19%	13%
nm	1	18%	24%	26%	21%	17%	19%	17%	25%	21%
	2	16%	20%	25%	18%	13%	15%	14%	23%	18%
	3	19%	23%	26%	20%	17%	19%	17%	25%	21%
	4	15%	19%	20%	17%	13%	15%	14%	22%	17%
	5	16%	19%	23%	17%	13%	14%	13%	23%	17%

Tablo incelendiğinde şu sonuçlara ulaşılabilir. Nesil sayısının artması ve çözüm sayısının artması bulunabilen çözüm yelpazesini genişletmekte ve dolayısıyla daha iyi çözümler elde edilebilmektedir. Değer kümeleriyle bulunabilen en küçük sonuçlar da göz önüne alındığında kullanılan yeni komşu bulma yöntemlerinden en başarılısı beşinci yöntem olarak tasarladığımızdır. Yani ilk olarak parça sayısının karesi kadar sayıda yeni çözümü dizgi sırasında değişiklik yaparak, sonrada parça tipi sayısının karesi kadar yeni çözümü besleyici düzeninde değişiklik yaparak elde etmek ve bu işlemi algoritma sonlanana kadar tekrarlamak.

Sonuç olarak genetik algoritmanın BDK problemleri için en uygun parametre değerleri nog=400, noi=200 ve nm=5 olarak belirlenmiştir ve bu değerler genetik algoritma rassal BDK'lar üzerinde koşturulurken kullanılacaktır.

4.4.3. Yapay arı kolonisi sonuçları

Yapay arı kolonisi metasezgiselinin gerçek BDK'lar üzerinde koşturulmasıyla elde edilen sonuçlar Tablo 31'te verilmiştir. Tablodaki her bir değer Yapay arı kolonisi metasezgiselinin bulabildiği en iyi çözümden uzaklığı göreceli olarak vermektedir.

Tablo 31— Yapay arı kolonisi parametre performans analizi

Parametre	Değerler	PS11AK08-9	PS11AK1011	PS11AK12-7	PS11AK15-4	PS11AK16-3	PS11AK16-4	PS11AK16-5	PS11AK17N3	Ortalama
Döngü	100	104%	96%	110%	90%	96%	99%	97%	115%	101%
	1000	50%	47%	54%	46%	51%	52%	51%	55%	51%
	10000	28%	26%	32%	27%	27%	28%	26%	30%	28%

Limit	100	66%	61%	71%	59%	63%	64%	63%	72%	65%
	1000	58%	54%	62%	52%	56%	57%	55%	64%	57%
	10000	58%	54%	62%	52%	56%	57%	56%	64%	57%
cs	10	60%	56%	65%	54%	58%	60%	58%	67%	60%
	20	61%	56%	65%	54%	58%	59%	58%	67%	60%
	50	61%	56%	65%	54%	58%	60%	58%	67%	60%
nm	1	52%	48%	56%	47%	50%	51%	50%	58%	51%
	2	51%	48%	56%	46%	50%	52%	50%	58%	51%
	3	52%	48%	57%	46%	49%	51%	49%	58%	51%
	4	69%	63%	73%	60%	65%	66%	65%	74%	67%
	5	52%	49%	57%	47%	51%	52%	51%	59%	52%

Döngü sayısının artması ile işçi arıların kaynaklara gönderilerek gözcü arıları çağırması yani komşu çözümlerin araştırılması işlemi daha çok yapılacak ve daha iyi çözümlere ulaşma ihtimali artacaktır. Zaten sonuçlar da bunu desteklemektedir. Öte yandan düşük bir limit değeri komşuları araştırılan bir çözümün yeteri kadar araştırılmadan terk edilmesine, yüksek bir limit değeri ise en iyi komşusu bulunan çözümde gereğinden fazla kalınmasına neden olacaktır. Sonuçlara bakıldığında limit parametresi için 1000 değerinin en uygunu olduğu görülmektedir. Kolonideki arı sayısının (cs) değerleri arasında bir performans farkı gözükmemektedir. Yüksek bir cs değeri daha fazla sayıda arama işleminin paralel olarak yürütülmesi anlamına gelir. Düşük bir cs değeri ise çok az çözüm üzerinde komşu çözümlerin aranması anlamına gelir. Değer kümeleriyle bulunabilen en küçük sonuçlar da göz önüne alındığında kullanılan yeni komşu bulma yöntemlerinden en başarılısı ikinci yöntem olarak tasarladığımızdır.

Sonuç olarak yapay arı kolonisi metasezgiselinin BDK problemleri için en uygun parametre değerleri *nog*=400, *noi*=200 ve *nm*=5 olarak belirlenmiştir ve bu değerler yapay arı kolonisi metasezgiseli rassal BDK'lar üzerinde koşturulurken kullanılacaktır.

4.4.4. Tekrarlamalı yöntem sonuçları

Yukarıda belirttiğimiz gibi tekrarlamalı yöntem uygulanırken SDSSP ve KAP için uygulayacağımız yöntemler projenin bu safhasına kadar geliştirdiğimiz ve belirlediğimiz yöntemler olacaktır.

SDSSP için tasarladığımız İlk Noktayı Toplam Masrafa göre Değistir Yöntemi (İNTMDY), Gelişmiş Sapakları Ertele (GSE) ve RRTLEM yöntemleri bu aşamada kullanılabilir. Bu yöntemler iyileştirme yöntemleriydi ve ilk çözümü ATMA algoritmasıyla elde ediyorduk. Öte yandan üçüncü dönemde yaptığımız çalışmalarda BDK dizgisi konusunda ortaya çıkan gerçek KAP örnekleri için bulunan en iyi metasezgiselin benzetimli tavlama olduğunu görmüştük. Bu metasezgiselin KAP örnekleri için en iyi performans gösteren versiyonu ise T=1000, R=20 a=1.5 ve b=1.1 parametreleriyle çalışıyordu. Dolayısıyla tekrar bir parametre değer analizine girmeden benzetimli tavlama bu parametrelerle uygulanmıştır.

Önceki altbölümlerde açıklanan tekrarlamalı yöntemi hatırlatacak olursak: KAP için rastgele bir çözümle başla ve bunu kullanarak toplam dizgi zamanını enazlayacak şekilde SDSSP için

bir çözüm önerisi bul. Sonrasında SDSSP için bulunan çözümü kullanarak ve bir önceki adımdaki besleyici düzeni çözümünden başlayarak yeni bir KAP çözümü üret. Bu tekrarlamalı süreci belli bir sayı kadar devam ettir.

Bu çalışma prensibiyle koşturduğumuz ve 20 defa tekrar eden yöntemin ara sonuçlarını örnek bir BDK üzerinde(PS11AK08-9) Tablo 32'de görebiliriz.

Tablo 32—Örnek bir BDK üzerinde tekrarlamalı yöntem sonuçları

Tekrar No	KAP çözüldükten sonraki dizgi zamanı	SDSSP çözüldükten sonraki dizgi zamanı
0	0	43,04232143
1	43,04232143	43,04232143
2	42,99142857	43,09460714
3	43,09460714	43,09460714
4	43,07853571	42,70932143
5	42,70932143	42,70932143
6	42,70932143	42,70932143
7	43,653	43,25896429
8	42,72892857	**40,74835714**
9	40,74835714	40,74835714
10	40,74835714	40,74835714
11	40,74835714	40,74835714
12	40,74835714	40,74835714
13	40,74835714	40,74835714
14	40,74835714	40,74835714
15	40,74835714	40,74835714
16	40,74835714	40,74835714
17	40,74835714	40,74835714
18	40,74835714	40,74835714
19	40,74835714	40,74835714
20	40,74835714	40,74835714

Buradan çıkardığımız sonuç ve (Duman, 1998)'de de belirtildiği gibi tekrarlamalı yöntem kullanımında 20 tekrarın fazlasıyla yeterli olabileceğidir. Gerçek BDK'lar üzerinde gerçekleştirilen tekrarlamalı yöntem çalışmamızın sonuçları diğer metasezgisel yöntemlerin sonuçlarıyla beraber özetlenerek Tablo 33'te gösterilmiştir. Bu sonuçlara göre tekrarlamalı yöntem (çok daha uzun koşma zamanı gerektirmesi bedeline karşılık) bütün BDK'larda en iyi sonucu vermektedir. Diğer yöntemlerin sıralaması ise BT, YAK ve GA olarak görülmektedir. Literatüre yeni kazandırılan YAK metasezgiselinin GA'yı geçmesi ve BT ile yarışır durumda olması YAK üzerinde daha fazla çalışma yapılmasına değebileceğini göstermektedir.

Tablo 33—Bütün yöntemlerin gerçek BDK'lar üzerinde sonuçları

BDK	Dizgi Zamanı (sn)				Koşma Zamanı (sn)			
	BT	GA	YAK	Tekrarlamalı	BT	GA	YAK	Tekrarlamalı
PS11AK08-9	44,18	71,63	44,56	39,82	8	57	34	1299
PS11AK1011	46,56	73,23	48,21	41,46	6	60	36	1778

PS11AK12-7	47,30	78,15	49,49	40,75	9	59	35	1144
PS11AK15-4	46,74	71,48	47,05	38,83	4	62	38	2517
PS11AK16-3	79,77	128,23	85,33	63,52	10	88	56	4186
PS11AK16-4	81,68	131,51	87,45	63,78	11	89	57	3755
PS11AK16-5	87,99	142,53	93,17	70,22	13	94	60	3890
PS11AK17N3	58,29	95,27	60,35	49,83	13	68	42	2843

Tekrarlamalı yöntemle elde edilen sonuçların problemin tümüne çözüm getiren metasezgisellere karşı gösterdiği performans farkı dikkat çekicidir. Şüphesiz ki bu farkın bir nedeni çok daha uzun süren koşum zamanıdır. Ancak, tekrarlamalı yöntemin yirmide bir koşum zamanı gerektiren ilk adımında elde edilen sonuçlar incelendiğinde bile diğerlerine göre belirgin bir performans farkı olduğu görülmektedir (tüm kartların tüm iterasyon sonuçlarını burada vermek çok yer tutacaktır ancak demek istediğimiz yukarıda örnek olarak verdiğimiz PS11AK12-7 isimli kart üzerinde de görülebilir). Dolayısıyla koşum zamanlarının eşitlenmesi halinde bile tekrarlamalı yöntem daha başarılıdır. Bunun nedeni, SDSSP probleminin çok karmaşık olması, farklı ağırlık gruplarından dolayı verilen bir çözüm üzerinde çoğu ikili yer değişikliğinin iyileştirme yapamayacağı ve dolayısıyla metasezgisellerin çok uzun zaman verilse dahi çözüm kalitesini fazla iyileştiremeyeceği beklentisiyle açıklanabilir. Öte yandan, problemin (özellikle SDSSP nin) özel yapısını dikkate alarak geliştirilmiş olan yapısal sezgisel algoritmaların (ATMA ve diğerleri) performansları çok daha iyi olabilmektedir. Aslında bu görüntü bizleri iki ana sonuca götürmektedir. Birincisi, metasezgiseller bir çok farklı probleme başarıyla uygulanabilmelerine rağmen bizim problemimiz için başarılı olamamaktadırlar. İkincisi ise, SDSSP için geliştirdiğimiz basit yapısal sezgiseller gerçekten başarılıdır.

4.4.5. Çözüm yöntemlerinin rassal BDK'lar üzerindeki testleri

Belirlenen parametre değerleri kullanılarak, önerilen yöntemler rassal BDK'lar üzerinde de test edilmiştir. Sonuçlar Tablo 34'da verilmiştir. Bu tabloda da her iki BDK tipi için de tekrarlamalı yöntemin diğer yöntemlere göre üstünlüğü ortaya çıkmaktadır. Bu durum, yukarıdakine benzer bir şekilde açıklanabilir. Önceki tablodan farklı olarak ise burada YAK'nin az da olsa BT nin önüne geçtiği göze çarpmaktadır. Proje çalışmasında öne sürülen çözüm yöntemlerinin önem ve değeri bu son iki tabloda net olarak görülmektedir.

Tablo 34—Rassal BDK sonuçları

	Rassal BDK'lar (ortalama değerleri)			
	Homojen		Yapısal	
Algoritma	Koşma zamanı	Dizgi zamanı (sn)	Koşma zamanı	Dizgi zamanı (sn)
BT	5,12	33,12	6,02	33,90
GA	42,85	52,73	43,27	54,76
YAK	23,66	32,73	23,93	33,41
Tekrarlamalı	770,60	26,83	684,97	26,77

5. Sonuç

Sonuç olarak 108M198 kodlu TÜBİTAK 1001 projemiz planlandığı gibi yürütülmüş ve hedeflerine ulaşmıştır. Bu çalışmanın yapılmasını mümkün kılan TÜBİTAK'a ve neredeyse proje çalışanlarımız kadar emek veren ve yorumlarıyla bizlere yön veren proje izleyicimize katkılarından dolayı çok teşekkür ederiz.

Projede hedeflenen teknik detaylardan daha önemli gördüğümüz bilim insanı yetiştirme amacı tam olarak yerine gelmiş ve bu proje süresince bir doktora öğrencisi ve iki yüksek lisans öğrencisi mezun olmuştur.

Proje sonucunda ortaya çıkan yayınlar şunlardır:

- Alkaya, A.F., Duman, E.: "A New Generalization of the Traveling Salesman Problem", Applied and Computational Mathematics, 9, (2), (2010), 162-175. (SCI Expanded)
- Alkaya, A.F.; Duman, E.: "A Literature Survey of the Operation Optimization in Chip Shooter Placement Machines", *Proceedings of PICMET'09*, Portland, OR, USA, August 2-6, (2009), 3296-3306.
- Kıyıcığı, B.M., Duman, E., Alkaya, A.F.: "Finding Best Performing Solution Algorithm for the QAP", Proceedings of IMS2010, Sarajevo, Bosnia Herzegovina, September 15-17, (2010).
- Duman, E., Alkaya, A.F., Demirkale, H.: "Optimizing the Operations of Chip Shooter Machines", Proceedings of META2010, Djerba Island, Tunisia, October 27-31, (2010).
- Demirkale, H., Duman, E., Alkaya, A.F.: "Exact and Metahueristic Approaches for Optimizing the Operations of Chip Mounter Machines", Proceeding of CISIM2010, Cracow, Poland, October 8-10, (2010), 120-125.

Ayrıca aşağıdaki iki makalede SCI indeksli dergilere sunulmuştur ve değerlendirme sürecindedir:

- "An Application of the Sequence Dependent Traveling Salesman Problem: Assembly Time Minimization of Chip Mounters" başlığıyla Journal of Operational Research Society.
- "Combining and Solving SDTSP and QAP Having Conflicting Objectives: A Case Study In PCB Assembly" Computers and Operations Research.

Bütünlük açısından bu yayınların tamamı rapora eklenmiştir.

Bu proje devamında ve sonrasında yapılabilecek çalışmalar iki ana başlık altında olabilir. Öncelikle burada ilk olarak tanımlanmış ve uygulaması tartışılmış olan SDSSP'nin başka türleri ve uygulama alanları araştırılabilir. İkinci olarak ise bu projeden elde edilen bilgi ve

tecrübelerin ışığı altında sektörde kullanılan farklı tiplerdeki dizgi makinelerinin eniyileme problemleri üzerinde çalışılabilir.

Referanslar,

1. AHUJA,R.K., Orlin, J.B., Tiwari,A., A greedy genetic algorithm for the quadratic assignment problem, *Computers & Operations Research*, 27, 917-934, (2000)
2. ALBIACH, J., Sanchis, J. M., Soler, D., An asymmetric TSP with time windows and with time-dependent travel times and costs, An exact solution through a graph transformation, *European Journal of Operational Research*, 189, 789–802, (2008)
3. ALKAYA, A.F., Duman, E., Eyler, M.A, Assembly time minimization for an electronic component placement machine, *WSEAS Transactions on Computers*, 7, 326-340, (2008)
4. ALKAYA, A.F. and Duman, E., A Literature Survey of the Operation Optimization in Chip Shooter Placement Machines, *Proceedings of PICMET'09*, Portland, OR, USA, August 2-6, (2009), 3296-3306.
5. APPLEGATE, D., Bixby, R., Chvátal, V., Cook, W., Finding tours in the TSP *Forschungsinstitut für Diskrete Mathematik* Report No. 99885
6. APPLEGATE, D.L., Bixby, R.E., Chvatal, V., Cook, W.J., The Traveling Salesman Problem, A Computational Study, *Princeton Series in Applied Mathematics*, (2006).
7. BABIN, G., Deneault, S.and Laporte, G., Improvements to the Or-opt heuristic for the symmetric travelling salesman problem, *Journal of the Operational Research Society*, 58, (2007), 402-407.
8. BALAS, E., Ceria, S., Cornuejols G. and Natraj, N., Gomory Cuts revisited, *Operations Research Letters*, 19, (1996a), 1-9.
9. BALAS, E., Ceria, S. and Cornuejols, G., Mixed 0-1 programming by lift-and-project in a branch-and-cut framework, *Management Science*, 42, (1996b), 1129-1246.
10. BİGRAS, L.-P., Gamache, M., Savard, G., The Time-Dependent Traveling Salesman Problem and Single Machine Scheduling Problems with Sequence Dependent Setup Times, *Discrete Optimization*, 5, (2008) 685-699.
11. BOYD, E.A., Fenchel Cutting Planes for Integer Programs, *Operations Research*, 42, (1994), 53-64.
12. BORCHERSA, B., Mitchel, J.E., A computational comparison of branch and bound and outer approximation algorithms for 0–1 mixed integer nonlinear programs
13. CHAN,F.T.S., Lau, K.W., Chan, P.L.Y, Choy, K.L., Two-stage approach for machine-part grouping and cell layout problems, *Robotics and Computer-Integrated Manufacturing*, 22 , (2006), 217–238.
14. CHWIF, L., Barretto, M.R.P., Moscato, L.A., A solution to the facility layout problem using simulated annealing, *Computers in Industry*, 36,(1998), 125-132.
15. CLAUSEN, J., Branch and Bound Algorithms - Principles and Examples, *Department of Computer Science, University of Copenhagen, Universitetsparken 1, DK-2100 Copenhagen, Denmark,* (1999).

16. CROES, G., A Method for Solving Traveling-Salesman Problems, *Operations Research*, 6, (1958), 791-812.

17. CUNG, V.,Mautor,T.,Michelon,P.,Tavares,A., A Scatter search based approach for the Quadratic Assigment Problem. , IEEE, (1997).

18. CSASZAR, P., Tirpak, T. M., Nelson, P. C., Optimization of a high-speed placement machine using tabu search algorithms, Annals of Operations Research, 96, (2000) 125-147.

19. DANTZIG, G., Fulkerson, R., Johnson, S., Solution of a Large Scale Traveling Salesman problem, *Operations Research 2* (1954), pp. 393-410.

20. DANTZIG, G.B., Ramser J.H., The truck dispatching problem, *Management Science*, 6, (1959) 80-91.

21. DIKOS,A., Nelson,P.C.,Tirpak,T.M.,Wang,W., Optimization of high-mix printed circuit card assembly using genetic algorithms, *Annals of Operations Research* 75, (1997), 303 – 324.

22. DREZNER, Z., Extensive experiments with hybrid genetic algorithms for the solution of the quadratic assignment problem, *Computers & Operations Research*, 35, (2008), 717-736.

23. DRUD, A., A GRG Code for Large Sparse Dynamic Nonlinear Optimization Problems, *Mathematical Programming*, 31, (1985) 153-191.

24. DRUD, A., Conopt user guide, ARKI Consulting and Development A/S, Bagsvaerd, Denmark

25. DRYSDALE, C., Report, IC Packaging Topped $30B in 2007, Circuits Assembly, The Journal for Mount and Electronics Assembly, (2008), http,//circuitsassembly.com/cms/content/view/6587/95/ (15.05.2008)

26. DUECK, G., New optimization heuristics, the great deluge algorithm and the record-to-record travel, *Journal of Computational Physics*, 104, (1993) 86-92.

27. DUMAN, E. , Optimization Issues in Automated Assembly of Printed Circuit Boards , Bogazici University, Unpublished PhD Thesis , 1998.

28. DUMAN E., Modelling the operations of a component placement machine with rotational turret and stationary component magazine, *Journal of the Operational Research Society*, 58, (2007) 317-325.

29. DUMAN, E., An Application of the Multiple TSP in Printed Circuit Board Assembly, Journal of Operations and Logistics, 2 (3), (2009), III.1-III.9

30. DUMAN E., Or I., Precedence Constrained TSP Arising in Printed Circuit Board Assembly, International Journal of Production Research, 42, (2004) 67-78.

31. DUMAN, E., Or, I., The quadratic assignment problem in the context of the printed circuit board assembly process, *Computers & OR*, 34, (2007) 163-179.

32. DUMONT, J., Robichaud V. Introduction to GAMS Software A Manual for CGE Modelers (2000)

33. DURAN, M.A., Grossman, I.E., An outer approximation algorithm for a class of mixed-integer nonlinear programs, *Math Program* 36 (1986), 307–339.

34. EL-BAZ, M.A., A genetic algorithm for facility layout problems of different manufacturing environments, *Computer & Industrial Engineering*, 47, (2004), 233-246.

35. ELLIS, K.P., Vittes, F.J., Kobza, J.E., Optimizing the Performance of a Surface Mount Placement Machine, IEEE Transactions on Electronics Packaging Manufacturing, 24, (2001) 160-170.

36. FLETCHER, R., Leyffer, S., Solving mixed integer nonlinear programs by outer approximation (1996).

37. FOX, K., Gavish, B., Graves, S.C., An n-Constraint Formulation of the (Time Dependent) Traveling Salesman Problem, *Operations Research*, 28, (1980) 1019–1021.

38. GARY, M.R. and Johnson, D.S., Computers and Intractability, A Guide to the Theory of NP-Completeness, *W.H. Freeman and Company*, (1979).

39. GENDRAU, M., Laporte, G., Musaraganyi, C., Taillard, E.D., A tabu search heuristic for the heterogeneous fleet vehicle routing problem, *Computers & OR*, 26, (1999) 1153-1173.

40. GEOFFRION, A.M., Generalized benders decomposition *JOTA*, 10(4), (1972) 237-260.

41. GOLDEN, B.L., Assad, A.A., Levy L., Gheysens F.G., The Fleet Size and Mix Vehicle Routing Problem, *Computers & OR*, 11, (1984) 49-66.

42. GOLDEN, B.L., Large-Scale Vehide Routing and Related Combinatorial Problems, MIT Operations Research Center, Cambridge, Mass., PhD Thesis (1976).

43. GOLDEN, B.L., Wasil, E.A., Kelly, J.P., Chao, I.M., The impact of metaheuristics on solving the vehicle routing problem, algorithms, problem sets, and computational results, In, Crainic T, Laporte G, editors. *Fleet management and logistics*. Boston, MA, Kluwer (1998).

44. GOMORY, R.E., Outline of an algorithm for integer solutions to linear programs, *Bulletin of the American Mathematical Society*, 64, (1958), 275-278

45. GOMORY, R.E., An algorithm for integer solutions to linear programs, *Recent Advances in Mathematical Programming*, R.L. Graves, P.Wolfe eds. McGraw-Hill, New York, (1963), 269-302.

46. GONG, D., Yamazaki,G., Gen,M., Xu,W., A genetic algorithm method for one-dimensional machine location problems, *Int. J. Production Economics*, 60-61, (1999), 337-342.

47. GROSSMANN, I.E., Viswanathan, J., Vecchietti, A., Dicopt, Engineering Research Design Center, Carnegie Mellon University, Pittsburgh, PA.

48. GRÖTSCHEL, M. and Holland, O., Solution of large-scale traveling salesman problems, Mathematical Programming, 51, 2, (1991), 141-202.

49. GUTIN, G. and Punnen, A., The Traveling Salesman Problem and its Variants, *Kluwer Academic Publishers*, (2002).

50. HAGHANI, A., Jung, S., A dynamic vehicle routing problem with time-dependent travel times, *Computers & Operations Research*, 32, (2005) 2959–2986.

51. HANSEN K., Kraup J., Improvement of the Held-Karp algorithm for the symetric traveling salesman problem, *Mathematical Programming, vol. 7*, (1982) 87-96,

52. HELD, M., Karp, R.M., The Traveling Salesman problem and minimum spanning trees, *Operations Research. 18* (1970) 1138-1162.

53. HO, W. Component Sequencing and Feeder Arrangement for PCB Assembly Machines, Integration, models, solutions (2004)

54. HOLLAND J.H.Adaptation in natural and artificial systems. University of Michigan Press, 1975.

55. ICHOUA, S., Gendreau, M., Potvin J.Y., Vehicle dispatching with time-dependent travel times, *European Journal of Operational Research*, 144, (2003) 379–396.

56. JEEVAN, K., Parthiban, A., Seetharamu, K. N., Azid, I. A., Quadir, G. A., Optimization of PCB Component Placement using Genetic Algorithms, Journal of Electronics Manufacturing, 11, (2002) 69-79

57. JOHN E. M., Branch and cut algorithms for Combinatorial Optimization Problems, *Oxford Univ. Press,* (2002) 65-77.

58. JON L., Raffensperger J. F.,Using AMPL for teaching the TSP, *INFORMS Transactions on Education, Vol. 7, No 1.*, (2006)

59. JÜNGER M., Reinelt G., Thiene S., Provably Good Solutions for the Traveling Salesman Problem, Preprint 94-31, IWR Heidelberg, (1994)

60. JÜNGER, M., Reinelt, G. and Thienel, S., Practical problem solving with cutting plane algorithms in combinatorial optimization, *Combinatorial Optimization, DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, AMS, (1995), 111-152.

61. KARABOGA, D., Basturk, B., A powerful and Efficient Algorithm for Numerical Function Optimization, Artificial Bee Colony (ABC) Algorithm, Journal of Global Optimization, Volume,39, Issue,3,pp,459-171, 2007.

62. KOCIS, G.R., Grossmann, I.E., Relaxation Strategy for the Structural Optimization of Process Flow-sheets, *Industrial and Engineering Chemistry Research*, 26, (1987), 1869

63. KOOPMANS, T., Beckmann, M., Assignment problems and the location of invisible economic activities, *Econometrica*, (1957), 53 – 76.

64. LAWLER L. E., Lenstra J.K., Rinnooy Kan A.H.G., Shmoys D.B., *The Traveling Salesman Problem*. John Wiley & Sons, (1985).

65. LETCHFORD, A.N. and Lodi, A., Strengthening Chavatal-Gomory Cuts and Gomory fractional cuts, *Operations Research Letters*, 30, 2, (2002), 74-82.

66. LI, F., Golden, B., Wasil, E., Very large-scale vehicle routing, new test problems, algorithms, and results, *Computers & OR*, 32, (2005), 1165-1179.

67. LI, F., Golden, B., Wasil,E., The open vehicle routing problem, Algorithms, large-scale test problems, and computational results, *Computers & OR*, 34, (2007) 2918-2930.

68. LI, F., Golden, B., Wasil, E., A record-to-record travel algorithm for solving the heterogeneous fleet vehicle routing problem, *Computers & OR*, 34, (2007) 2734-2742.

69. LIM, M.H., Omatu,S., Efficient genetic algorithms using simple genes exchange local search policy for the quadratic assignment problem, *Computational Optimization and Applications*, 15, (2000), 249-268.

70. LIN, S., Computer solutions of the traveling salesman problem, Bell System Technical Journal, 44, (1965), 2245–2269.

71. LOIOLA, E.M., Abreu N.M., Netto, P.O.B., Hahn, P., Querido, T., A survey of the quadratic assignment problem, *European Journal of Operational Research*, Volume 176, Issue 2, 16 (2007), 657-690.

72. MAHDAVI, I., Paydar M.M., Solimanpur, M., Heidarzade, A., Genetic algorithm approach for solving a cell formation problem in cellular manufacturing, *Expert Systems with Applications* 36, (2009), 6598–6604.

73. MALANDRAKI, C., Daskin, M.S., Time Dependent Vehicle Routing Problems, Formulations, Properties and Heuristic Algorithms, *Transportation Science*, 26, (1992) 185-200.

74. MARCHAND, H., Martin, A., Weismantel, R. and Wolsey, L., Cutting planes in integer and mixed integer programming, *Discrete Applied Mathematics*, 123, (2002), 397-446.

75. MARTI, R. , Laguna, M., Glover,F. Principles of Scatter search , European Journal of Operational Research 169, (2006), 359-372.

76. MARTIN, G.T., Solving the traveling salesman problem by integer linear programming, *CEIR*, (1966)

77. MAWDESLEV, M.J, Jibourib, S.H., Proposed genetic algorithms for construction site layout, *Engineering Applications of Artifical Intelligence*, (2003), 501-509.

78. MCCARL, Bruce A. McCarl GAMS User Guide (2009).

79. MILIOTIS P., Using cutting planes to solve the symmetric traveling salesman problem, *Mathematica Programming 5* (1978) 177-188

80. MISEVICIUS A., Kilda B., The comparison of crossover operators for the quadratic assignment problem, *Information Technology and Control*, 34,2 (2005).

81. MITCHELL, J.E., Branch-and-Cut Algorithms for Combinatorial Optimization Problems, *Handbook of Applied Optimization*, Oxford University Press, (2002), 65-77.

82. NEHI, M., Gelareh, S., A survey of meta-heuristic solution methods for the quadratic assignment problem, *Applied Mathematical Sciences*, 1, 46, (2007), 2293-2312.

83. OR, I., Traveling Salesman Type Combinatorial Problems and Their Relation to the Logistics of Blood Banking, *Northwestern University*, Unpublished PhD Thesis, (1976).

84. PADBERG, M., Rinaldi, G., Optimization of a 532 city symmetric traveling salesman problem by branch and cut , *Operations Research Letters , 6,* pp.1-7 , (1987)

85. PADBERG, M. and Rinaldi, G., A branch-and-cut algorithm for the resolution large-scale symmetric traveling salesman problem, *SIAM Review*, 33,(1991), 60-100.

86. PICARD, J.C., Queyranne, M., The Time-Dependent Traveling Salesman Problem and Its Application to the Tardiness Problem in One-Machine Scheduling, *Operations Research*, 26, (1978) 86-110.

87. RAMKUMAR, A.S, Ponnambalam, S.G., Jawahar, N., A new iterated fast local search heuristic for solving QAP formulation in facility layout design, *Robotics and Computer-Integrated Manufacturing* 25, (2009),620– 629.

88. RAVINDRA K.A., James B.O., Ashish, T., A greedy genetic algorithm for the quadratic assignment problem, *Computers & Operations Research* 27, (2000), 917-934.

89. SILIH, S., Zula, T., Kravanja, Z., Kravanja, S., MINLP Optimization of Mechanical Structures, University of Maribor, Faculty of Civil Engineering (2000).

90. SCHNEIDER, J., The time-dependent traveling salesman problem, *Physica A*, 314, (2002) 151–155.

91. STEWART, W.R., A computationally efficient heuristic for the traveling salesman problem, *Proceedings of the 13th Annual Meeting of Southeastern TIMS*, Myrtle Beach, SC, USA, (1977), 75–83.

92. TAILLARD E.D., A heuristic Column Generation Method for the Heterogeneous Fleet VRP, *RAIRO*, 33, (1999) 1-14.

93. TARANTILIS, C.D., Kiranoudis, C.T., Vassiliadis, V.S., A list based threshold accepting metaheuristic for the heterogeneous fixed fleet vehicle routing problem, *Journal of the Operational Research Society*, 54, (2003) 65-71.

94. TARANTILIS, C.D., Kiranoudis, C.T., Vassiliadis, V.S., A threshold accepting metaheuristic for the heterogeneous fixed fleet vehicle routing problem, *European Journal of Operational Research*, 152, (2004) 148-158.

95. TARANTILIS, C.D., Kiranoudis, C.T., A flexible adaptive memory-based algorithm for real-life transportation operations, Two case stduies from dairy and construction sector, *EJOR,* 179, (2007) 806-822.

96. TIRPAK, T.M., Design to manufacturing information management for electronics assembly, International Journal of Flexible Manufacturing Systems, 12, (2000) 189-205.

97. WATERS, C.D.J., A Solution Procedure for the Vehicle-Scheduling Problem Based on Iterative Route Improvement, *The Journal of the Operational Research Society*, 38, (1987) 833-839.

98. WIEL, R.J.V., Sahinidis, N.V., Heuristic Bounds and Test Problem Generation for the Time Dependent Traveling Salesman Problem, *Transportation Science*, 29, (1995) 167-183.

99. WIEL, R.J.V., Sahinidis, N.V., An exact solution approach for the time-dependent traveling-salesman problem, *Naval Research Logistics*, 43, (1996) 797-820.

100. WINSTON, W.L., Operations Research, Applications and Algorithms, *Duxbury Press*, (1994).

101. WINSTON, W.L., Venkataramanan, M., Introduction to Mathematical Programming, Operations Research, Brooks/Cole-Thomson Learning, California, (2003)

102. YIP, P.P., ve Pao, Y., A guided evolutionary simulated annealing approach to the quadratic assignment problem, *IEEE transactions on systems,man, and cybernetics*, 24, 9, (1994).

103. YUAN, P.,Hu, Y.,Liu H.,Gao.,H. Scatter search algorithm for multi-headed Mounter, EBSCO Publishing, (2003).

TÜBİTAK
PROJE ÖZET BİLGİ FORMU

Proje No:108M198
Proje Başlığı: **Elektronik Dizgi İşlemlerinin Eniyilenmesi Ve Değişken Maliyetli Seyyar Satıcı Problemi**
Proje Yürütücüsü ve Araştırmacılar: **Doç. Dr. Ekrem DUMAN** **Yrd. Doç. Dr. Ali Fuat ALKAYA**
Projenin Yürütüldüğü Kuruluş ve Adresi: **Doğuş Üniversitesi** **Acıbadem, Kadıköy, 34722 İstanbul .**
Destekleyen Kuruluş(ların) Adı ve Adresi: **Doğuş Üniversitesi**
Projenin Başlangıç ve Bitiş Tarihleri:15 Kasım 2008-15 Kasım 2010
Öz (en çok 70 kelime) **Bu projede baskılı devre kartları (BDK) dizgi işlemlerinde ortaya çıkan yeni bir gezgin satıcı problemi (GSP) üzerinde çalışmalar yapılmış, sıraya dayalı GSP adı verilen bu problemin önce matematiksel modeli geliştirilmiş daha sonra onu çözecek özgün yöntemler geliştirilmiştir. Bunun dışında, bu problemin görüldüğü iki tip dizgi makinesinin diğer ilgili optimizasyon problemleri (karesel atama problemi - KAP) de çözülerek toplam dizgi süreleri enazlanmıştır. KAP ve problemin bütününün çözümlerinde metasezgisel yöntemlerden yararlanılmıştır.**
Anahtar Kelimeler: Gezgin satıcı problemi, sıraya dayalı gezgin satıcı problemi, karesel atama problemi, dizgi optimizasyonu, metasezgiseller

Fikri Ürün Bildirim Formu Sunuldu mu? **Evet** ☐ **Gerekli Değil** ☐

Fikri Ürün Bildirim Formu'nun tesliminden sonra 3 ay içerisinde patent başvurusu yapılmalıdır.

Projeden Yapılan Yayınlar:

- Alkaya, A.F., Duman, E.: "A New Generalization of the Traveling Salesman Problem", Applied and Computational Mathematics, 9, (2), (2010), 162-175. (SCI Expanded)
- Alkaya, A.F.; Duman, E.: "A Literature Survey of the Operation Optimization in Chip Shooter Placement Machines", *Proceedings of PICMET'09*, Portland, OR, USA, August 2-6, (2009), 3296-3306.
- Kıyıcığı, B.M., Duman, E., Alkaya, A.F.: "Finding Best Performing Solution Algorithm for the QAP", Proceedings of IMS2010, Sarajevo, Bosnia Herzegovina, September 15-17, (2010).
- Demirkale, H., Duman, E., Alkaya, A.F.: "Exact and Metahueristic Approaches for Optimizing the Operations of Chip Mounter Machines", Proceeding of CISIM2010, Cracow, Poland, October 8-10, (2010), 120-125.
- Duman, E., Alkaya, A.F., Demirkale, H.: "Optimizing the Operations of Chip Shooter Machines", Proceedings of META2010, Djerba Island, Tunisia, October 27-31, (2010).