ESNEK KISITLAR TABANLI ÖBEKLEME

ELİF TUĞÇE ÖRS

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

AĞUSTOS 2011
ANKARA

Fen Bilimleri Enstitü onayı

_____

Prof. Dr. Ünver KAYNAK
Müdür

Bu tezin Yüksek Lisans derecesinin tüm gereksinimlerini sağladığını onaylarım.

_____

Doç. Dr. Erdoğan DOĞDU
Anabilim Dalı Başkanı

Elif Tuğçe ÖRS tarafından hazırlanan ESNEK KISITLAR TABANLI ÖBEKLEME adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

_____

Yrd. Doç. Dr. Osman ABUL
Tez Danışmanı

Tez Jüri Üyeleri

Başkan: Doç Dr. Erdoğan DOĞDU _____

Üye : Yrd. Doç. Dr. Pınar ŞENKUL _____

Üye : Yrd. Doç. Dr. Osman ABUL _____

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada orijinal olmayan her türlü kaynağa eksiksiz atıf yapıldığını bildiririm.

Elif Tuğçe ÖRS

Üniversitesi : **TOBB Ekonomi ve Teknoloji Üniversitesi**
Enstitüsü : **Fen Bilimleri Enstitüsü**
Anabilim Dalı : **Bilgisayar Mühendisliği**
Tez Danışmanı : **Yrd. Doç. Dr. Osman ABUL**
Tez Türü ve Tarihi : **Yüksek Lisans – Ağustos 2011**

Elif Tuğçe ÖRS

ESNEK KISITLAR TABANLI ÖBEKLEME

ÖZET

Öbekleme önemli bir insan aktivitesidir. Aynı özellikleri paylaşan nesne grupları insanların dünyayı algılamasında ve tanımlamasında önemli rol oynar. Öbekleme birçok uygulama alanına sahiptir. Gerçek dünya uygulamaları çeşitli kısıtları sağlayacak şekilde bir öbekleme analizini gerektirir. Fakat, özellikle çok boyutlu ve/veya hacimli veri kümeleri söz konusu olduğunda, yalnızca öbekleme parametrelerine dayanarak anlamlı öbekler yaratmak zor olabilmektedir. Bundan dolayı, birçok uygulamada kullanıcının seçimleri ve koyduğu kısıtların göz önüne alınması istenir. Bizim çalışmamızda esnek kısıtlar kullanılarak, kesişmeyen kısmi öbeklemelerin elde edilmesi hedeflenmiştir. Öbeklemelerin yerine getirilmesi ve verilen kısıtları sağlamada ne kadar başarılı olduğunu ölçmek için bulanık, olasılıksal ve ağırlıklı yarı halka modelleri kullanılmıştır. Optimizasyon için ise genetik algoritmalardan faydalanılmıştır. Bahsi geçen işlemlerin gerçekleştirilmesinde kullanılmak üzere Java programlama dili kullanılarak bir araç geliştirilmiştir. Geliştirilen araç esnek kısıtların tanımlanması, öbekleme algoritmalarının çalıştırılması, veri kümeleri ve öbeklemelerin görsel olarak gösterilmesi, sonuçların hesaplanması ve öbek doğrulama yöntemlerinin kullanılmasını sağlamaktadır. Bu araçtan faydalanılarak seçilen veri kümeleri üzerinde kullanıcı tanımlı esnek kısıtlarına göre anlamlı öbekler oluşturmaya çalışan deneysel çalışmalar da yapılmıştır. Deney sonuçları kapsamlı olarak sunulmuş ve sonuçlar analiz edilmiştir.

Anahtar Kelimeler: Veri Öbekleme, Veri Madenciliği, Öbekleme Analizi, Esnek Kısıtlar, Kısıtlarla Öbekleme, Genetik Algoritmalar.

University : **TOBB University of Economics and Technology**
Institute : **Institute of Natural and Applied Sciences**
Science Programme : **Computer Engineering**
Supervisor : **Asst. Prof. Dr. Osman ABUL**
Degree Awarded and Date : **M. Sc. – August 2011**

Elif Tuğçe ÖRS

SOFT CONSTRAINTS BASED CLUSTERING

ABSTRACT

Clustering is an important human activity. Object groups sharing the same characteristics have a significant role in human perception of the world. Clustering has many application areas. Real world applications demand for cluster analysis which satisfies various user/domain constraints. But, it becomes an important challenge to obtain meaningful clusters by solely tuning clustering parameters, especially when high dimensional and/or high volume data sets are considered. As a result, in many of such applications, user preferences and domain constraints should be taken into consideration. The objective with this work is to obtain disjoint partial clusterings by employing soft constraints. Fuzzy, probabilistic and weighted semi-rings are used to do the clustering and as well to assess the degree of soft constraints satisfaction. Genetic algorithms are used for optimization purposes. A tool, written in Java, is developed to implement what is considered. The tool has the capability of accepting/exploiting user defined soft constraints, executing clustering algorithms, displaying data sets and resulting clusterings, and calculating the clustering metrics and validity indices. The tool is experimentally evaluated on select datasets to obtain soft constraints based clusterings. To assess the performance, extensive experimental results are presented and analyzed.

Keywords: Data Clustering, Data Mining, Cluster Analysis, Soft Constraints, Constrained Clustering, Genetic Algorithms.

TEŞEKKÜR

Değerli bilgilerini benimle paylaşan ve hiçbir konuda benden yardımını esirgemeyen tez danışmanım Yrd. Doç. Dr. Osman ABUL'a, asistanlık yaptığım süre boyunca desteklerini esirgemeyen TOBB ETÜ Bilgisayar Mühendisliği Bölümü Öğretim Üyelerine teşekkür ederim. Öğrenim hayatım boyunca beni destekleyen anne ve babama, TOBB ETÜ asistanlarından değerli arkadaşım Seçkin Anıl ÜNLÜ'ye; Bilim, Sanayi ve Teknoloji Bakanlığı Metroloji ve Standardizasyon Genel Müdürlüğü Takograf Şube Müdürü Hasan H. MUTLU'ya çalışmam boyunca bana verdikleri destek için teşekkür ederim.

İÇİNDEKİLER

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

ŞEKİLLERİN LİSTESİ

KISALTMALAR

Bu çalışmada kullanılmış olan kısaltmalar açıklamaları ile birlikte aşağıda sunulmuştur.

Kısaltmalar	Açıklama
CL	Cannot Link kısıtı
EKGA	Esnek Kısıtlar GA
EKGA ÖS	Esnek Kısıtlar GA (Örnek Seviyesinde)
EKGA ÖS Ç	Esnek Kısıtlar GA (Örnek Seviyesinde Çöp öbekli)
GA	Genetik Algoritma
JGAP	Java Genetic Algorithms and Programming kütüphanesi
IDE	Integrated Development Environment
ML	Must Link kısıtı
MUTGA	Minimum Uzaklıklar Toplamı Genetik Algoritması
PCA	Principal Component Analysis yöntemi
SSE	Sum of the Squared Error

SEMBOL LİSTESİ

Bu çalışmada kullanılmış olan simgeler açıklamaları ile birlikte aşağıda sunulmuştur.

Simgeler	Açıklama
c	Kısıt sayısı
D	Veri kümesi
I	İterasyon sayısı
K	Öbek sayısı
m	Nokta sayısı
n	Veri boyutu
P	Öbekleme
α	Kısıt esnekliği eğimi
δ	İki veri noktası arasındaki uzaklık
π	Öbek populasyonu
σ	Büyüklük (Öbek sayısı)
τ	Çöp sayısı

1. GİRİŞ

"Hiçbir insanın bilgisi öğrenmiş ve görmüş geçirmiş olduğunun ötesine geçemez."

John Locke

An Essay Concerning Human Understanding

Öbekleme ve öbek analizi önemli bir insan aktivitesidir. Aynı özellikleri paylaşan nesne grupları, insanların dünyayı algılamasında ve tanımlamasında önemli rol oynar. İnsanlar, zaman geçtikçe nesneleri gruplara ayırma -öbekleme- ve nesneleri bu gruplara atama -sınıflandırma- konusunda yetenek kazanırlar. Çocukluğun erken dönemlerinde genellikle farkında olmadan kazanılan bu yetenek sayesinde kedileri ve köpekleri, binaları ve araçları, insanları ve hayvanları sürekli gelişen bir bilinçaltı öbekleme tasarısıyla ayırt etmeyi öğrenirler. Sonuç olarak, verinin anlaşılmasında öbekler potansiyel sınıflardır ve öbekleme analizi de bu sınıfların otomatik olarak bulunması için kullanılan tekniklerin tümüne birden verilen isimdir [1, 2].

Öbekleme biyoloji, tıp, antropoloji, pazarlama ve ekonomi gibi birçok uygulama alanına sahiptir. Öbekleme uygulamaları bitki ve hayvan sınıflandırma, hastalık sınıflandırma, görüntü işleme, örüntü tanıma ve dokümanlardan bilgi elde etme gibi uygulamaları içerir. İlk kullanıldığı alanlardan birisi topolojik taksonomidir. Son kullanımları ise kullanım sıklıklarına göre örüntülerin tespiti için web kayıt verilerinin incelenmesini kapsar [3, 4].

Öbekleme analizinin uygulama alanlarından bazıları genel hatlarıyla aşağıda açıklanmıştır:

- Biyoloji: Biyologlar benzer fonksiyonlara sahip gen gruplarını bulmak için ellerindeki fazla miktarda genetik bilgiye öbekleme analizi uygular.
- Arama Motorları: Öbekleme analizi Google, Yahoo, vb. arama motorlarında sıkça kullanılan bir tekniktir. Analiz sayesinde elde edilen bilgi alt öbeklere bölünerek kullanılır. Sorgu ifadeleri değiştirilerek elde edilen hiyerarşik bilginin sadece ilgi çeken kısımları alınır ve kullanılır.

- İklim: Artan küresel ısınmayla birlikte iklim değişikliklerinin takibi de önem kazanmaktadır. Kutup bölgelerindeki atmosfer basınçlarında ve okyanusun kara iklimine önemli derecede etkide bulunan alanlarındaki örüntülerin tespit edilmesinde öbekleme analizinden faydalanılır. Böylece Dünya'nın içinde bulunduğu iklim koşulları hakkında bilgi edinilir.
- Psikoloji: Çeşitli depresyon tiplerinin tespitinde örüntü analizi kullanılır.
- Tıp: Hastalığın yerel ve zamansal dağılımındaki örüntüleri tespit etmek için kullanılır.
- Pazarlama: Müşterileri çeşitli analizler ve market aktiviteleri için gruplara bölmede kullanılır.

Gerçek dünya uygulamaları çeşitli kısıtları sağlayacak şekilde bir öbekleme analizini gerektirir. Fakat özellikle çok boyutlu veriler söz konusu olduğunda yalnızca öbekleme parametrelerine dayanarak anlamlı öbekler yaratmak çok zordur. Öbekleme analizi benzerlik/uzaklık fonksiyonları tarafından gerçekleştirilse de kullanıcılar genellikle uygulama gereksinimleri hakkında net bir fikre sahiptirler. Bu fikir öbekleme analizinin yönünü belirlemede ve öbeklemenin sonucunu etkilemede rol oynayabilir [5]. Bundan dolayı birçok uygulamada kullanıcının seçimleri ve koyduğu kısıtların göz önüne alınması istenir.

Bu çalışmada da kullanıcı kısıtlarına göre anlamlı öbekler oluşturmaya çalışan öbekleme algoritmaları bulmak hedeflenmiştir. Bulunan algoritmaların hepsi esnek kısıt-tabanlı, kesişmeyen, kısmi bir öbekleme elde etmeyi amaçlamaktadır. Bu öbekleme tipini daha detaylı incelemek için öbekleme analizinden ve öbekleme türlerinden bahsedilmesi gerekmektedir.

2. ÖBEKLEME ANALİZİ

Sınıflandırma, nesneleri gruplara ayırmada etkili bir yöntem olsa da genellikle pahalı bir maliyetle etiketlenmiş örüntülere gereksinim duyar. Fakat genellikle bunun tam tersi istenmektedir; yani önce benzerliklerine göre veriyi gruplara ayırmak daha sonra bu az sayıdaki grubu etiketlemek. Verilerin gruplanması açısından öbekleme, sınıflandırma ile benzerdir fakat sınıflandırmada olduğu gibi gruplar önceden belirlenmemiştir. Bunun yerine gruplama asıl verideki karakteristiklere göre belirlenen benzerlikler sayesinde gerçekleştirilir [2, 3] . Bu grupların her birine öbek denir.

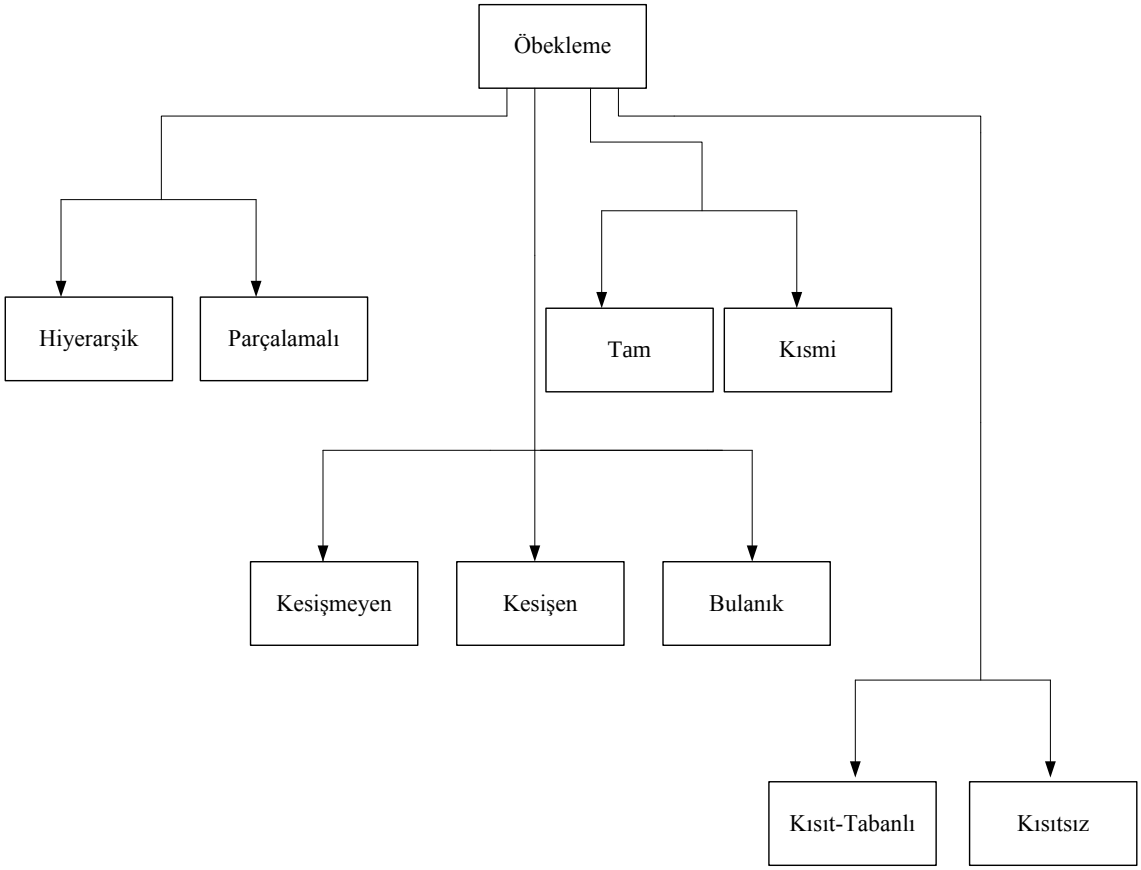
Çeşitli öbek tanımları bulunmaktadır. Bunlardan bazıları aşağıda verilmiştir:

- Benzer elemanların kümesi. Farklı öbeklerdeki elemanlar benzer değildir.
- Bir öbekte bulunan noktalar arasındaki uzaklık, bu öbekteki bir nokta ile başka öbekteki bir nokta arasında bulunan uzaklıktan daha azdır.
- Anlamlandırılabilir ve/veya yararlanılabilir verilerden oluşan kümedir.
- Diğerleriyle kıyaslandığında aynı gruptaki verilerin birbiriyle daha yüksek benzerliğe, fakat diğer gruplardaki verilerle çok az benzerliğe sahip olduğu kümedir.

Birçok uygulamada öbek kavramı iyi tanımlanmamıştır. Bu durum bir öbeği neyin (hangi verilerin) oluşturduğuna karar vermenin güçlüğünden kaynaklanmaktadır. Bir öbeği kesin olarak tanımlamak zordur çünkü en iyi tanım verinin yapısına ve istenen sonuçlara bağlı olarak değişir. Öbeklerin hepsine birden öbekleme denilmekte ve öbekleri elde etmek için öbekleme analizinden faydalanılmaktadır.

2.1. Öbekleme Çeşitleri

Öbeklerin meydana getirdiği öbeklemenin Şekil 2.1'de görüldüğü üzere birçok kıstasa göre çeşidi bulunmaktadır: hiyerarşik veya parçalamalı, kesişen veya kesişmeyen, tam veya kısmi, kısıtlı veya kısıtsız.



Şekil 2.1. Öbekleme yaklaşımları için bir taksonomi

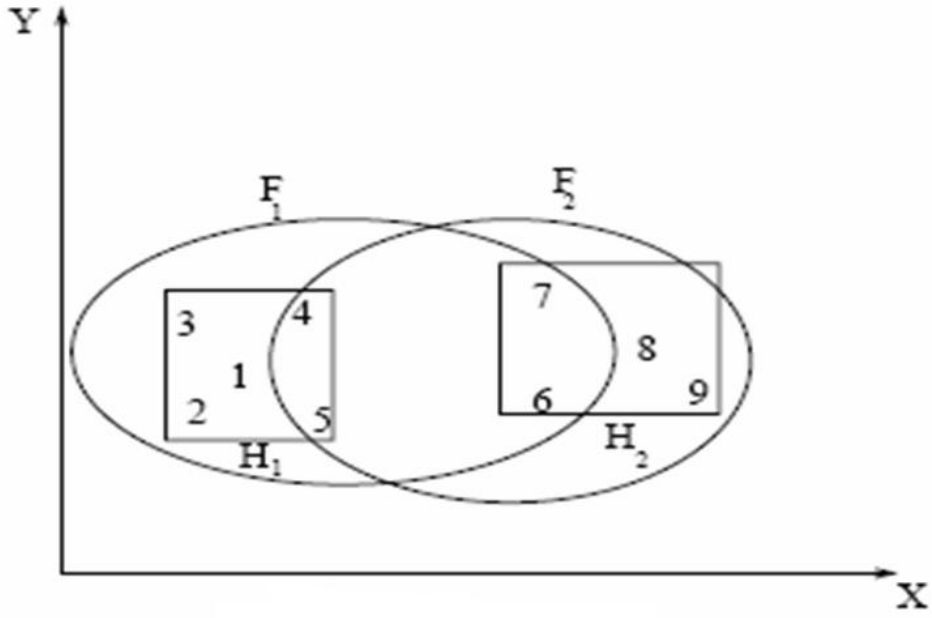
2.1.1. Hiyerarşik veya Parçalamalı

Öbeklerin alt-öbeklere sahip olmasına izin verilmişse bu öbekleme tipine hiyerarşik öbekleme denir.

Veri nesneleri, her nesne sadece bir öbeğin elemanı olacak şekilde kesişmeyen alt kümelere bölünmüşse bu öbekleme tipine parçalamalı öbekleme denir.

2.1.2. Kesişmeyen veya Kesişen

Her nesnenin tek bir öbeğe ait olduğu öbekleme tipine kesişmeyen, bir nesnenin birden fazla öbeğin elemanı olabildiği öbekleme tipine kesişen öbekleme denir. Bu iki öbeklemeden farklı olarak bulanık öbeklemede ise bir nesnenin hangi öbeğe ait olduğu bir üyelik fonksiyonu ile belirlenir. Bulanık bir öbekleme örneği Şekil 2.2'de görülmektedir. Şekilde H_1 ve H_2 kesişmeyen F_1 ve F_2 ise kesişen bulanık öbekleri göstermektedir.



Şekil 2.2. Kesişmeyen ve Kesişen (Bulanık) öbekleme [4]

2.1.3. Tam veya Kısmi

Her nesneyi mutlaka bir öbeğe atayan öbeklemeye tam, bazı nesneleri herhangi bir öbeklemeye atamayan öbeklemeye ise kısmi öbekleme denir.

2.1.4. Kısıtlı veya Kısıtsız

Kullanıcının tanımladığı veya uygulamanın tipine yönelik kısıtlara dayalı öbekler yaratan öbeklemeye kısıt tabanlı öbekleme denir.

Eğer öbekleme için kısıt verilmediyse gerçekleştirilen öbeklemeye kısıtsız öbekleme denir.

2.2. Temel Öbekleme Algoritmaları

Öbeklemeler elde edilirken öbekleme analizi kullanılmaktadır. Oluşturulacak öbeklemenin karakteristiği nasıl olursa olsun öbekleme analizinin tüm öbeklemeler için geçerli olan temel özellikleri şunlardır:

- En iyi öbek sayısı bilinmez.
- Öbeklerle ilgili bir ön bilgi bulunmayabilir.
- Öbek sonuçları dinamiktir.

Öbekleme analizini gerçekleştirmek için kullanılan çeşitli algoritmalar vardır. Bu algoritmalardan biri parçalamalı öbekleme yaklaşımına giren ve basitliğinden dolayı sıkça kullanılan K-means [6] algoritmasıdır.

2.2.1. K-means

K-means algoritması, veri kümesini önceden sabit olarak belirlenmiş k adet öbek olacak şekilde sınıflandıran bir algoritmadır. Her öbek için bir merkez belirlenmesi esastır. Bu merkezlerin seçimi önemli bir noktadır. Merkez seçimi akıllıca gerçekleştirilirse algoritmanın verdiği sonuç da iyi olacaktır. Burada akıllıca seçimden kasıt merkezlerin birbirlerinden mümkün oldukça uzak seçilmesidir. Merkez olarak seçilen noktaların veri kümesinin elemanı olması gibi bir koşul aranmamaktadır. Merkez seçiminden sonra veri kümesindeki her bir nokta ile seçilen her bir merkez arasındaki uzaklığa tek tek bakılır ve nokta kendisine en yakın olan merkezin ait olduğu öbeğin etiketini alır. Tüm noktalar etiketlendikten sonra bu noktaların ortalamaları hesaplanır ve elde edilen sonuçlar yeni öbek merkezleri olarak atanır. Noktaların öbeklere atanması ve öbek merkezlerinin yeniden hesaplanması belli bir eşik değerinin -her bir öbekte bulunan noktaların öbeğin

merkezine olan uzaklıklarının kareleri toplamı (*sum of the squared error, SSE*)-altına inene kadar devam eder. Eşik değerinin altına inilince algoritma sonlandırılır. Bu noktada algoritmanın belli bir eşik değerine yakınsaması şart değildir. Algoritma önceden belirlenen iterasyon sayısı kadar da çalıştırılıp sonlandırılabilir.

K-means basit bir çalışma prensibine sahip olduğu gibi algoritma için sadece verileri ve öbek merkezlerini saklamak yeterli olduğundan yer ve zaman karmaşıklığı bakımından da makul bir algoritmadır. *m* nokta sayısı, *n* boyut (özellik) sayısı, *K* öbek sayısı ve *I* iterasyon sayısını göstermek üzere gereken yer miktarı *O((m+K)n)* çalışma zamanı ise *O(IxKxmxn)* kadardır.

2.2.2. Fuzzy C-Means

Bir diğer parçalamalı algoritma da k-means algoritmasının bulanık versiyonu olan fuzzy c-means algoritmasıdır. Bu algoritmada bir veri birden fazla öbeğe ait olabilir. Verinin bir öbeğe ait olma derecesi üyelik fonksiyonu ile belirlenir. K-means algoritmasında da olduğu gibi önce öbek sayısı belirlenir. Öbek sayısı belirlendikten sonra k-meansten farklı olarak her bir veriye rastgele ağırlık değerleri atanır. Burada sözü geçen ağırlıklardan kasıt verinin öbeklere ait olma dereceleridir. Ağırlık değerleri en az 0 en fazla 1 olabilir. Öbeğe ait olma dereceleri öbek merkezine olan uzaklığın tersi ile şu şekilde ilişkilidir:

$$u_k(x) = \frac{1}{d(center_k, x)} \tag{2.1}$$

Daha sonra katsayılar normalleştirilir ve $m > 1$ parametresi ile bulanıklaştırılır:

$$u_k(x) = \frac{1}{\sum_j \left(\frac{d(center_k, x)}{d(center_j, x)}\right)^{2/(m-1)}} \tag{2.2}$$

Tüm noktalara ağırlık değerleri atandıktan sonra öbek merkezleri hesaplanır. Bir öbeğin merkezi öbeğe ait olma değerlerinin aritmetik ortalamasıyla hesaplanır:

$$center_k = \frac{\sum_x u_k(x)^m x}{\sum_x u_k(x)^m} \tag{2.3}$$

Merkezler hesaplanıp güncellendikten sonra noktaların yeni merkezlere olan uzaklıklarına göre öbeklere atanması gerçekleştirilir. K-means algoritmasında olduğu gibi belli bir iterasyon sayısınca veya önceden belirlenen eşik değerinin altına inilene kadar son merkezlerin güncellenmesi ve noktaların öbeklere atanması işlemleri tekrar edilir [7].

Bu çalışmada kesişmeyen bir öbekleme elde etmek amaçlandığından fuzzy c-means algoritması da kesişen öbeklemeler elde ettiğinden algoritmada ufak bir değişiklik yapılması gerekmiştir. Algoritma çalışmasını tamamladıktan ve son ağırlık değerleri belli olduktan sonra her bir nokta en çok ait olduğu öbeğe atanmıştır. Böylece her nokta tek bir öbek numarasıyla etiketlenmiş dolayısıyla da kesişmeyen öbeklemeler elde edilmiş olur.

2.3. Öbekleme Problemleri

Öbekleme gerçek-dünya veri tabanına uygulandığında birçok ilginç problem ortaya çıkar:

- Aykırı değerleri (*outliers*) kontrol etmek güçtür. Bu değerler hiçbir öbeğe ait olmazlar. Tek başlarına birer öbek olarak kabul edilebilirler. Fakat öbekleme algoritması daha büyük öbekler bulmaya teşebbüs ederse aykırı değerler bazı öbeklere ait olmaya zorlanacaktır. Bu işlem var olan iki öbeği birleştirerek ve aykırı öbeği kendi öbeğinde bırakarak kalitesiz öbekler yaratılmasına sebep olur.
- Veri tabanındaki dinamik veri öbek ilişkisinin zamanla değişebileceğini gösterir.

- Her öbeğin semantik anlamını yorumlamak zor olabilir. Sınıflandırmada, sınıfların etiketleri önceden bilinmektedir, öbeklemedeyse etiketler önceden bilinmez. Öbekleme süreci, bir öbekler kümesi yaratmayı bitirdiğinde, her bir öbeğin tam anlamı kesin olmayabilir. Bu noktada, her bir öbeği etiketlemek veya yorumlamak için alanında uzman bir kişi gerekir.
- Öbekleme probleminin tek bir doğru cevabı yoktur. Aslında birçok cevap bulunabilir. Gerekli öbek sayısını tam olarak kestirmek kolay değildir. Yine uzman bir kişinin yardımına ihtiyaç duyulur.
- Boş öbekler veya belli bir eleman sayısından daha az sayıda elemana sahip öbekler oluşabilir [3, 8].

Gerçek dünya uygulamalarında araştırmacı genellikle oluşması gereken (oluşması beklenen) öbeklemeyle ilgili önbilgiye sahiptir. Yukarıda sözü geçen geleneksel öbekleme algoritmalarının önbilgi avantajından yararlanma imkânı yoktur [9]. Gerçek uygulamalarda karşılaşılan problemlerin geliştirilmesinde kullanıcı tarafından verilen kısıtlar etkili bir rol oynamaktadır öyle ki girilen kısıtların birçok öbekleme algoritmasının sonuçlarını geliştirdiği görülmüştür [10]. Bu nedenle kullanıcının kısıtlar koyarak oluşacak öbeklemeyi şekillendirmesi daha anlamlı sonuçlar elde edilmesine yardımcı olmaktadır. Böylece öbekleme problemleri belli ölçüde giderilebilmektedir.

2.4. Kısıtlarla Öbekleme

Öbekleme algoritmaları genellikle denetimsiz (*unsupervised*) algoritmalardır. Fakat bazı durumlarda veri örneklerine ek olarak problem tanımıyla ilgili ek bilgiler de bulunabilir. Bu bilgilerden öbekleme sonuçlarını geliştirmek ve gerçek dünya problemlerinden anlamlı sonuçlar çıkarmaya yönelik çeşitli yöntemler geliştirilmiştir. Bu yöntemlerden bir tanesi kullanıcıdan geribildirim alarak öbekleme yapmaktır. Geribildirim öbekleme algoritmasının sonraki iterasyonlarında sağlamaya çalıştığı kısıtlar olarak gönderilir [5]. Bir diğer yöntem temel öbekleme algoritmalarını değiştirerek verilen kısıtları gerçekleştirebilecek yeni algoritmalar

üretmektir. Etiketli veriler başlangıç öbeklerini oluşturmak için kullanıldıktan sonra k-means gibi algoritmaların amaç fonksiyonları değiştirilerek öbekleme işlemini gerçekleştiren algoritmalar türetilir. Burada amaç noktaları, kısıtları mümkün olduğunca ihlal etmeden öbeklere atamaktır [11, 12]. Hem etiketsiz (*unlabeled*) hem de etiketli (*labeled*) verilerden faydalanan algoritmalara yarı denetimli (*semi-supervised*) algoritmalar denilmektedir.

Yarı denetimli algoritmalarda kullanılan kısıtları örnek seviyesinde (*instance-level*) ve öbek seviyesinde (*cluster-level*), katı (*hard*) ve esnek (*soft*) kısıtlar olmak üzere iki grupta incelemek mümkündür. Yapılan çalışmada kullanılan kısıtlar esnek kısıtlardır. Esnek kısıtlarla örnek seviyesinde kısıtlar da öbek seviyesinde kısıtlar da birleştirilerek öbeklemeler elde edilebilir. Elde edilen öbeklemelerden son kısımda detaylı olarak bahsedilecektir. Bu bölümde ise kısıt tabanlı algoritmalarda kullanılan, bahsi yukarıda geçen kısıt tiplerinden bahsedilecektir.

3. KISITLAR

3.1. Kısıt Türleri

Yarı denetimli öbekleme denetimsiz öğrenmeye yardımcı olmak için az miktarda etiketli veri kullanır. Etiketler verinin hangi öbeğe ait olduğunu gösterir. Etiketli verilerden faydalanmak için örnek seviyesinde kısıtlar kullanılır. Örnek seviyesinde kısıtlardan başka bir de öbek yapısının genel özelliklerini kısıt olarak getiren öbek seviyesinde kısıtlar bulunmaktadır [11, 13]. Bahsedilen iki tip kısıt da katı veya esnek kısıtlarla birleştirilerek kullanılabilir.

3.1.1. Örnek Seviyesinde Kısıtlar

Örnek seviyesinde kısıtlar must-link (*ML*) ve cannot-link (*CL*) kısıtları olmak üzere iki gruba ayrılırlar. Bu kısıtlar, hangi verilerin birlikte gruplanıp gruplanmayacağını ön bilgi olarak belirtmede yararlıdır.

- ML kısıtları aynı öbekte olması gereken iki veriyi tanımlar.
- CL kısıtları aynı öbekte olmaması gereken iki veriyi tanımlar.

ML kısıtları veriler üzerinde ikili geçişli bir ilişki tanımlar. Böylece kısıt kümelerini kullanırken kısıtlar üzerinde kapalı geçişlilikten söz edebiliriz.

Kısıtlarla yapılan birçok çalışmada örnek-seviyesinde kısıtlar kullanılmaktadır. Wagstaff ve Cardie bir çalışmalarında COBWEB algoritmasını kısıtlarla birleştirerek daha iyi sonuçlar elde etmişlerdir [14]. Wagstaff ve arkadaşları başka bir çalışmalarında örnek seviyesinde kısıtlardan faydalanarak kısıtlı bir k-means algoritması türetmiştir [9]. Wagstaff ve arkadaşları bir diğer çalışmalarında ise örnek seviyesinde kısıtlar kullanılmasının öbekleme sonuçlarını geliştirdiğini tutarsızlık (*inconsistency*) ve uyumsuzluk (*incoherence*) kavramlarından faydalanarak göstermişlerdir [10]. Sözü geçen çalışmaların hepsinde kısıtlarla gerçekleştirilen öbeklemelerin kısıtlar olmadan gerçekleştirilen öbekleme sonuçlarından daha iyi sonuç verdiği gözlenmiştir.

3.1.2. Öbek Seviyesinde Kısıtlar

Bir öbekteki minimum eleman sayısı, öbeğin maksimum çapı gibi veri grupları hakkındaki bilgiler öbek seviyesinde kısıtlar şeklinde modellenebilir [15]. Bradley ve arkadaşları öbek seviyesinde kısıtlardan faydalanarak veri boyutunun ve istenen öbek sayısının fazla olduğu durumlarda boş öbekler oluşmasını engelleyen bir algoritma bulmuşlardır [8].

Bu çalışmada esnek kısıtlar kullanılarak kesişmeyen, kısmi öbeklemeler elde etmek amaçlanmaktadır. Kısmi öbeklemelerde bazı nesneler hiçbir öbeğe ait olmayabilirler. Hiçbir öbeğe ait olmayan bu nesneler de kendi aralarında özel bir öbek oluşturmaktadırlar. Bu öbeğe çöp öbeği (*trash cluster*) denilmektedir.

3.2. Kesişmeyen Kısmi Öbekleme

D; $X_1, X_2, \ldots X_{|D|}$ noktalarından oluşan bir küme ve $P=\{p_1,\ldots,p_{n+1}\}$ ise D kümesinin kesişmeyen bir kısmi öbeklemesi olsun. Burada p_{n+1} özel bir öbek olup çöp öbeğini göstermektedir. Buna göre tüm öbeklerin birleşimi veri kümesine yani D'ye eşittir ve P öbeklemesinde bulunan herhangi iki öbeğin kesişimi boş kümedir. Eğer p_{n+1} boş değilse bu öbekleme kısmidir.

Bir veri kümesi D ve D'nin bir öbeklemesi olan P tanımlı olsun. Bu öbekleme üzerinde tanımlayabileceğimiz minimum populasyon, maksimum çap, minimum yoğunluk gibi çeşitli kısıtlar bulunmaktadır.

3.2.1. Minimum Populasyon *(Min Pop)*

Populasyon bir öbekte bulunan nesne sayısıdır. π, pozitif bir tam sayı olsun ve herhangi bir öbekleme algoritması çalıştırılıp öbekleme elde edilmeden belirlensin. Bir öbeklemede bulunan her bir öbeğin populasyonu π'ye eşit veya π'den büyükse minimum populasyon kısıtı sağlanmış olur.

3.2.2. Maksimum Populasyon *(Max Pop)*

π, pozitif bir tam sayı olsun ve herhangi bir öbekleme algoritması çalıştırılıp öbekleme elde edilmeden belirlensin. Bir öbeklemede bulunan her bir öbeğin populasyonu π'ye eşit veya π'den küçükse maksimum populasyon kısıtı sağlanmış olur.

3.2.3. Minimum Çap *(Min Diam)*

Bir öbekte bulunan tüm noktalar içinde biribirine en uzakta bulunan iki nokta arasındaki uzaklık çaptır. x_u ve x_v aynı öbekte biribirinden en uzakta bulunan iki nokta ve δ, pozitif bir reel sayı olsun. Bir öbeklemede bulunan her bir öbeğin x_u ve x_v noktaları arasındaki uzaklığı (çapı) δ'dan büyükse minimum çap kısıtı sağlanmış olur.

3.2.4. Maksimum Çap *(Max Diam)*

x_u ve x_v aynı öbekte biribirinden en uzakta bulunan iki nokta ve δ, pozitif bir reel sayı olsun. Bir öbeklemede bulunan her bir öbeğin x_u ve x_v noktaları arasındaki uzaklığı (çapı) δ'dan küçükse maksimum çap kısıtı sağlanmış olur.

3.2.5. Minimum Öbekler Arası Uzaklık *(Min Icd)*

Bir öbeklemede bulunan herhangi iki farklı öbeğe ait herhangi iki nokta arasındaki en küçük uzaklık öbekler arası uzaklıktır. n, bir öbeklemedeki öbek sayısını göstermek üzere i ve j, *1*'den n'e kadar n dahil bir tam sayı; $i \neq j$; x_u, p_i öbeğine ait bir nokta x_v'de p_j öbeğine ait bir nokta ve δ, 0'dan büyük bir reel sayı olsun. Bir öbeklemede bulunan olası tüm x_u ve x_v ikilileri arasındaki en küçük uzaklık (öbekler arası uzaklık) δ'dan büyükse minimum öbekler arası uzaklık kısıtı sağlanmış olur.

3.2.6. Maksimum Öbekler Arası Uzaklık *(Max Icd)*

n, bir öbeklemedeki öbek sayısını göstermek üzere *i* ve *j*, 1'den *n*'e kadar *n* dahil bir tam sayı; $i \neq j$; x_u, p_i öbeğine ait bir nokta x_v de p_j öbeğine ait bir nokta ve δ, 0'dan büyük bir reel sayı olsun. Bir öbeklemede bulunan olası tüm x_u ve x_v ikilileri arasındaki en küçük uzaklık (öbekler arası uzaklık) δ'dan küçükse maksimum öbekler arası uzaklık kısıtı sağlanmış olur.

3.2.7. Minimum Çöp *(Min Trash)*

Hiçbir öbeğe ait olmayan toplam eleman sayısı çöptür. τ, pozitif bir tam sayı olsun. Bir öbeklemede bulunan çöp sayısı τ'dan büyükse minimum çöp kısıtı sağlanmış olur.

3.2.8. Maksimum Çöp *(Max Trash*)

Hiçbir öbeğe ait olmayan toplam eleman sayısı çöptür. τ, pozitif bir tam sayı olsun. Bir öbeklemede bulunan çöp sayısı τ'dan küçükse maksimum çöp kısıtı sağlanmış olur.

3.2.9. Minimum Büyüklük *(Min Size)*

Bir öbeklemedeki öbek sayısı büyüklüktür. σ, pozitif bir tam sayı olsun. Bir öbeklemede bulunan öbek sayısı σ'dan büyükse minimum büyüklük kısıtı sağlanmış olur.

3.2.10. Maksimum Büyüklük *(Max Size)*

Bir öbeklemedeki öbek sayısı büyüklüktür. σ, pozitif bir tam sayı olsun. Bir öbeklemede bulunan öbek sayısı σ'dan küçükse maksimum büyüklük kısıtı sağlanmış olur.

3.2.11. Diğer Kısıtlar

Yukarıdaki kısıtlardan başka ilgilenilen bakış açısına göre daha pek çok kısıt tanımı yapılabilir. Buna örnek olarak toplam birleşme (*total cohesion*) kısıtı verilebilir. Toplam birleşme bir öbekte bulunan tüm noktalar arasındaki uzaklıkların toplamıdır.

Birkaç kısıt birleştirilerek yeni bir kısıt tanımı da yapılabilir. Buna örnek olarak minimum yoğunluk kısıtını verebiliriz. Minimum yoğunluk minimum populasyon ve maksimum çap kısıtlarının birleşiminden oluşmaktadır.

Minimum populasyon ve maksimum çap birbiriyle çelişen (*conflicting*) iki kısıttır. Minimum yoğunluk kısıtının sağlanması için iki kısıtın da aynı anda sağlanması gerekmektedir. Bir öbekleme algoritmasına minimum yoğunluk kısıtında olduğu gibi birbiriyle çelişen kısıtlar vermek, katı kısıtılarla çalışıldığında problemi sonuçsuz bırakmaktadır. Bunun yerine esnek kısıtlar kullanmak problemin sonucuna öbekleme algoritmalarındaki gibi bir amaç fonksiyonu (*objective function*) yaklaşımını getirerek daha etkili bir sonuç elde edilmesini sağlar.

Katı kısıtlar, yalnızca 'doğru' veya 'yanlış' değerlerinden birini alırlar. 'Doğru' kabul edilir, 'yanlış' ise kabul edilmez anlamına gelmektedir. Esnek kısıtlar, kısıtın ne kadar sağlandığıyla ilgili bir değer alırlar. Bir veya birden çok kısıtın sağlanması ve bu kısıtların sağlanması bakımından öbeklemelerin karşılaştırılması işlemlerinin matematiksel gösterimi için yarı halkaların (*semi-rings*) kullanımı uygundur [16].

3.3. Öbekleme Kısıtları

Esnek kısıtların öbeklemeler üzerinde kullanılmasında, verilen kısıtların ne kadar sağlandığının hesaplanmasında ve kısıtlar sağlanarak elde edilen öbeklemelerin karşılaştırılmasında yarı halkalar kullanılır. Yarı halkalar $S = < A, +, \times, 0, 1>$ biçiminde modellenir. Bu beşlideki A, yarı halkanın alabileceği tanımlı değerlerin kümesini; $+$, yarı halkanın en büyük değerini almasını sağlayan operatörü; $\times$, yarı halka değerinin hesaplanma yöntemini belirleyen operatörü; 0 ve 1 ise sırasıyla A kümesinin alabileceği alt ve üst değerleri ifade etmektedir.

Üç çeşit yarı halka tipi kullanılarak üç çeşit hesaplama gerçekleştirilebilir. Bu yarı halka tipleri bulanık, olasılıksal ve ağırlıklı yarım halkadır.

3.3.1. Bulanık Yarı Halka (*Fuzzy Semi-ring*)

Bulanık yarı halkalarda minimum değer 0, maksimum değer 1'dir. Kısıtlarla gerçekleştirilen her öbekleme bu kapalı aralıkta bir değer alır. Bulanık yarı halkada en az sağlanan kısıtın değerini arttırmak amaçlanır, bunu yaparken de tüm kısıtların seviyelerini dengeler. Buna bağlı olarak hesaplama sonucu en az sağlanan kısıtın değeridir. Bulanık yarı halka $S_F = <[0, 1], max, min, 0, 1>$ şeklinde ifade edilir.

3.3.2. Olasılıksal Yarı Halka (*Probabilistic Semi-ring*)

Olasılkısal yarı halkalarda minimum değer 0, maksimum değer 1'dir. Kısıtlarla gerçekleştirilen her öbekleme bu kapalı aralıkta bir değer alır. Olasılıksal yarı halkada hesaplama sonucu her bir kısıtın sağlanma değerinin birbirleriyle çarpımıdır. Olasılıksal yarı halka $S_P = <[0, 1], max, \times, 0, 1>$ şeklinde ifade edilir.

3.3.3. Ağırlıklı Yarı Halka (*Weighted Semi-ring*)

Ağırlıklı yarı halkalarda minimum değer 0, maksimum değer artı sonsuzdur. Kısıtlarla gerçekleştirilen her öbekleme bu kapalı aralıkta bir değer alır. Ağırlıklı yarı halkada tüm kısıtların toplamda sağlanma değerlerini arttırmak amaçlanır. Bazı kısıtlar büyük bir maliyet (*cost*) ödenerek ihmal edilse de global maliyeti minimum yapmayı hedefler. Buna bağlı olarak hesaplama sonucu her bir kısıtın sağlanma değerinin birbirleriyle toplamıdır. Ağırlıklı yarı halka $S_W = <R^+, min, sum, +\infty , 0>$ şeklinde ifade edilir.

3.4. Öbeklemede Esnek Kısıtların Kullanımı

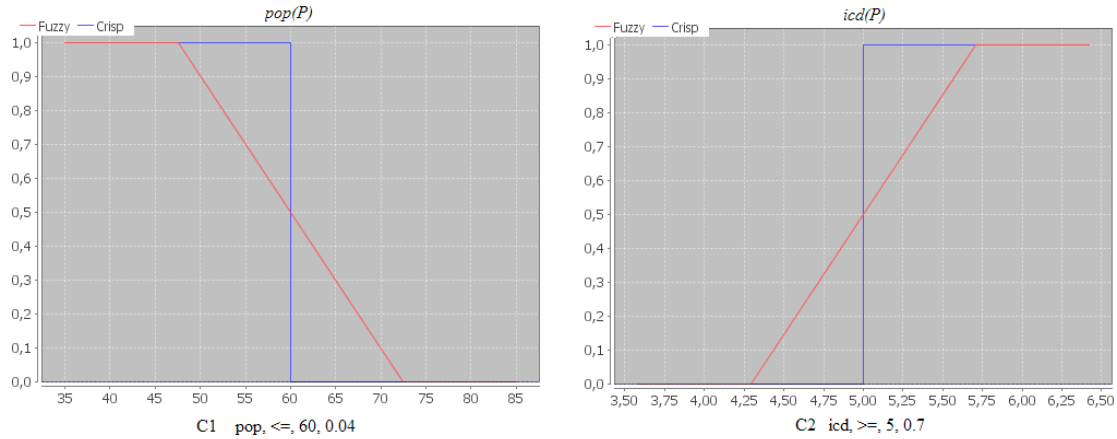
Esnek kısıtların öbekleme üzerinde kullanımı [16] bir dörtlüyle (*quadruple*) gösterilir. Buna göre kısıt tipi (öbekteki eleman sayısı, öbek çapı, vb.), kısıt aralığının yönünü belirleyen işaret (≤ veya ≥ işareti), kısıt aralığının merkezi (*t*, yarı halkanın 0,5 değerine karşılık gelen değer) ve kısıtın esnekliğini ayarlayan eğim (α) öbeklemede kullanılacak her bir kısıtın yaratılmasında kullanılan dört temel öğedir.

Kısıtları tanımlarken göz önünde bulundurulması gerekenler vardır. Bir kısıtın öbeklemedeki önemini arttırmak için eğimi azaltmak gerekir. Bununla birlikte eğim tek başına kısıtın esnekliğini ayarlasa da kısıtın önemini arttırıp azaltmada tek başına yeterli olmamaktadır. *t* değeri de kısıt önemini belirlemekte kullanılır. Yeterince iyi olmayan öbeklemelerde bir kısıtın değeri diğer kısıtlara göre arttırılırsa sonuç olarak düşük değerler elde edilir.

Şekil 3.1 'de verilen kısıtların ağırlıklı bir yarı halka modelindeki kombinasyonunu göz önünde bulunduralım. Bu kısıtlardan ilki pop (P) <= 60 [-0.04], ikincisi de icd (P) >= 5 [0.7] olarak verilmiştir. P_x ve P_y iki öbekleme olsun öyle ki; $pop(P_x) = 50$, $pop(P_y) = 65$, $icd(P_x) = 4.75$, $icd(P_y) = 5.5$'tir.

Bu durumda $C_1(P_x) = 0.9$; $C_2(P_x) = 0.325$; $C_1(P_x)$ x $C_2(P_x) = toplam$ (0.9, 0.325) = 1,225;

$C_1(P_y)$ = 0.3; $C_2(P_y)$ = 0.85; $C_1(P_y)$ x $C_2(P_y)$ = *toplam* (0.3, 0.85) = 1,115 olur.



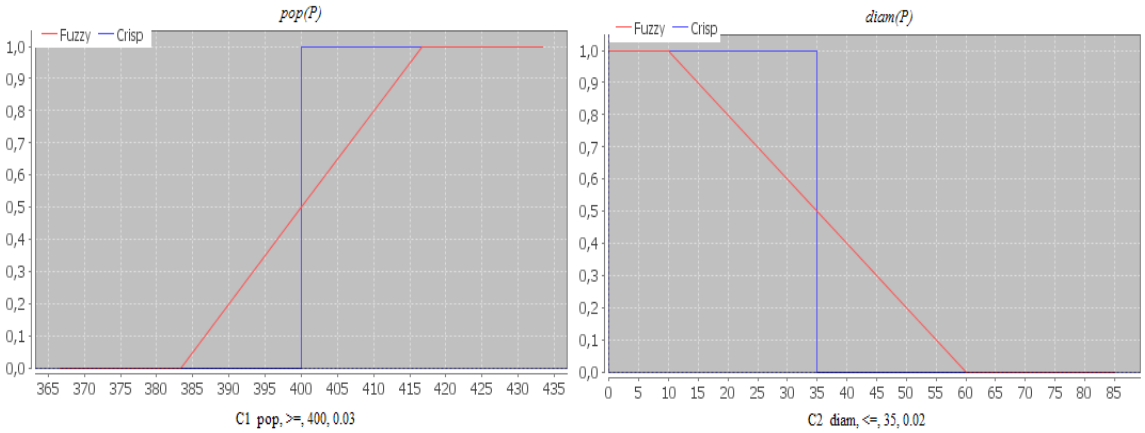
Şekil 3.1. pop (P) <= 60 [-0.04] ve icd (P) >= 5 [0.7] kısıtlarının grafiksel gösterimi

Sonuç olarak, P_x öbeklemesinin P_y öbeklemesinden daha anlamlı olduğu görülür.

Şekil 3.2'de de pop (P) >= 400 [0.03] ve diam (P) <= 35 [-0.02] kısıtları görülmektedir. P_x ve P_y iki öbekleme olsun öyle ki; $pop(P_x)$ = 390, $pop(P_y)$ = 450, $diam(P_x)$ = 20, $icd(P_y)$ = 65'tir. Bu kısıtların olasılıksal yarı halka modelindeki kombinasyonunu göz önünde bulunduralım.

Bu durumda $C_1(P_x)$ = 0.2; $C_2(P_x)$ = 0.8; $C_1(P_x)$ x $C_2(P_x)$ = *çarpım* (0.2, 0.8) = 0.16; $C_1(P_y)$ = 1; $C_2(P_y)$ = 0; $C_1(P_y)$ x $C_2(P_y)$ = çarpım (1, 0) = 0 olur.

Sonuç olarak, P_x öbeklemesinin P_y öbeklemesinden daha anlamlı olduğu görülür.



Şekil 3.2. pop (P) >= 400 [0.03] ve diam (P) <= 35 [-0.02]kısıtlarının grafiksel gösterimi

4. ESNEK KISITLAR TABANLI ÖBEKLEME ALGORİTMALARI

Tüm öbekleme algoritmalarının amacı bir veri kümesindeki elemanları mümkün olduğunca birbirine benzer olanlar aynı öbekte olacak şekilde gruplara ayırmaktır. Kısıt tabanlı öbeklemede de aynı amaç geçerlidir bununla birlikte istenilen kısıtların sağlanması da önemlidir. Birbirine benzer veri öbekleri oluştururken istenilen tüm kısıtların sağlanması genellikle mümkün olmamaktadır. Esnek kısıtlar tabanlı öbekleme kısıtlardan belli ölçüde taviz vererek daha iyi öbeklemeler elde etme amacı taşır.

Geliştirilen esnek kısıtlar tabanlı öbekleme aracının Clustering arayüzünde çeşitli öbekleme algoritmaları çalıştırılabilmektedir. Bu arayüzde bulunan algoritmaların bir kısmı temel öbekleme algoritmaları olmakla birlikte birçoğu da bizim tarafımızdan esnek kısıtlar tabanlı öbekleme problemine çözümler getirmek amacıyla tasarlanmıştır.

Araçta bulunan temel öbekleme algoritmaları temel k-means, merkez seçimi iyileştirilmiş k-means ve bulanık c-means'tir. Merkez seçimi iyileştirilmiş k-means algoritmasının temel k-means algoritmasından tek farkı başlangıçta öbek merkezlerini rasgele değil de belirli bir düzene göre seçmesidir.

Öbek merkezleri seçilirken veri kümesindeki her bir noktanın her boyut değerine tek tek bakılır. Tüm boyutların veri kümesinde tanımlı olan en büyük ve en küçük değerleri bulunur. Daha sonra yine her boyut için en büyük değerden en küçük değer çıkarılır. Elde edilen değer öbek sayısına bölünür. Her boyut için bölüm en küçük değere eklenir böylece ilk öbek merkezi hesaplanmış olur. İkinci merkez de ilk merkez koordinatlarına bölüm değerinin eklenmesiyle elde edilir. Merkez seçim işlemi *k* öbek için gerçekleştirilir.

Temel öbekleme algoritmalarının araçta bulunma sebebi bu algoritmalarla elde edilen öebekleme sonuçlarıyla bu çalışmada tasarlanan algoritma sonuçlarının karşılaştırılmasıdır. Karşılaştırma yapmak amacıyla temel öbekleme

algoritmalarından farklı olarak tasarlanan rasgele öbekleme algoritması da araca koyulmuştur.

Rasgele öbekleme algoritması adından da anlaşıldığı üzere rasgele atama işlemi gerçekleştirir. Her bir nokta rasgele seçilen bir öbeğe atanır ve öbekleme elde edilerek algoritma sonlandırılır.

Örnek seviyesinde kısıtlarla bu çalışmada gerçekleştirilen öbekleme algoritmalarını kıyaslamak için kullanılan MPCK-means algoritmasının [11] da kodu yazılarak araca eklenmiştir. Araçta bulunan diğer algoritmalar ise bu çalışma için tasarlanan algoritmalardır. Bu algoritmalar genetik algoritmalardır.

Genetik algoritmalar problemlere çözümler bulabilmek için doğal genetik metodlarından esinlenen algoritmalardır. Bu algoritmalarda aday çözümleri temsil eden kromozomlar bulunmaktadır. Kromozomlar genlerden oluşur ve her bir genin değerine de alel denir. Bir veya birden fazla kromozom bir araya gelerek populasyonu oluştururlar. Populasyondaki her bir kromozomun çözüm olup olmayacağına karar veren bir uygunluk fonksiyonu (*fitness function*) vardır. Uygunluk fonksiyonuna göre değeri yüksek olan kromozomlara diğer kromozomlarla evrimleşmeleri için fırsat verilir.

Kromozomlar arasında gerçekleşen üç çeşit işlem vardır. Bunlar seleksiyon, çaprazlama ve mutasyondur. Seleksiyon uygunluk fonksiyonu yüksek olan bireyin seçilmesidir. Çaprazlama iki kromozomdaki belirli kısımların yerlerinin birbiriyle değiştirilmesidir. Mutasyon kromozomdaki genlerden bazılarının değiştirilmesidir.

Literatürde genetik algoritmaların öbekleme problemlerine uygulanmasıyla ilgili birçok çalışma bulunmaktadır. K-means algoritması gibi denetlenmeyen bir algoritmanın amaç fonksiyonu verilerin öbek içindeki varyansını minimum yapmak üzere etiketli verilerden de faydalanacak şekilde değiştirilerek genetik algoritmalar kullanılmıştır [17]. Birleştirici hiyerarşik öbekleme yöntemine kıyasla en iyi öbek sayısı ve en iyi öbekleme sonucunu eşzamanlı olarak verecek tekniklerin

geliştirilmesinde genetik algoritmalardan faydalanılmıştır [18]. Öbek içi birleşme (*within-cluster cohesion*) ve öbekler arası ayrım (*between-cluster isolation*) tarifleri üzerine kurulu bir uyum kriterini optimize etmeye dayalı bir genetik algoritma olan COWCLUS algoritması öbek analizine bir yaklaşım olarak yaratılmıştır [19].

Küresel şekilde olmayan veri kümelerinin öbeklenmesi için de genetik öbekleme algoritması bulunmuştur. Bu algoritma verileri özelliklerinin benzerliklerine göre öbeklerken olması gereken öbek sayısını da otomatik olarak bulmaktadır [20]. Büyük veri kümelerinin öbeklenmesinde ise genetik algoritmaların iyi sonuçlar verdiği gözlenmiştir [21].

Bir diğer çalışmada [22] genetik algoritmanın operasyonları seleksiyon, çaprazlama ve mutasyon değiştirilerek hibrid bir genetik bulanık k-modlar öbekleme algoritması elde edilmiştir. Hibrid algoritmalara örnek olarak Hybrid GEN-GRASP [23] algoritmasını da verebiliriz. Bu algoritma genetik algoritmalarla açgözlü bir algoritma olan GRASP algoritmalarının birleşimi sonucu ortaya çıkmıştır. Genetik algoritmayla doğrulama kriterleri birleştirilerek de öbekleme algoritması yaratılmıştır [24]. Doğrulama kriterlerinin kullanıldığı başka bir çalışma da bulunmaktadır [25]. Bu çalışmada Davies-Bouldin indeksi [26], Dunn indeksi [27] vb. doğrulama kriterleri kullanılmıştır. Çaprazlama operasyonu değiştirilerek ideal öbek sayısı hesaplanmaya çalışılmıştır.

Günlük hayatta karşılaşılan problem çözümleri için de genetik öbekleme algoritmaları tasarlanmıştır. Çok depolu araç rotalama problemi bu problemlerden birisidir [28]. Bir diğer uygulama üretim simülasyonu içindir. Üretim hızının ayarlanmasında öbek analizleri ile genetik algoritmalar kullanılmıştır [29]. Kısmi en küçük kareler yol modellemesinde (*PLS path modeling*) yeni bir yaklaşım olarak genetik algoritmalar kullanılmış bu algoritma da PLS-PMC olarak adlandırılmıştır [30].

Bu uygulamalardan başka farklı 20 tavuk ırkından alınan 600 bireyin multilokus genotipleri kullanılarak genetik öbeklemenin değerlendirilmesi yapılmıştır [31]. Elde

edilen sonuçlar populasyon yapısını çok iyi belirleme potansiyeline sahip bir teknik olduğunu göstermiştir.

4.1. Minimum Uzaklıklar Toplamı Genetik Algoritması (MUTGA)

Bu algoritmanın amaç fonksiyonu k-means algoritmasının amaç fonksiyonunu temel almaktadır. Aynı k-means'teki gibi noktaların merkezlere olan uzaklıkları toplamının minimum olması hedeflenir. Toplamın minimum olması hedeflendiği için de bu problem bir optimizasyon problemi gibi düşünülür. Optimizasyon problemlerinin çözümünde genetik algoritmalardan faydalanılır [32, 33]. Bu algoritma da bir genetik algoritmadır.

Söz konusu algoritmada her bir kromozom bir veri kümesini temsil etmektedir. Kromozomda bulunan her bir gen bir veriye karşılık gelmektedir. Genlerin alelleri ise öbek numarasıdır.

Algoritmanın uygulaması gerçekleştirilirken bir java genetik algoritma paketi olan JGAP kütüphanesi kullanılmıştır [34].

4.2. MUTGA ve K-Means

Önce k-means algoritması *n* iterasyon çalıştırılır. Bu *n* iterasyonun son *m* tanesindeki öbeklenmiş veri kümeleri alınır. *m* veri kümesinin her biri bir kromozom olarak *m* bireyden oluşan bir populasyon oluşturulur. Son olarak bu populasyon üzerinde MUTGA çalıştırılır.

4.3. Esnek Kısıtlar Genetik Algoritması (EKGA)

Bu algoritmanın MUTGA algoritmasından farkı uygunluk fonksiyonunun farklı olmasıdır. Ek-1'de örnek olarak verilen bir kısıt dosyasını girdi olarak alır ve bu kısıtları ağırlıklı yarı halka modeliyle tanımlayarak sağlamaya çalışır. *c* adet kısıt varsa uygunluk fonksiyonunun alacağı en yüksek değer de ağırlıklı halka modeline

göre *c* olur. Tüm kısıtların eşit derecede sağlanması istendiğinden ağırlıklı halka modeli kullanılmaktadır. Bulanık ve olasılıksal yarı halka modelleriyle de algoritmalar oluşturulabilir.

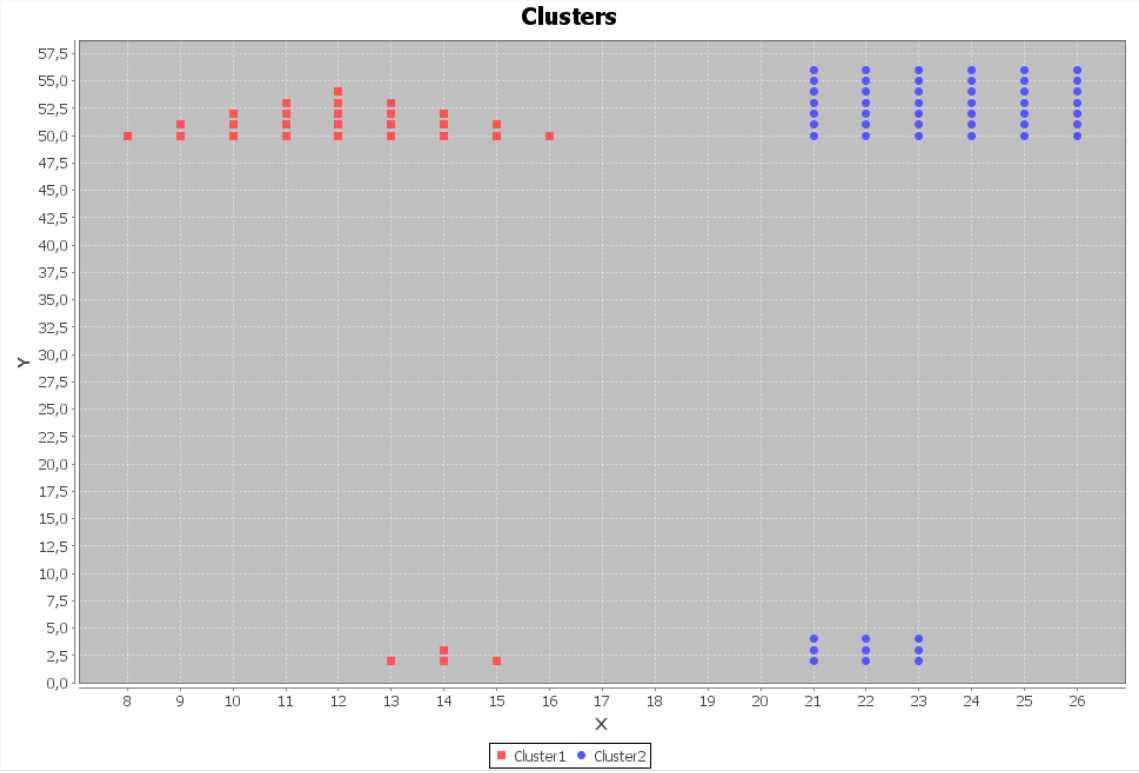
4.4. EKGA ve K-means

Önce k-means algoritması *n* iterasyon çalıştırılır. Bu *n* iterasyonun son *m* tanesindeki öbeklenmiş veri kümeleri alınır. Oluşacak populasyonun *m* bireyi bu kromozomlardan oluşur. *x* adet kromozom da rasgele yaratılarak *m* bireyin olduğu populasyona eklenir. Son olarak da *m+x* bireylik populasyon üzerinde EKGA çalıştırılarak kısıtlar sağlanmaya çalışılır.

4.5. Dikey Öbekleme Algoritması

Dikey öbekleme algoritması Şekil 4.1'deki örneğe bakılarak daha iyi açıklanabilir. Örnekteki noktalara yakından bakıldığında dört grup halinde toplandıkları fark edilmektedir. Bu gruplardan ikisi üçgen, ikisi de dikdörtgen şeklindedir. Bu noktaların tümü veri kümesini oluşturmaktadır. Veri kümesi iki ayrı öbek oluşturacak şekilde gruplanmak istensin. Gruplama işini yaparken de k-means algoritması kullanılsın. K-means algoritması noktaları benzer olacak şekilde gruplarken aralarındaki uzaklığa bakacaktır. Sonuç olarak iki öbekten oluşan bir öbekleme elde edildiğinde büyük üçgen ve büyük dikdörtgen şeklindeki nokta grupları birinci öbeği, küçük üçgen ve küçük dikdörtgen şeklindeki nokta grupları da ikinci öbeği meydana getirecektir. Şekil 4.1'de olduğu gibi aynı şekle sahip olan gruplarının aynı öbekte olması istendiğinde k-means vb. algoritmalar bunu gerçekleştiremez.

Dikey öbekleme algoritması ise elde edilmek istenen öbeklemenin şekline yakın bir veri kümesini model olarak alıp üzerinde genetik algoritma çalıştırarak istenen şekildeki öbeklemeyi elde etmeye çalışır. Burada gerçekleştirilen model seviyesinde kısıtlar tabanlı (*model-level constraints based*) bir öbeklemedir.



Şekil 4.1. Dikey öbekler

Model seviyesinde kısıtlar tabanlı öbekleme kullanılarak yapılan çalışmada ise iki tip model kullanılmıştır [35]. Bu modellerden ilki elde edilecek öbeklemenin benzemesi istenmeyen öbekleme örnekleri, ikincisi ise öbeklemenin ilişkili olması istenmeyen negatif özelliklerdir.

4.6. EKGA – Örnek Seviyesinde (EKGA – ÖS)

Öbek seviyesindeki kısıtlarla gerçekleştirilen EKGA'nın örnek seviyesindeki kısıtlar için gerçekleştirilen versiyonudur.

4.7. EKGA ve EKGA - ÖS

EKGA'nın hem öbek hem de örnek seviyesinde kısıtlarla gerçekleştirilmesidir.

4.8. EKGA Çöp (EKGA - Ç)

Bu algoritmanın EKGA'dan farkı, elde edilmek istenenen öbek sayısından bir fazla sayıda öbekle bir öbekleme oluşturulmasıdır. Öbekleme algoritmasına öbek sayısı *k* olarak verilmişse *k+1* öbek oluşturulur ve bu *k+1* numaralı öbek çöp öbeği olarak atanır.

4.9. EKGA ve EKGA – ÖS Çöp

EKGA - Ç'nin hem öbek hem de örnek seviyesinde kısıtlarla gerçekleştirilmesidir.

5. ESNEK KISITLAR TABANLI ÖBEKLEME ARACI

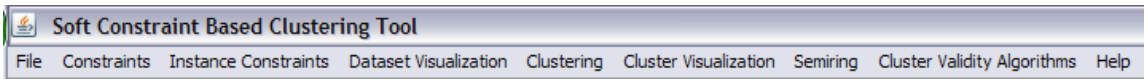
Kısıtların yaratılması, yaratılan kısıtların grafiksel olarak gösterimi, üzerinde öbekleme gerçekleştirilen veri kümelerinin gösterimi, öbekleme algoritmalarının çalıştırılması, oluşturulan öbeklemelerdeki öbeklerin görüntülenmesi, yarı halkalar üzerinde öbekleme sonuçlarının hesaplanması ve elde edilen öbeklemelerin orijinalleriyle karşılaştırılmasında kullanılan öbek doğrulama algoritmalarının çalıştırılması işlemlerini gerçekleştiren bir araç geliştirilmiştir.

5.1. Aracın Genel Özellikleri

Esnek kısıtlar tabanlı öbekleme aracı, Java programlama dili kullanılarak NetBeans IDE ortamının 6.8 sürümüyle gerçekleştirilmiştir. Veri kümeleri ve elde edilen öbekleme sonuçları Notepad dosyalarında saklanmaktadır. Yukarıda bahsi geçen işlemleri gerçekleştirmek için yedi adet görsel arayüz tasarlanmıştır.

5.2. Arayüzler

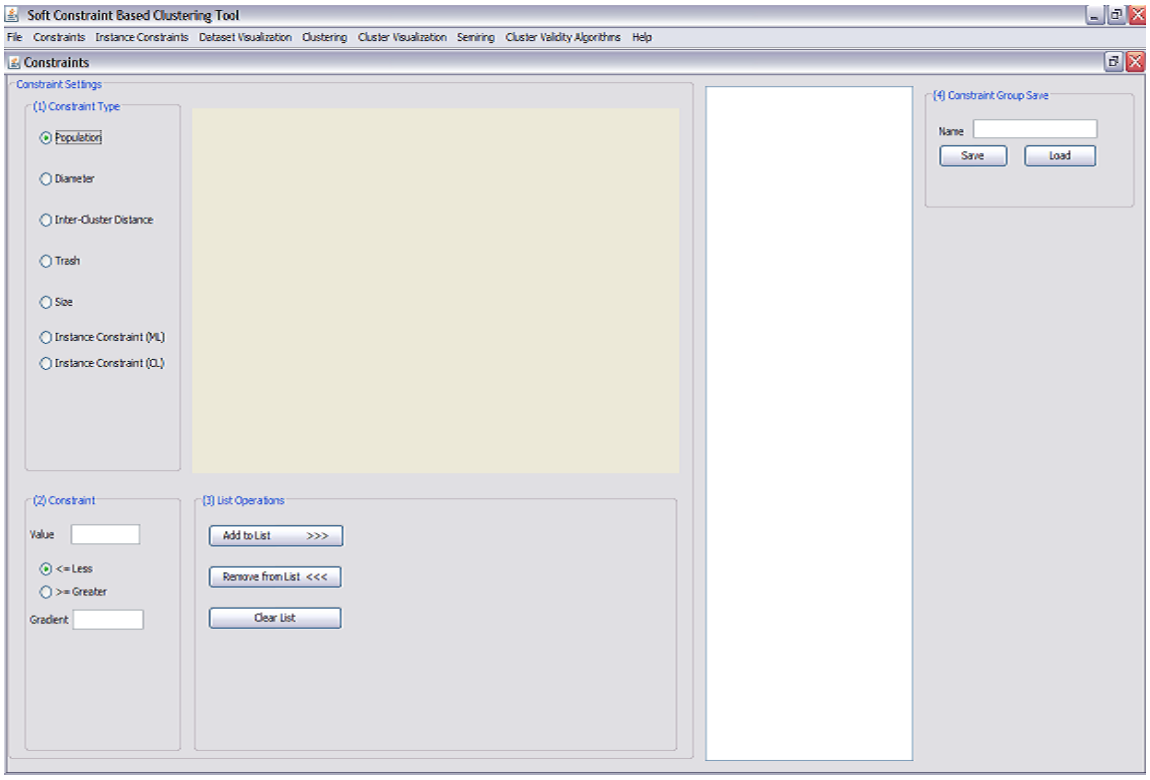
Arayüzler, Şekil 5.1'de görüldüğü üzere, her bir araç menüsü başlığı altında bir arayüz açılacak şekildedir.



Şekil 5.1. Esnek Kısıtlar Tabanlı Öbekleme Aracı ana menü

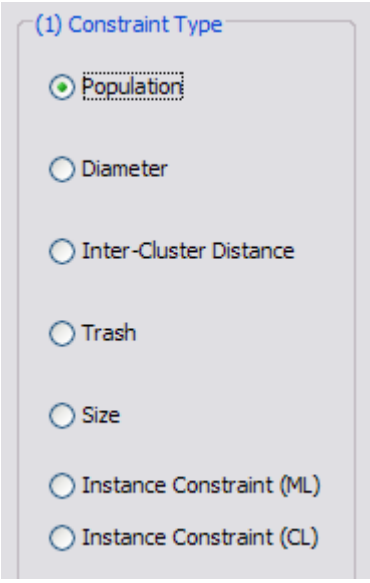
5.2.1. Constraints Arayüzü

Kısıt tabanlı öbekleme yapmanın ilk adımı veri kümesi üzerinde kullanılacak olan kısıtların belirlenmesidir. Constraints arayüzünde kısıtların özellikleri belirlenir. Arayüzün varsayılan hali Şekil 5.2'deki gibidir.



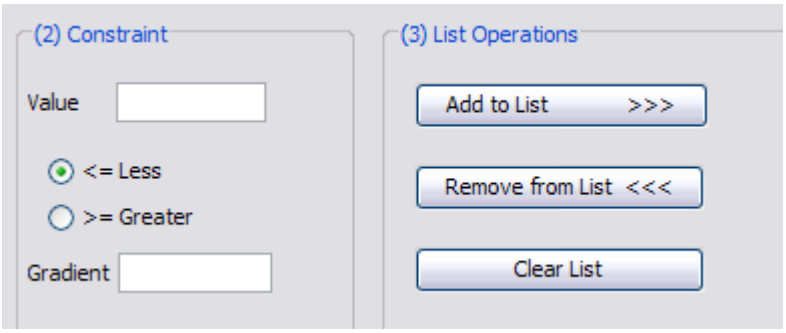
Şekil 5.2. Constraints arayüzü

Arayüz dört alt bölümden oluşmaktadır. İlk bölüm kısıt tipinin belirlendiği Constraint Type'dır. Bu bölümde Şekil 5.3'te de görüldüğü üzere Population, Diameter, Inter-Cluster Distance, Trash, Size, Instance Constraint (ML) ve Instance Constraint (CL) olmak üzere yedi tip kısıttan bir tanesi seçilmektedir. İlk beş kısıt öbek seviyesinde kısıtlar, son iki kısıt da örnek seviyesinde kısıtlardır.



Şekil 5.3. Kısıt tipi seçimi

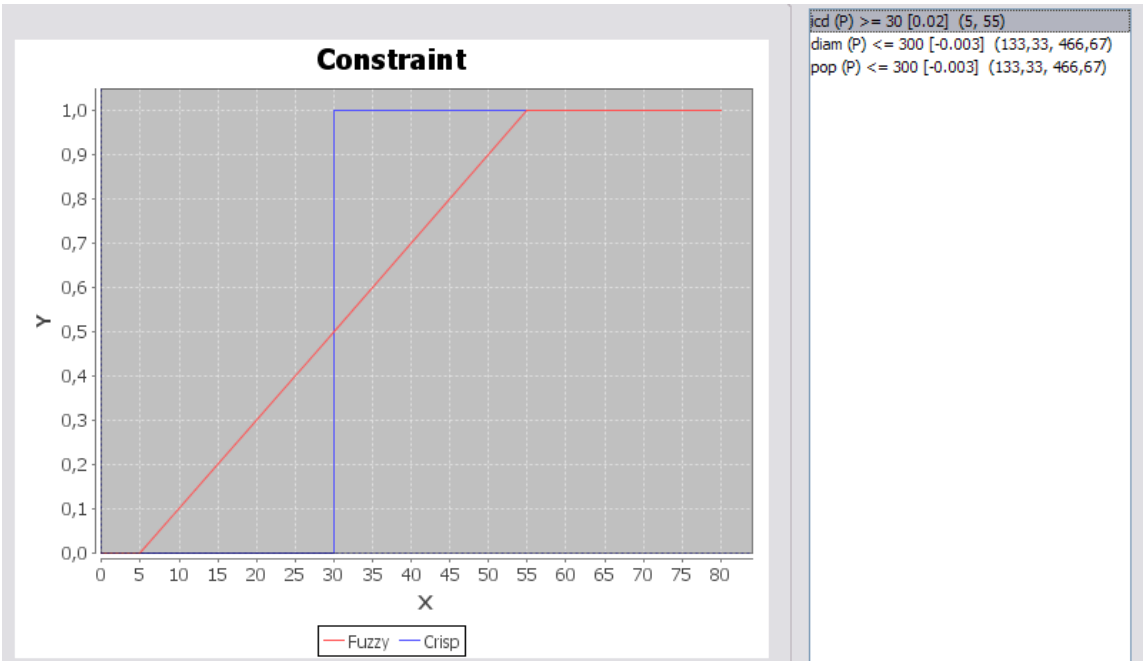
İkinci bölüm kısıtın orta noktası, işareti ve eğiminin girildiği Constraint'tir. Buraya yazılan değerler üçüncü bölüm olan List Operations'daki Add to List düğmesi kullanılarak kısıt listesine eklenir. İkinci ve üçüncü bölümler Şekil 5.4'te görülmektedir.



Şekil 5.4. Kısıt değerleri ve liste işlemleri

Liste üzerinde gerçekleştirilen dört işlem vardır. Bunlardan üçü List Operations bölümündeki düğmeler aracılığıyla sonuncusu ise listedeki kısıtın seçilmesiyle gerçekleştirilir. Remove from List düğmesi seçili kısıtı (elemanları) listeden kaldırır. Clear List listeyi boşaltır. Şekil 5.5'teki gibi bir kısıt seçildiğinde, kısıtın yarı halka

üzerindeki grafiksel gösterimi çizdirilir. Grafiksel gösterim için jfreechart-1.0.13 [36] ile jcommon-1.0.16 hazır kütüphaneleri NetBeans kütüphanelerine eklenmiş ve bu kütüphanelerden faydalanılmıştır.



Şekil 5.5. Yarı halka üzerinde kısıt gösterimi

Şekil 5.5'teki örnekte görüldüğü üzere kısıtların hem katı hem de esnek halleri çizdirilmektedir. Mavi çizgi katı, kırmızı çizgi esnek kısıtı göstermektedir. Bu örnekte listedeki ilk kısıt seçilmiş ve çizdirilmiştir. Buna göre öbekler arası uzaklık (icd (P)) değerinin 30'dan büyük olması istenmektedir. Katı kısıtlarda esneklik olmadığından (eğim sıfır olduğundan) iki öbek arasındaki uzaklık 30'dan küçükse kısıtın sağlanma değeri 0, 30 ve 30'dan büyükse kısıtın sağlanma değeri 1 olur.

Esnek kısıtın yorumu ise farklıdır ve çizdirilirken hesaplanması gereken değerleri buunmaktadır. Örnek için esneklik parametresi 0.02 olarak girilmiştir. Kısıtın merkezi olan 30, 0.5 değerine karşılık gelir. Bu bilgilerden yola çıkarak eğimi ve bir noktası bilinen bir doğru denklemi elde edilir. Bir doğru denklemi şu şekilde ifade edilir:

$$y = mx + n \tag{5.1}$$

Örnekteki değerler denklemde yerine koyulur:

$$0.5 = 0.02 * 30 + n \tag{5.2}$$

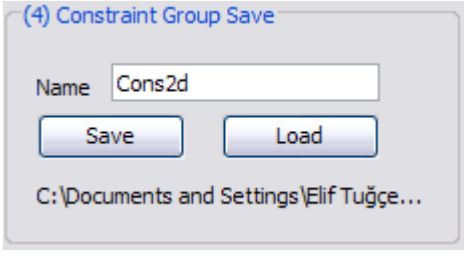
Buradan da *n* değeri elde edilir. Örnek için *n=-0.1*'dir. *n* değerini hesapladıktan sonra kısıtın alabileceği minimum (0) ve maksimum (1) değerlere karşılık gelen noktalar hesaplanır:

$$0 = 0.02 * x_0 - 0.1 \tag{5.3}$$

$$1 = 0.02 * x_1 - 0.1 \tag{5.4}$$

6.3 denkleminden $x_0=5$, $x_1=55$ olarak hesaplanır. Bu sonuçlara göre ve Şekil 5.5'ten de görüleceği üzere iki öbek arasındaki uzaklık 5 ve 5'ten az olursa kısıtın sağlanma değeri 0, 55 ve 55'ten fazla olursa kısıtın sağlanma değeri 1 olur. 5 ve 55 aralığındaki değerler içinse doğru denklemi kullanılarak kısıtın sağlanma değeri hesaplanır. Bazı değerlerin karşılıkları çizimden rahatlıkla görülmektedir. Örneğin iki öbeğin arasındaki uzaklık 20 iken kısıtın sağlanma değeri 0.3, uzaklık 45 iken ise 0.8'tir.

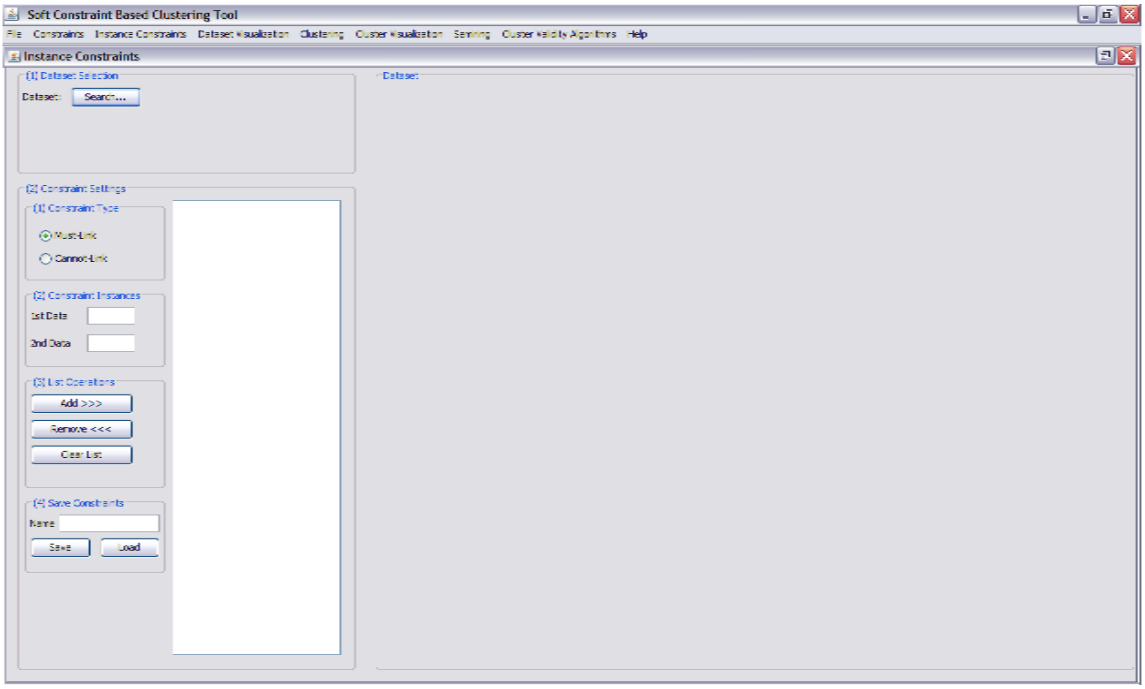
Şekil 5.6'da görülen Constraints arayüzünün son ve dördüncü bölümünde ise oluşturulan kısıtların kaydedilmesi veya kaydedilen kısıtların kayıtlı olduğu dizinden açılarak listeye yüklenmesi işlemleri gerçekleştirilir. Kısıtlar.txt uzantılı bir not defteri dosyasına kaydedilir. Bir kısıt dosyası örneği Ek 1'de verilmiştir.



Şekil 5.6. Kısıtların kayıt işlemleri

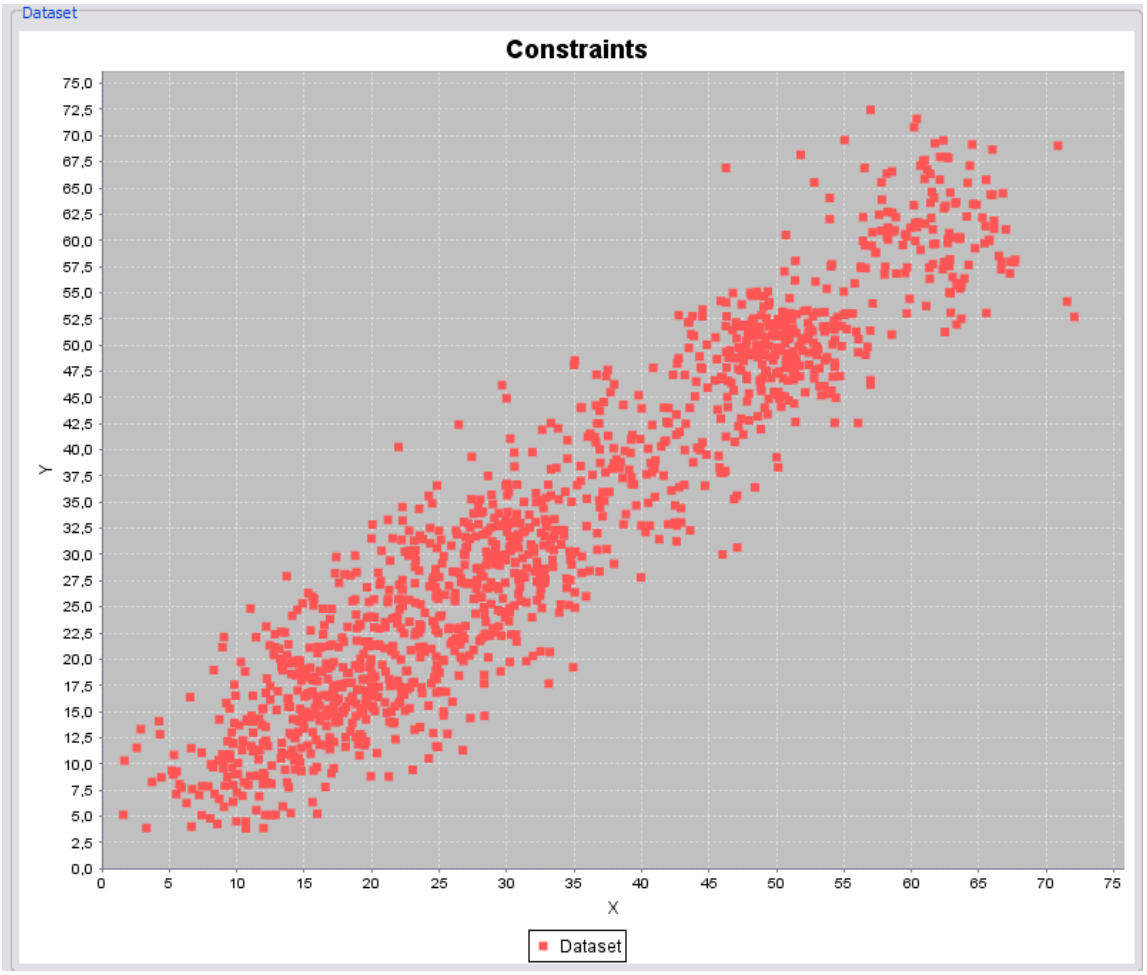
5.2.2. Instance Constraints Arayüzü

Instance constraints arayüzü örnek-seviyesindeki kısıtların oluşturulması için kullanılır. Arayüz Şekil 5.7'de görülmektedir.



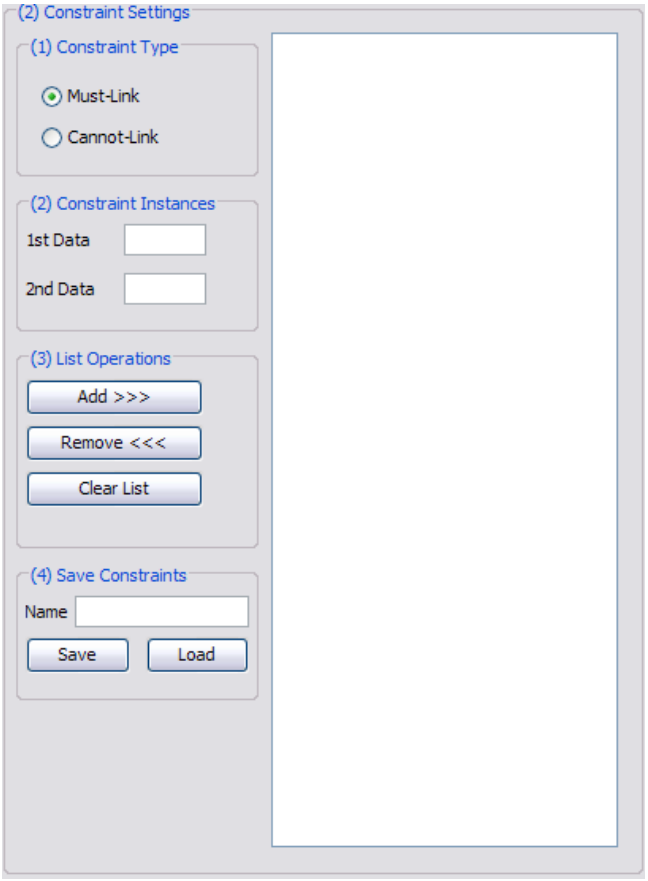
Şekil 5.7. Örnek seviyesinde kısıtlar arayüzü

Bu arayüzün ilk bölümü olan Dataset Selection, üzerinde kısıtların tanımlanacağı veri kümesini seçmeye yarar. Search düğmesine basılarak istenilen veri dosyası seçilir. Dosya seçildiğinde veri kümesi görsel olarak ekranda gösterilir. Şekil 5.8'de gösterilen veri kümesi 2-boyutlu 1300 noktadan oluşan bir veri kümesidir. Çok boyutlu verilerin gösterimi Dataset Visualization arayüzünde anlatılacaktır.



Şekil 5.8. Veri kümesi gösterimi

Ekranda gösterilen veri kümesinden arasında kısıt bulunması istenilen noktaların seçilmesi işlemi ve konulacak kısıtın tipi Şekil 5.9'da görülen ikinci bölüm Constraint Settings alanında yapılır. ML veya CL kısıt tiplerinden biri seçildikten sonra arasına konulacak iki noktanın belirleyici numaraları (*id*) girilir. Noktaların veri kümesindeki numarası, *x* ve *y* koordinatları noktanın üzerine gelindiğinde ekranda görülebilmektedir. Şekil 5.10'da, Şekil 5.8'de görülen veri kümesine JFreeChart kütüphanesinin bir özelliği olan yakınlaştırma işlemi uygulanmış, ilgilenilen noktaların seçimi kolaylaştırılmıştır.

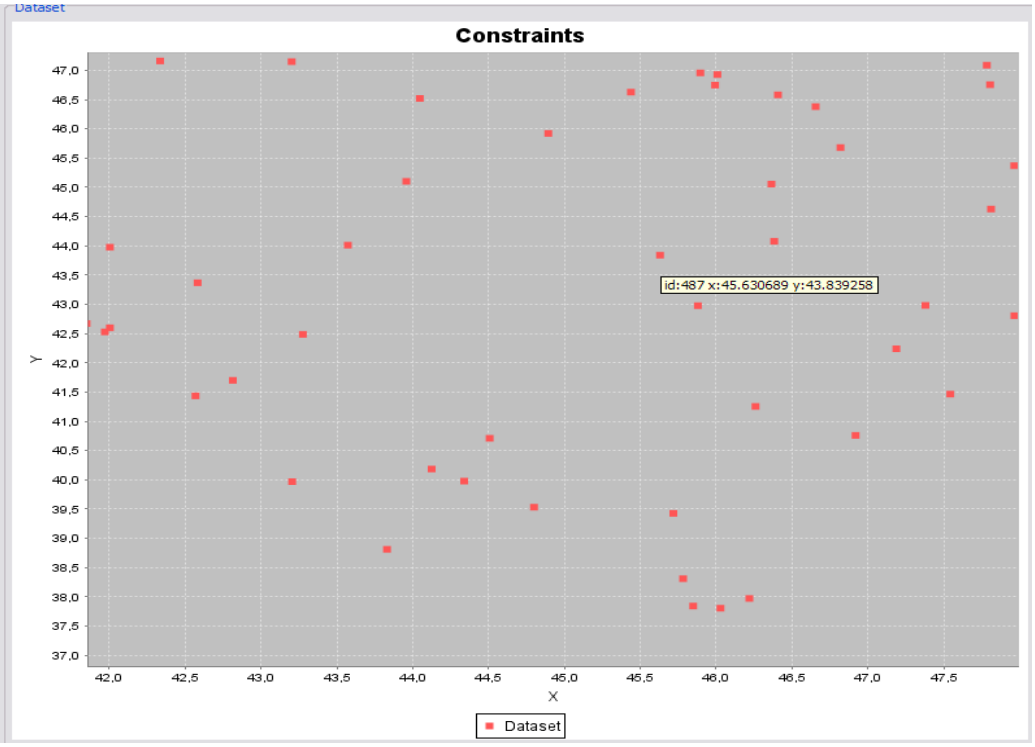


Şekil 5.9. Kısıt ayarları

Oluşturulan kısıtların listelenmesi, oluşturulan listelerin kaydedilmesi işlemleri Constraints ekranındaki gibi gerçekleştirilir. Constraints arayüzündeki işlemlerden farklı olarak bu arayüzde oluşturulan kısıtlar kaydedilirken şu format kullanılır:

id_1 id_2 1
id_1 id_2 -1

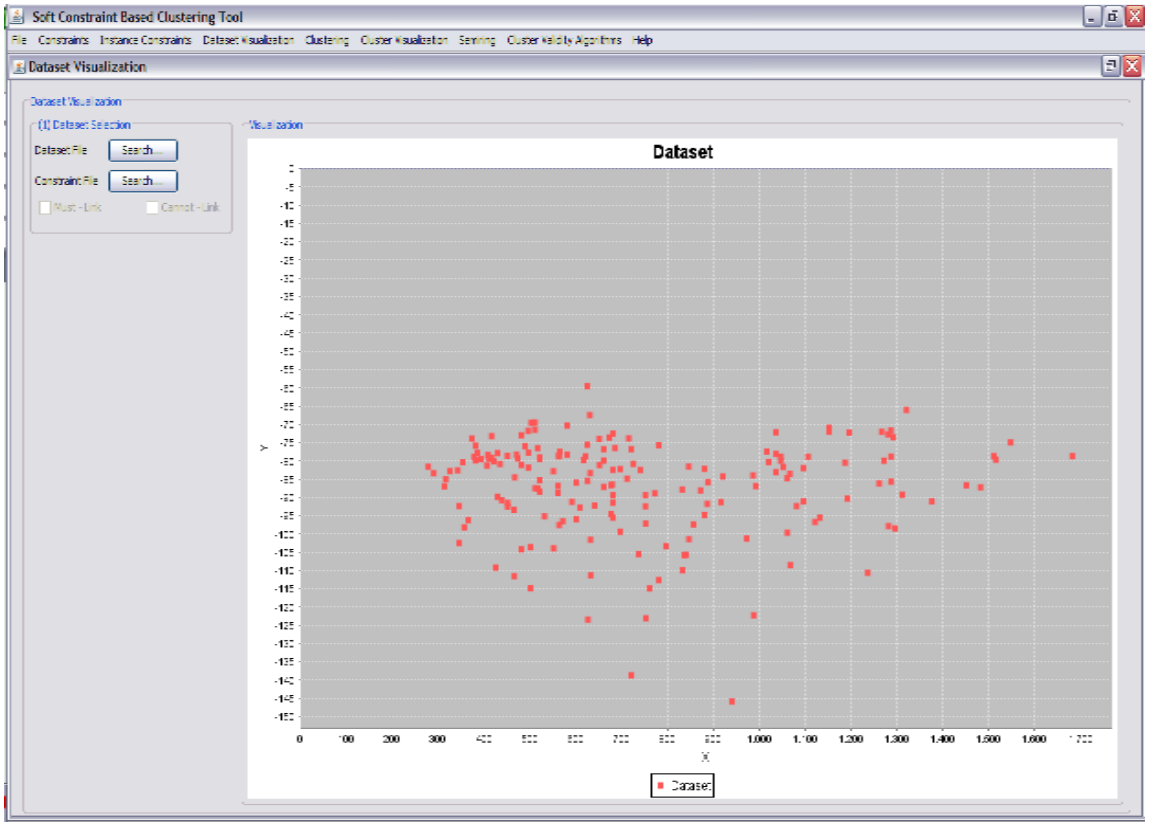
Burada id_1 ilk noktanın veri kümesindeki belirleyici numarası, id_2 ikinci noktanın veri kümesindeki belirleyici numarasıdır. Üçüncü sütundaki değer kısıtın tipini gösterir. ML kısıtlarını göstermek için 1, CL kısıtlarını göstermek için -1 kullanılır.



Şekil 5.10. Noktaların yakından incelenmesi

5.2.3. Dataset Visualization Arayüzü

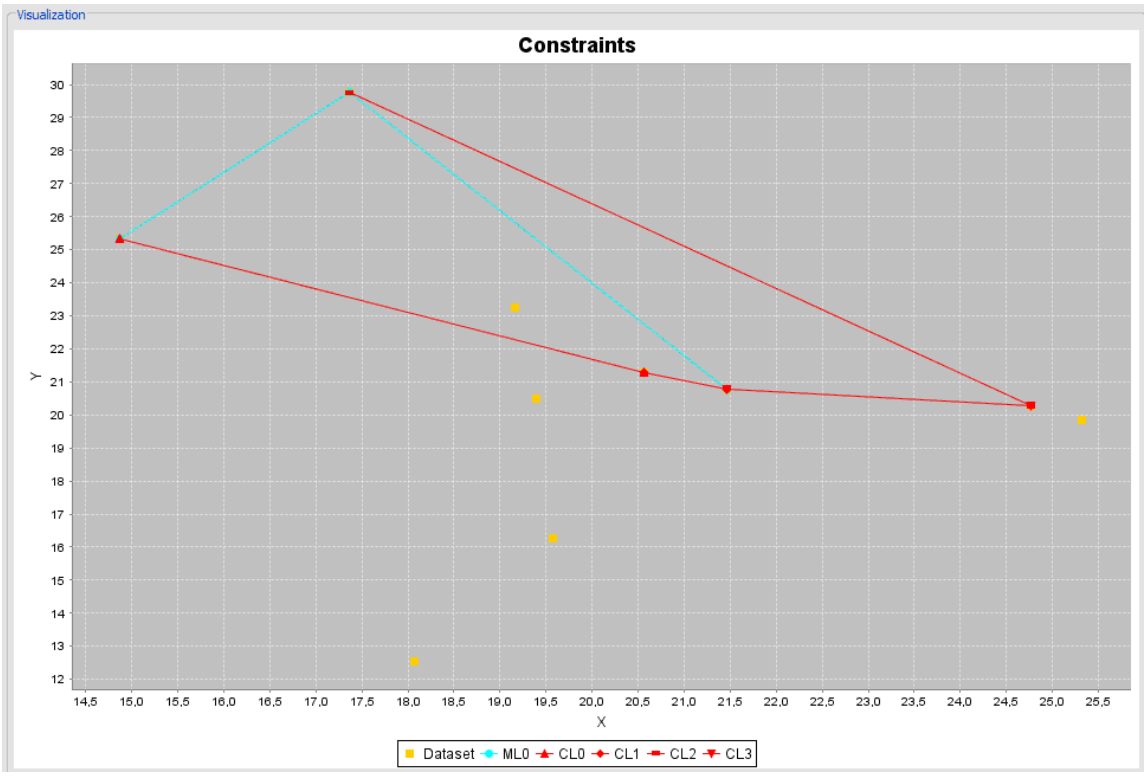
Şekil 5.11'de görülen Dataset Visualization arayüzü öbeklenecek veri kümelerini ve eğer varsa bu veri kümeleri üzerinde tanımlanmış örnek seviyesinde kısıtları görüntülemeye yarar.



Şekil 5.11. Dataset Visualization arayüzü

Şekil 5.11'de görüntülenen UCI Machine Learning Repository'den alınmış Wine [37] gerçek veri kümesidir. Wine, 178 veriden oluşan 13 boyutlu (13 özellikli) bir veri kümesidir. Çok boyutlu veri kümelerini 2 boyutlu şekilde gösterebilmek için çok boyutlu veri kümelerine temel bileşenler analizi (*principal component analysis*) [38] uygulanmaktadır. Buradaki temel bileşenler analizini gerçekleştiren PCA sınıfı JavaStatSoft [39] istatistiksel yazılım programından alınmıştır.

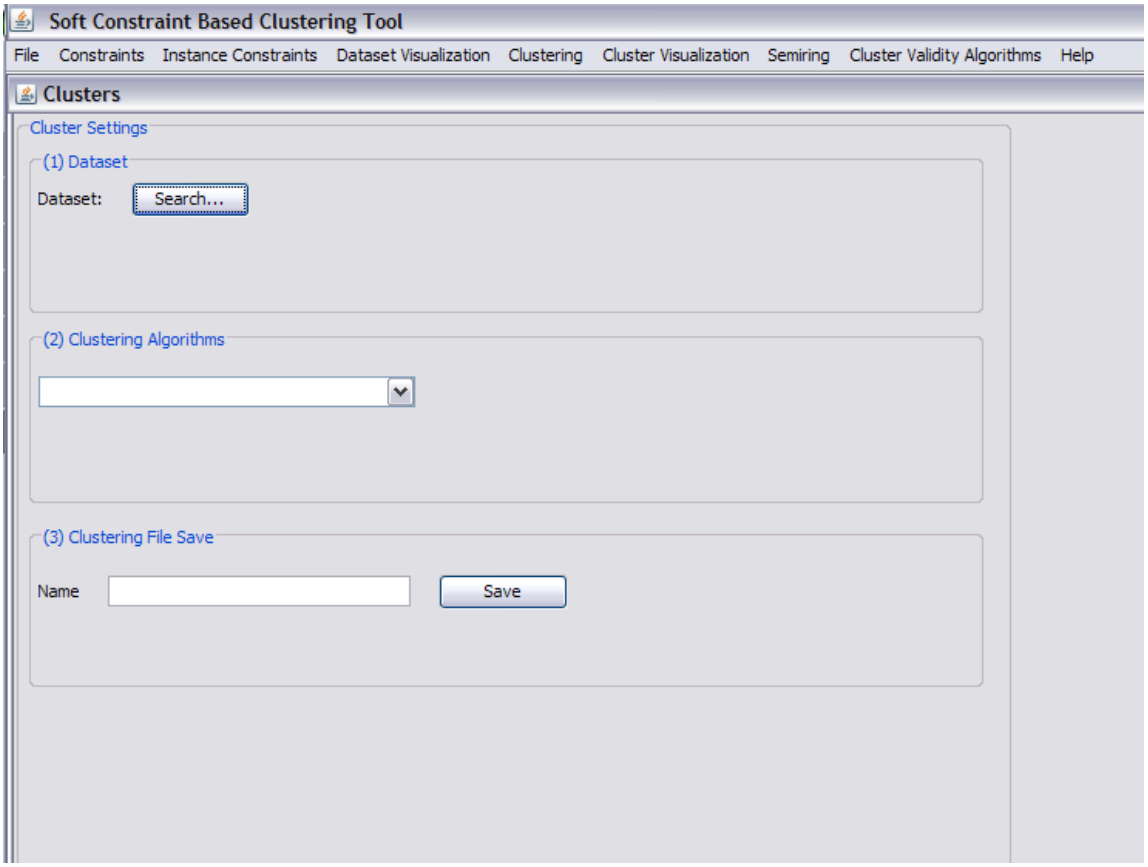
Veri kümesi üzerinde tanımlanmış örnek-seviyesinde kısıtlar varsa Şekil 5.12'deki gibi yine bu arayüz üzerinde görütülenebilir. Bu grafiklerde mavi çizgiler ML kısıtıyla, kırmızı çizgiler de CL kısıtıyla birbirine bağlı olan verileri göstermektedir.



Şekil 5.12. Veri kümesi üzerine tanımlı örnek seviyesinde kısıtlar

5.2.4. Clustering Arayüzü

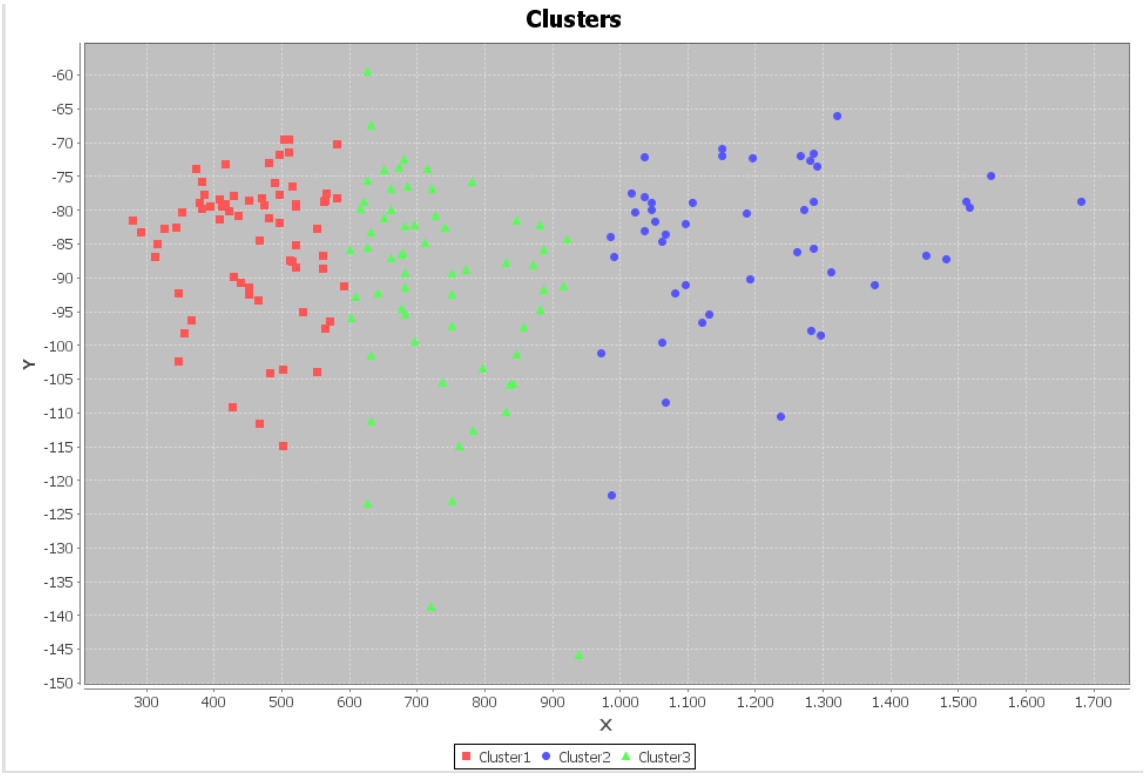
Clustering arayüzü seçilen veri kümesi üzerinde çeşitli öbekleme algoritmalarının çalıştırıldığı ve sonuçta oluşan öbekleme dosyalarının kaydedildiği arayüzdür. Bu arayüzde çalıştırılan algoritmalardan yedinci bölümde bahsedilecektir. Genel hatlarıyla Clustering arayüzü Şekil 5.13'teki gibidir.



Şekil 5.13. Clustering arayüzü

5.2.5. Cluster Visualization Arayüzü

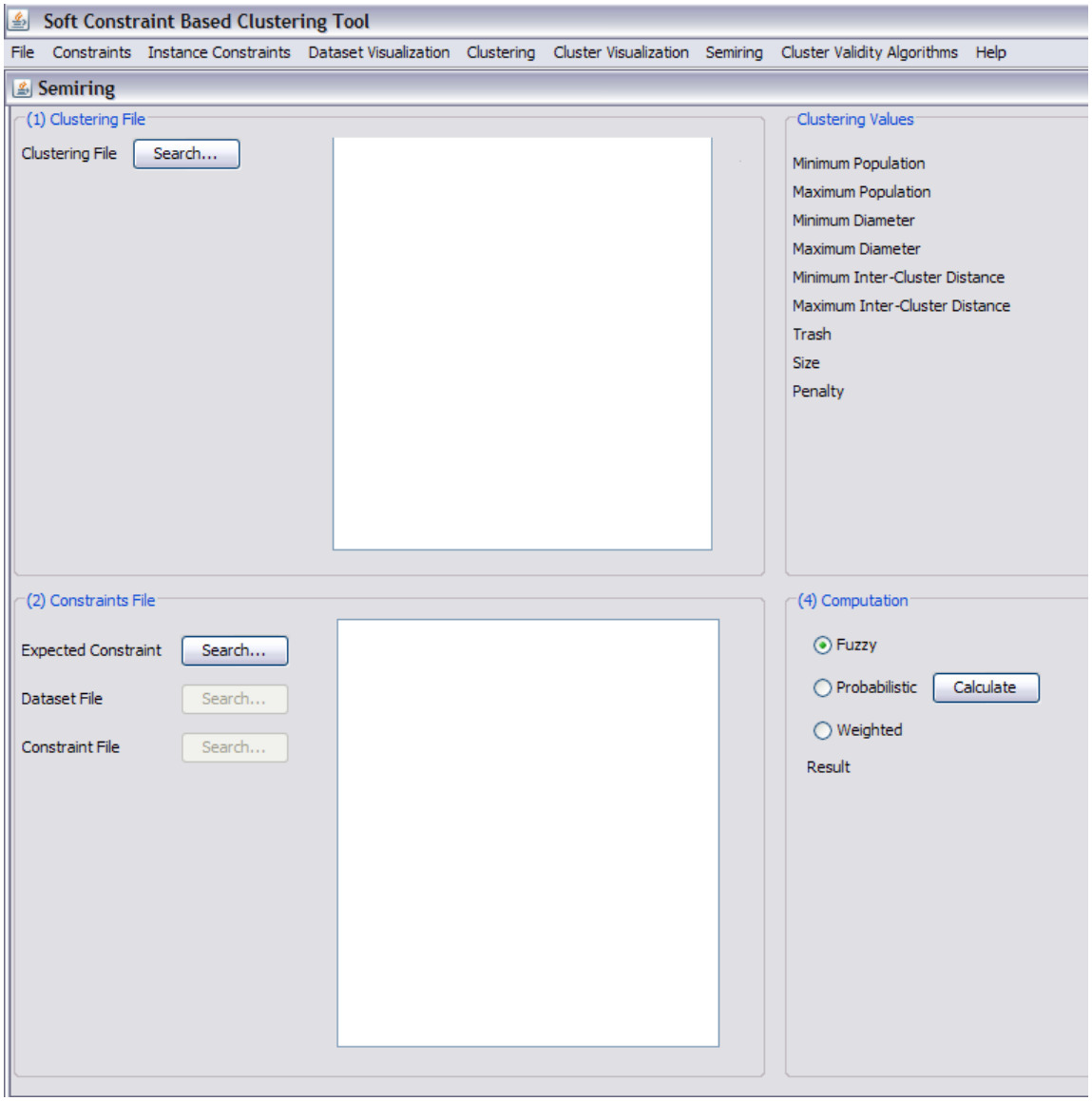
Cluster Visualization işlevsel olarak Dataset Visualization gibi çalışmaktadır. Tek farkı öbekleri ayrı ayrı göstermesidir. Şekil 5.14'te görüldüğü gibi her bir öbeğe ait noktalar farklı renklerle gösterilir. Arayüz tüm öbekleri aynı anda gösterebileceği gibi sadece seçilen öbekleri de gösterebilir. Aynı zamanda veri kümesi üzerinde tanımlı örnek seviyesinde kısıtlar varsa bunların gösterilmesini de sağlar.



Şekil 5.14. Cluster Visualization arayüzünde öbeklerin görüntülenmesi

5.2.6. Semi-ring Arayüzü

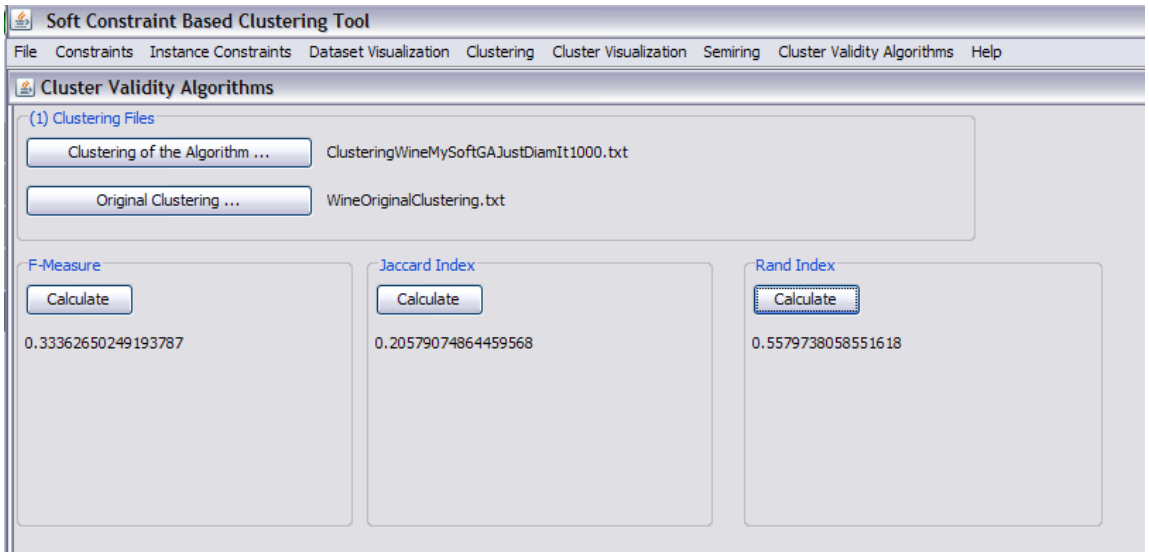
Semi-ring arayüzü bir öbeklemedeki öbek sonuçlarının gösterilmesini sağlar. Bu öbeklemedeki öbeklerin en az ve en fazla populasyon, çap, öbekler arası uzaklık değerleri ile öbeklemede bulunan çöp sayısı, öbek sayısı ve eğer varsa örnek-seviyesindeki kısıtların ihlal sayısı bu arayüzde hesaplanır. Öbeklemedeki bu değerler ile sağlanması istenen kısıtların karşılaştırılması ve bulanık, olasılıksal veya ağırlıklı yarı halka modellerinden seçilen herhangi bir modele göre öbeklemedeki kısıtların sağlanma değeri de bu arayüzde hesaplanır. Semi-ring arayüzünün varsayılan görünümü Şekil 5.15'teki gibidir.



Şekil 5.15. Semi-ring arayüzü.

5.2.7. Cluster Validity Algorithms Arayüzü

Cluster Validity Algorithms arayüzü bir son bölümde detaylı olarak bahsedilecek öbek doğrulama algoritmalarından üç tanesini kullanarak orijinal öbekleme kümesiyle öbekleme algoritması kullanılarak elde edilen öbekleme kümelerini karşılaştırır. İstenilen öbek doğrulama algoritmasına göre de öbekleme algoritmasının orijinale ne kadar yaklaştığı sonucunu verir. Şekil 5.16'da bu arayüzün kullanımına bir örnek verilmiştir.



Şekil 5.16. Cluster Validity Algorithms arayüzü

6. PERFORMANS DEĞERLENDİRMESİ

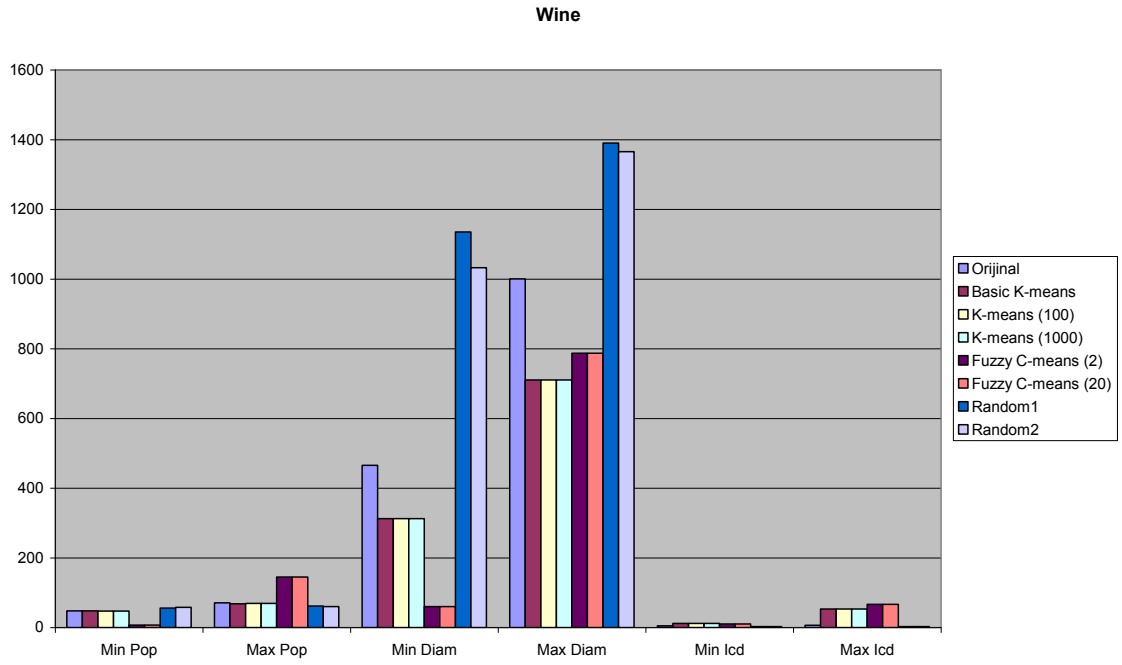
Bu çalışmada esnek kısıtlara uygun öbeklemeler elde edilmesi amacıyla genetik algoritmalardan faydalanılarak yeni öbekleme algoritmaları geliştirilmiş ve Java programlama dili kullanılarak bu öbekleme algoritmalarının gerçekleştirildiği bir uygulama tasarlanmıştır. Uygulama kullanılarak algoritmalar veri kümeleri üzerinde çeşitli kısıtlarla çalıştırılmış ve başarı oranları incelenmiştir. Bu bölümde veri kümeleri üzerinde gerçekleştirilen denemelerden ve denemelerin sonuçlarından bahsedilecektir.

Sonuçlar üç farklı veri kümesi üzerinde yapılan denemeler sonucunda elde edilen verilerden çıkarılmıştır. Bu veri kümelerinden ikisi UCI Machine Learning Repository'den alınmış olan Wine ve Soybean (small) [40] gerçek veri kümeleridir. Üçüncüsü ise MATLAB kullanılarak oluşturulan yapay bir veri kümesidir. Yapay veri kümesi iki boyutlu 1300 örnekten oluşmaktadır. Bu veri kümesini oluşturan kod Ek 2'de bulunmaktadır.

Geliştirilen arayüz kullanılarak üç veri kümesi üzerinde de arayüzde bulunan algoritmalar çalıştırılmış elde edilen populasyon, çap, öbekler arası uzaklık değerleri kaydedilmiştir. Ayrıca eğer varsa çöp, ML ve/veya CL penaltıları, kullanılan kısıtlar, kromozom sayıları, genetik operatörler ve iterasyon sayıları da tutulmuştur. Bu değerlerle birlikte üç adet öbek doğrulama algoritması her bir algoritmadan elde edilen her bir öbekleme üzerinde çalıştırılmış bunların sonuçları da kaydedilmiştir.

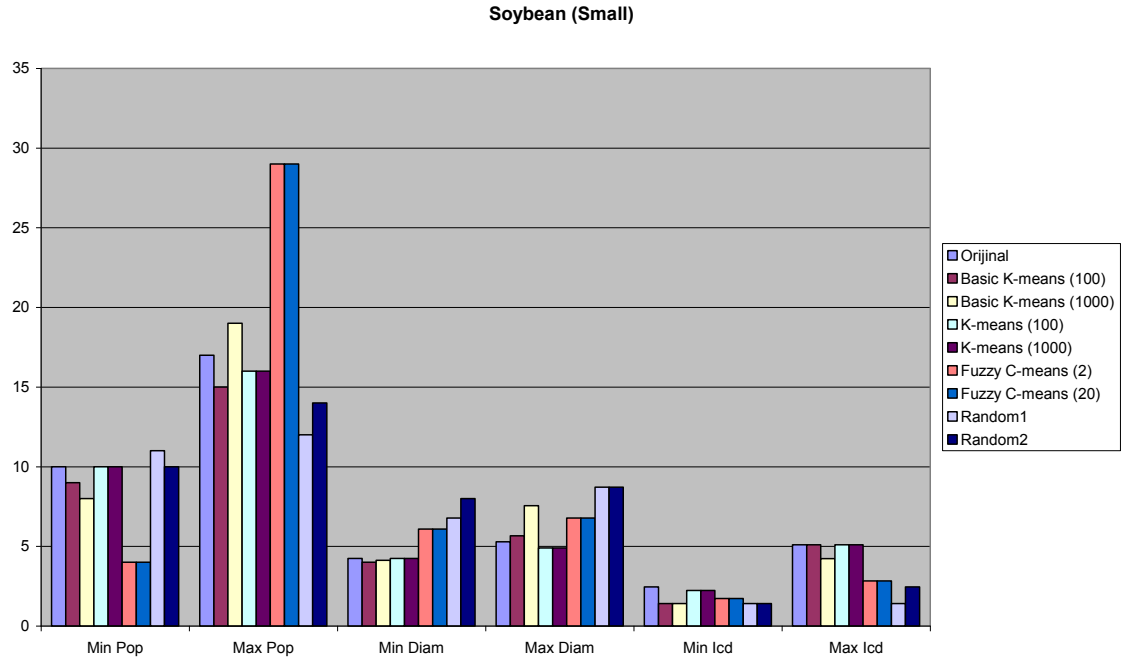
Arayüzde bulunan temel öbekleme algoritmalarıyla elde edilen öbeklemelerdeki değerler ve orijinal veri kümelerindeki değerler sırasıyla Wine, Small Soybean ve Yapay 2d kümeleri olmak üzere Şekil 6.1, Şekil 6.2 ve
Şekil 6.3'de görülmektedir. Her veri kümesi üzerinde temel k-means algoritması 100 iterasyon olarak, merkez seçimi geliştirilmiş k-means algoritması 100 ve 1000 iterasyon olarak, bulanık c-means algoritması bulanıklık değerleri 2 ve 20 olarak, rasgele öbekleme algoritması da iki kere çalıştırılarak elde edilen öbeklemelerdeki minimum populasyon, maksimum populasyon, minimum çap, maksimum çap,

minimum öbekler arası uzaklık ve maksimum öbekler arası uzaklık değerleri gösterilmiştir.



Şekil 6.1. Temel öbekleme algoritmalarının Wine veri kümesi öbekleme değerleri

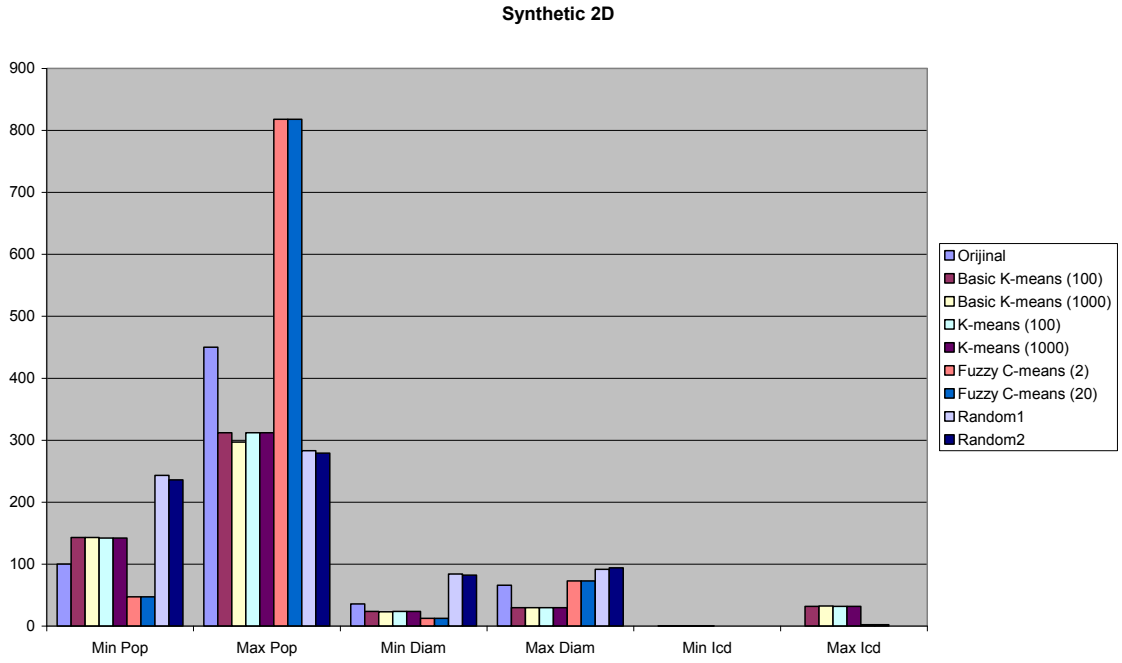
Orijinal olarak 3 farklı sınıftan oluşan Wine veri kümesi, 13 özellik değeri belli 178 örnek veri içermektedir. Boyut sayısı fazla olduğundan öbekler, öbek merkezi çevresinden dağınık ve öbekler arası uzaklıkları birbirine yakın şekilde oluşmuştur. Bu tür bir veri kümesinde k-means algoritmaları populasyon kısıtları için iyi sonuçlar verirken çap ve öbekler arası uzaklık değerlerinde iyi tahminler yapamamaktadır. Öbek merkezine olan uzaklığı minimum yapmak isteyen amaç fonksiyonuyla çalıştığından bu tür veri kümeleri için iyi sonuçlar vermemektedir. Rasgele öbekleme yapan algoritma tesadüfi de olsa k-means'den iyi sonuç vermektedir.



Şekil 6.2. Temel öbekleme algoritmalarının Soybean (Small) veri kümesi öbekleme değerleri

Orijinal olarak 4 farklı sınıftan oluşan Soybean (Small) veri kümesi, 35 boyutlu 47 örnek veri içermektedir. K-means algoritması maksimum çap ve minimum öbekler arası uzaklık değerleri hariç tüm değerleri orijinal veri kümesiyle aynı hesaplamıştır. Söz konusu iki değerde ise diğer algoritmalara kıyasla en yakın değeri bulmuştur.

Yapay olarak oluşturulan Synthetic 2D veri kümesi, orijinal olarak 5 farklı sınıftan oluşan, 2 boyutlu 1300 veri içermektedir. Veri sayısı çok, verilerin birbirlerine olan uzaklığı azdır. Bu nedenele birbirine yakın öbeklerden oluşmaktadır. Karşılaştırmanın daha iyi gözlenmesi için Min Icd ve Max Icd değerleri Şekil 6.4 ve Şekil 6.5'te yeniden gösterilmiştir.



Şekil 6.3. Temel öbekleme algoritmalarının Synthetic 2d veri kümesi öbekleme değerleri

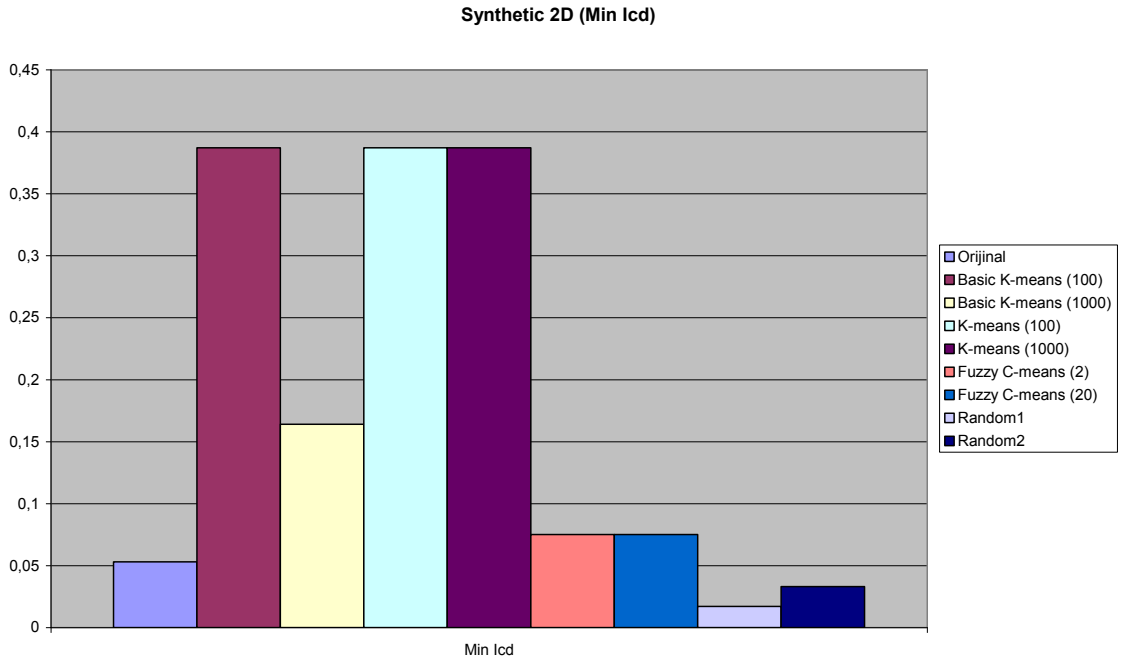
Populasyon değerleri için en iyi hesaplamaları k-means algoritması yaparken, çap ve öbekler arası uzaklık değerleri için en iyi hesaplamaları bulanık c-means algoritması hesaplamıştır.

Bu sonuçlardan yola çıkarak üç veri kümesi üzerinde de en yakın sonuçları k-means algoritmasının hesapladığını söyleyebiliriz.

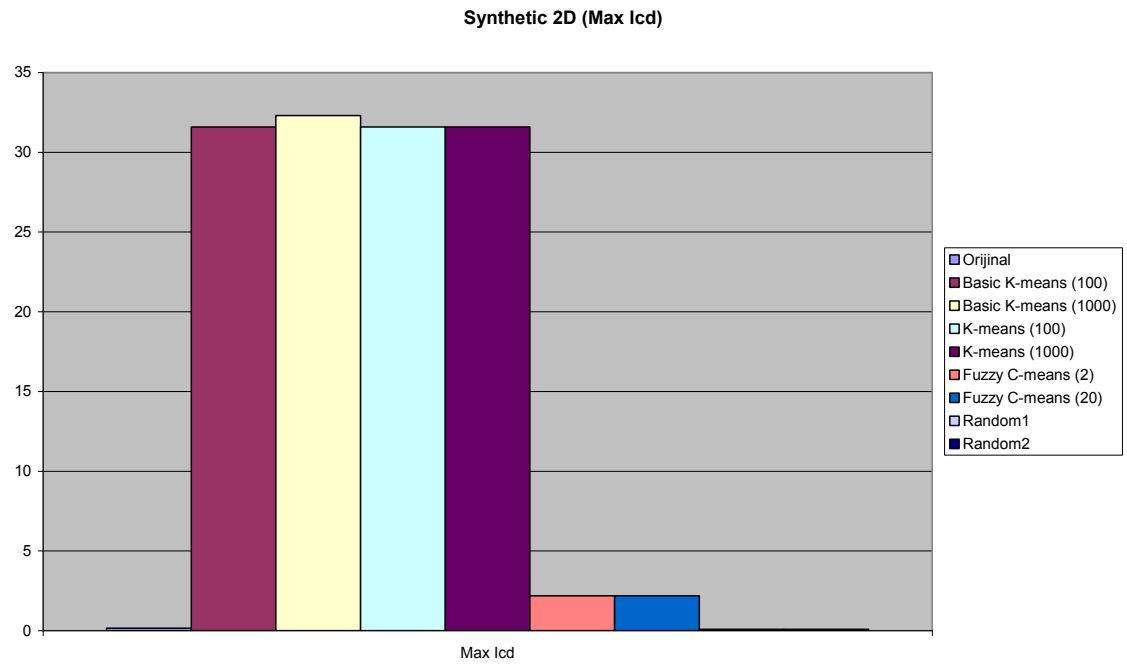
Temel algoritmalara ek olarak genetik algoritmalarla da denemeler yapılmıştır. Bu denemelerde önce genetik algoritma operatörleri olan mutasyon, doğal seleksiyon ve çaprazlama parametrelerine değişik değerler verilerek söz konusu parametrelerin sonuçlar üzerindeki etkileri incelenmiştir. Denemeler sonucunda parametrelerin değiştirilmesinin önemli bir etki yaratmadığı görülmüştür. Sonuç olarak da belirli değerler seçilmiş ve tüm denemelerde bu değerler kullanılmıştır. Mutasyon hızı veri kümesinin beşte biri olarak ayarlanmıştır. Doğal seleksiyon ayarında en iyi kromozom seçen operatör kullanılmış, bu operatör de populasyondaki

kromozomların en iyi beşte ikisini seçecek şekilde ayarlanmıştır. Çaprazlama hızı ise 0.4 olarak ayarlanmıştır yani her beş kromozomdan ikisi çaprazlamaya tabi tutulmaktadır.

K-means algoritmasını göz önünde bulundurarak, bu algoritmanın amaç fonksiyonunu kullanan MUTGA ile veri kümeleri üzerinde denemeler yapılmıştır. Bu denemelerde algoritma 5 kromozomdan oluşan bir populasyonda 10000, 20000 ve 50000 iterasyon çalıştırılmış, son olarak da 20 kromozomluk bir populasyonda 10000 iterasyon çalıştırılmıştır. Wine ve Soybean (Small) veri kümelerinde iterasyon sayısı ile



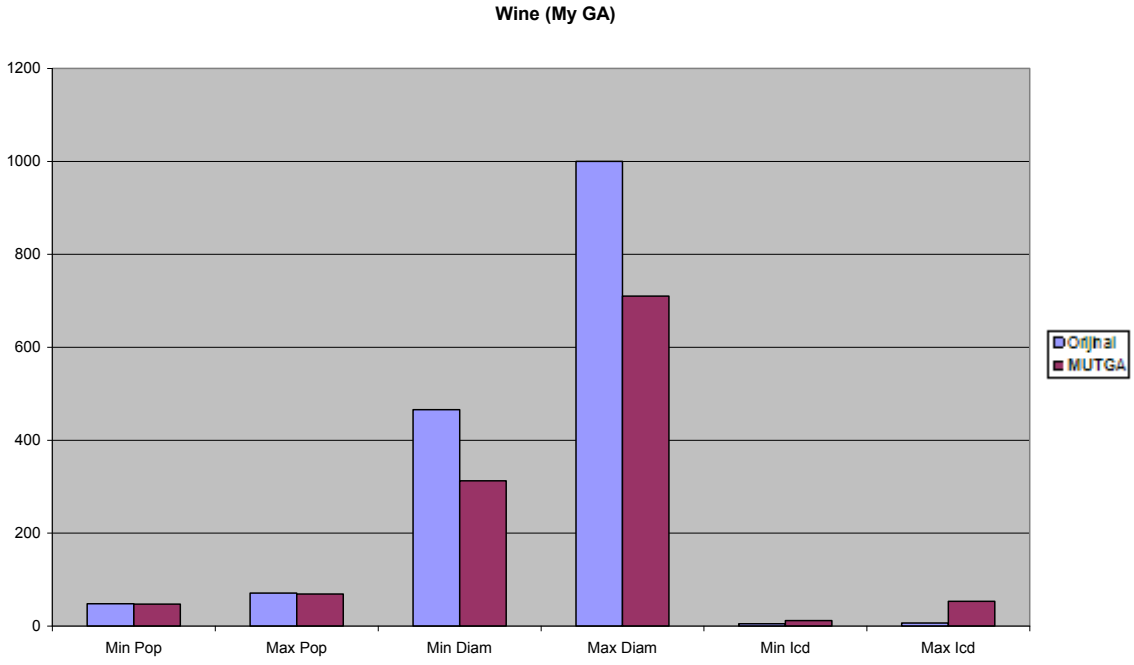
Şekil 6.4. Synthetic 2D kümesinde Min Icd değerleri



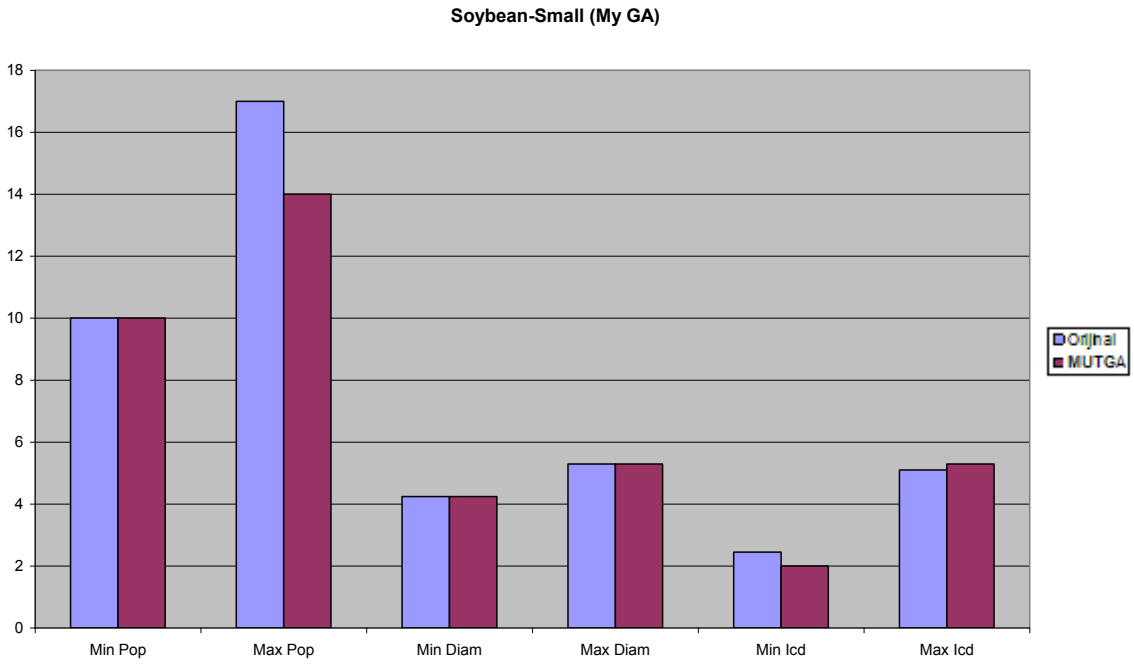
Şekil 6.5. Synthetic 2D kümesinde Max Icd değerleri

populasyon sayısının değişmesi sonuçları etkilememiş, kalabalık bir veri kümesi olan Synthetic 2D kümesinde ise bu değerlerin değişmesi sonucu da değiştirmiştir.

Şekil 6.6 ve Şekil 6.7'de sırasıyla Wine ve Soybean (Small) veri kümelerinde elde edilen sonuç ile orijinal veri kümesindeki değerlerin karşılaştırması görülmektedir.

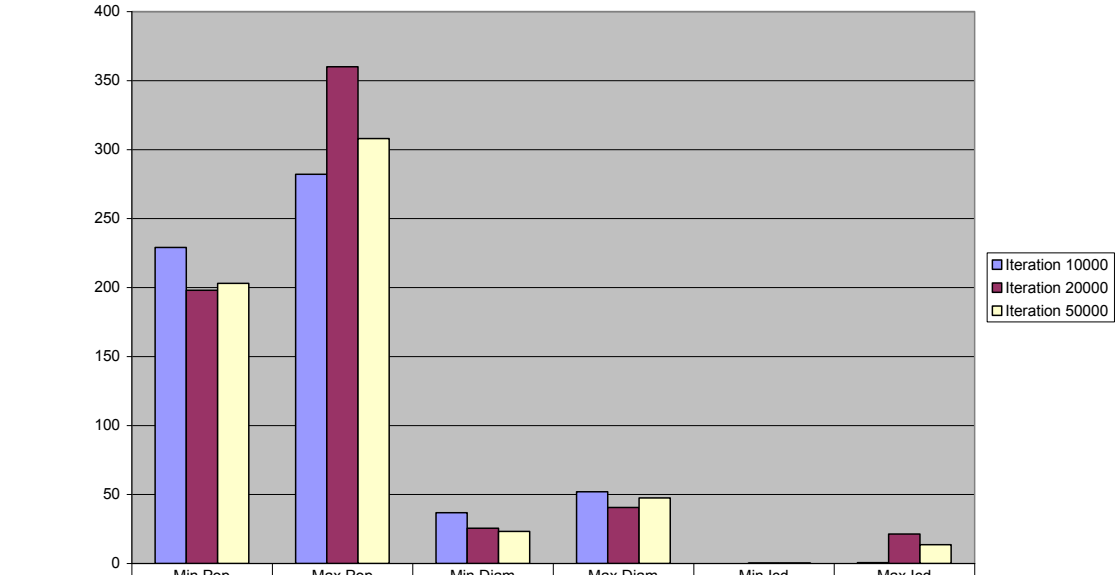


Şekil 6.6. Wine veri kümesinde MUTGA ile elde edilen değerler



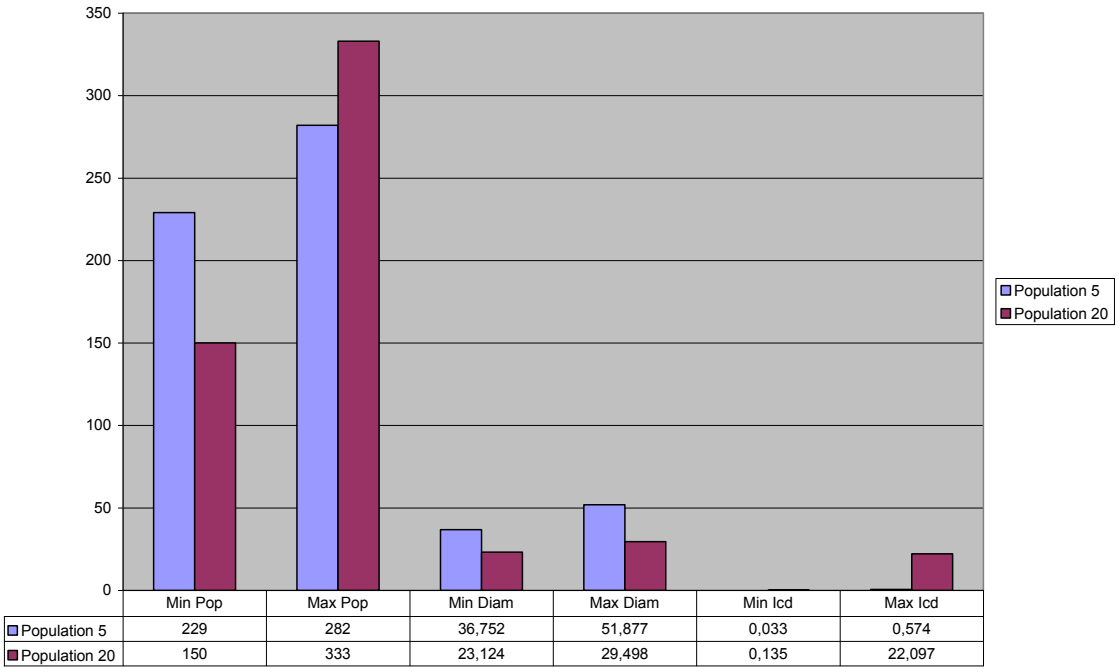
Şekil 6.7. Soybean (Small) veri kümesinde MUTGA ile elde edilen değerler

Şekil 6.8 ve Şekil 6.9'da ise MUTGA algoritmasının Synthetic 2D kümesi üzerinde çalıştırılmasıyla elde edilen değerler görülmektedir. Görüldüğü üzere bu veri kümesinde iterasyon ve kromozom sayısının değişmesi sonucu da etkilemektedir. Genel olarak iterasyon ve kromozom sayılarının artmasının MUTGA için sonucu iyileştirdiğini söyleyebiliriz.



	Min Pop	Max Pop	Min Diam	Max Diam	Min Icd	Max Icd
Iteration 10000	229	282	36,752	51,877	0,033	0,574
Iteration 20000	198	360	25,45	40,484	0,074	21,339
Iteration 50000	203	308	23,246	47,464	0,121	13,526

Şekil 6.8. Synthetic 2D veri kümesinde MUTGA iterasyona bağlı değerleri



	Min Pop	Max Pop	Min Diam	Max Diam	Min Icd	Max Icd
Population 5	229	282	36,752	51,877	0,033	0,574
Population 20	150	333	23,124	29,498	0,135	22,097

Şekil 6.9. Synthetic 2D veri kümesinde MUTGA ile elde edilen değerler

Kısıtların kullanıldığı algoritmalarla da çalışmalar yapılmıştır. EKGA algoritmasının Wine veri kümesi üzerindeki sonuçları Çizelge 5.1'deki gibidir. İterasyon ve kromozom sayısından farklı olarak değişen başka bir değişken de kısıtlardır. Kısıtlar, ".txt" uzantılı dosyalarda saklanmaktadır.

Çizelge 6.1 EKGA Wine kümesindeki değerler

Wine	**Min Pop**	**Max Pop**	**Min Diam**	**Max Diam**	**Min Icd**	**Max Icd**	**Iter.**	**Pop.**
Orijinal	48	71	465,378	1000,026	4,784	6,302	NaN	NaN
EKGA 1	55	63	1160,011	1390,163	2,61	3,885	10000	20
2	56	63	1145,046	1402,191	2,61	3,27	10000	20
3	59	60	1138,192	1402,191	2,62	2,949	10000	20
4	59	60	1085,346	1368,112	5,805	5,976	100	500
5	36	83	1173,161	1368,112	5,976	6,335	100	500
6	59	60	1097,393	1368,112	5,805	5,971	100	750

Veri kümeleri üzerinde kullanılan tüm kısıt dosyalarının içeriği Ek 3'te verilmiştir. Çizelge 6.1'de EKGA'ya ait olan ilk ve beşinci satır için Ek 3'teki 4, ikinci satırı için 2, diğer satırları için de 5 numaralı kısıt dosyaları kullanılmıştır.

Çizelge 6.2 EKGA Soybean (Small) veri kümesindeki değerleri

Soybean (Small)	Min Pop	Max Pop	Min Diam	Max Diam	Min Icd	Max Icd	Iter.	Pop.
Orijinal	10	17	4,242	5,291	2,449	5,099	NaN	NaN
EKGA 1	6	17	7,211	9	1,414	1,732	10000	20
2	10	14	7,937	8,602	1,414	1,732	10000	20
3	11	12	7,81	8,944	1,414	1,414	10000	20
4	11	12	8,124	8,485	1,414	1,732	100	500

Çizelge 6.2'de EKGA'ya ait olan ilk satırı için Ek 3'teki 14, ikinci satırı için 12, üç ve dördüncü satırları için de 15 numaralı kısıt dosyaları kullanılmıştır.

Çizelge 6.3 EKGA Synthetic 2D veri kümesindeki değerleri

Synthetic 2D	Min Pop	Max Pop	Min Diam	Max Diam	Min Icd	Max Icd	Iter.	Pop.
Orijinal	100	450	35, 656	65,859	0,053	0,151	NaN	NaN
EKGA 1	256	261	83, 484	90,471	0,077	0,11	100	20
2	258	261	83, 888	93,876	0,078	0,102	200	20
3	257	261	84, 583	91,398	0,017	0,077	100	20
4	169	381	83, 088	94,245	0,11	0,115	100	20
5	260	260	84	89,874	0,107	0,131	100	100
6	176	369	84, 713	89,502	0,115	0,121	100	100

Çizelge 6.3'de EKGA'ya ait olan ilk, ikinci ve beşimci satırları için Ek 3'teki 25, üçüncü satırı için 22, dört ve altıncı satırları için de 24 numaralı kısıt dosyaları kullanılmıştır. 24 numaralı kısıt dosyası tek bir öbekler arası uzaklık kısıtını içermektedir. Bu kısıtı sağlamak için genetik algoritma amaç fonksiyonunun nasıl değiştiği de Ek 4'te verilmiştir.

Elde edilen öbeklemelerle dosyalardaki kısıtların ne kadar sağlandığı ise Çizelge 6.4'de görülmektedir.

Çizelge 6.4 Veri kümeleri üzerinde tanımlı kısıtların sağlanma değerleri

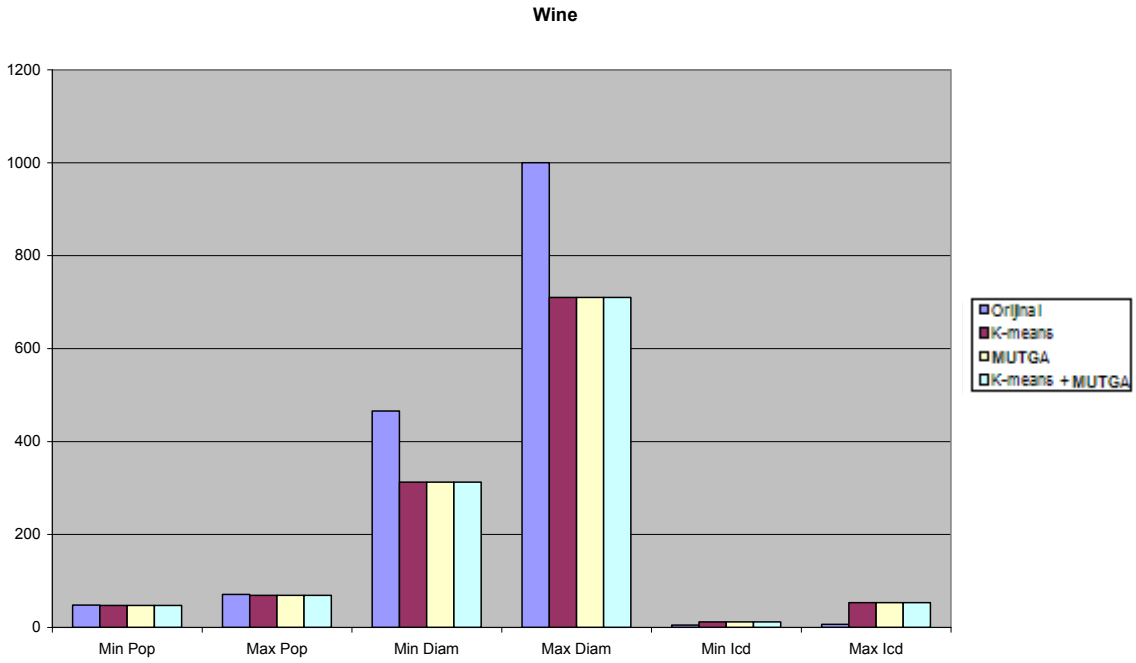
Wine	**Constraint File**	**Result**	**Total**
EKGA			
1	SoftWineDiamIcd.txt	0	2
2	SoftWineConsJustDiam.txt	0	1
3	SoftWineCons.txt	0,5	3
4	SoftWineCons.txt	1,5	3
5	SoftWineDiamIcd.txt	1	2
6	SoftWineCons.txt	1,5	3
Soybean (Small)	**Constraint File**	**Result**	**Total**
EKGA			
1	ConsSoybeanSmallDiamIcd.txt	0,1	2
2	ConsSoybeanSmallJustDiam.txt	0	1
3	ConsSoybeanSmall.txt	0,8	3
4	ConsSoybeanSmall.txt	0,8	3
Synthetic 2D	**Constraint File**	**Result**	**Total**
EKGA			
1	Soft2dCons.txt	0,82	3
2	Soft2dCons.txt	0,83	3
3	Soft2dConsJustPop.txt	0,54	1
4	Soft2dConsJustIcd.txt	0,6	1
5	Soft2dCons.txt	1,12	3
6	Soft2dConsJustIcd.txt	0,66	1

K-means algoritmasıyla genetik algoritmalar birleştirilerek sonuçların iyileştirilmesi hedeflenmiştir. K-means ve MUTGA ile K-means ve EKGA algoritmalarıyla elde edilen değerler sırasıyla Çizelge 6.5 ve Çizelge 6.6'da gösterilmektedir.

Çizelge 6.5 K-means ve MUTGA değerleri

K-means + MUTGA	**Min Pop**	**Max Pop**	**Min Diam**	**Max Diam**	**Min Icd**	**Max Icd**	**Iter.**	**Pop.**
Orijinal Wine	48	71	465, 378	1000,026	4,784	6,302	NaN	NaN
Wine	47	69	312, 425	710,075	11,546	53,33	1000+ 1000	20
Orijinal Soybean (Small)	10	17	4,242	5,291	2,449	5,099	NaN	NaN
Soybean (Small)	10	14	4,242	5,291	2	5,099	1000 + 1000	20
Orijinal Synthetic 2D	100	450	35, 656	65,859	0,053	0,151	NaN	NaN
Synthetic 2D	260	260	83, 555	94,245	0,017	0,077	1000 + 1000	20

K-means ve MUTGA algoritmalarının amaç fonksiyonu yaklaşık olarak aynı olduğundan iki algoritmanın sonuçları ve birlikte çalıştırıldığında elde edilen sonuçlar arasında fazla fark bulunmamaktadır. Bu durum Wine veri kümesi için Şekil 6.10, Soybean (Small) veri kümesi için Şekil 6.11 ve Synthetic 2D veri kümesi için Şekil 6.12'de de görülmektedir.



Şekil 6.10. Wine veri kümesinde K-means ve MUTGA karşılaştırması

Çizelge 6.6 K-means ve EKGA değerleri

Wine	Min Pop	Max Pop	Min Diam	Max Diam	Min Icd	Max Icd	Iter.	Pop.
Orijinal	48	71	465, 378	1000,026	4,784	6,302	NaN	NaN
K-Means + EKGA 1	46	66	312, 425	695,844	10,823	22, 682	1000 + 100	30
2	59	60	1202,321	1390,163	2,61	3,27	1000 + 100	30
3	41	69	312, 425	645,276	4.673	15, 249	1000 + 100	30
4	36	73	447, 126	630,09	4,784	6,786	1000 + 1000	30
5	45	71	343, 432	992,14	6,238	53,33	1000 + 100	100
6	46	66	312, 425	695,844	10,823	22, 682	1000 +	30

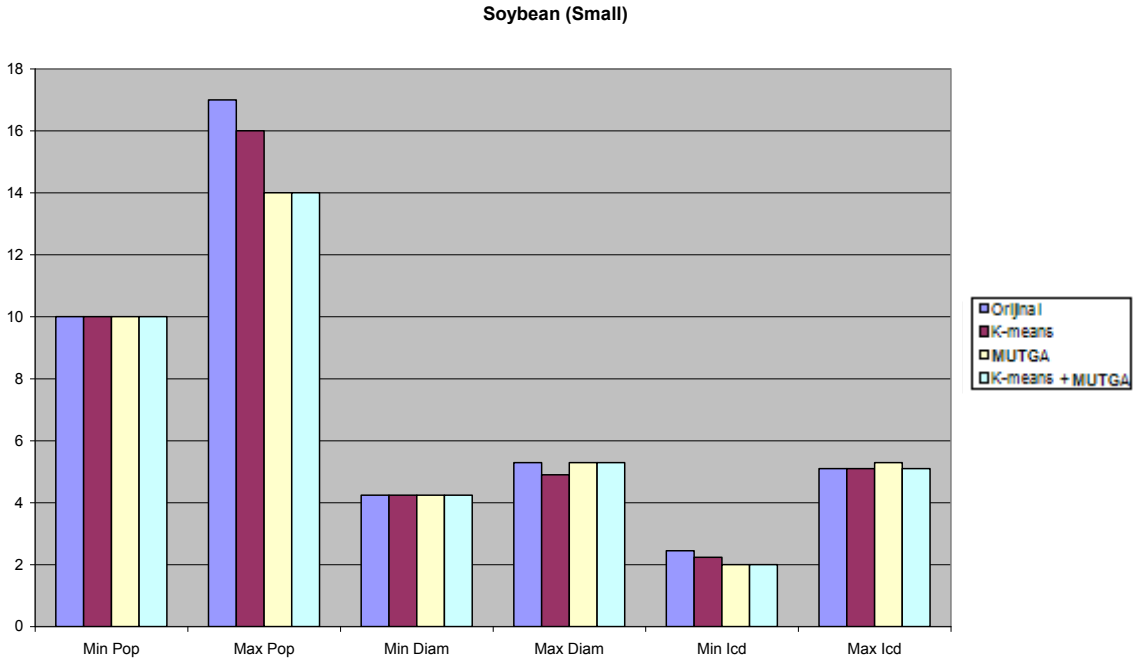
							100	
Soybean (Small)	**Min Pop**	**Max Pop**	**Min Diam**	**Max Diam**	**Min Icd**	**Max Icd**	**Iter.**	**Pop.**
Orijinal	10	17	4,242	5,291	2,449	5,099	NaN	NaN
K-Means + EKGA 1	11	12	7,745	8,602	1,414	1,732	1000 + 100	30
2	11	12	7,681	9	1,414	2,449	1000 + 100	100
3	10	16	4,242	4,898	2,236	5,099	1000 + 100	100
4	7	19	7	9	1,414	2	1000 + 100	100
5	9	14	8	8,66	1,414	1,732	1000 + 1000	100
Synthetic 2D	**Min Pop**	**Max Pop**	**Min Diam**	**Max Diam**	**Min Icd**	**Max Icd**	**Iter.**	**Pop.**
Orijinal	100	450	35, 656	65,859	0,053	0,151	NaN	NaN
K-Means + EKGA 1	143	300	24, 252	32,67	0,226	25, 643	1000 + 100	30
2	144	301	23,72	31,661	0,151	31, 684	1000 + 200	30
3	142	300	24, 252	33,284	0,157	35, 187	1000 + 100	100
4	142	312	23, 672	29,498	0,387	31,5 87	1000 + 100	30

Çizelge 6.6 ‘da görülen değerler elde edilirken kullanılan kısıt dosyaları ve her deneme için bulunan sonuçlar Çizelge 6.7’deki gibidir. K-means, EKGA ve bu iki algoritmanın birlikte kullanılmasıyla elde edilen yeni algoritmanın bu algoritmalarla elde edilen sonuçları geliştirip geliştirmediği sırasıyla Wine veri kümesi için Şekil 6.13, Soybean (Small) veri kümesi için Şekil 6.14 ve Synthetic 2D veri kümesi için Şekil 6.15’de görülmektedir.

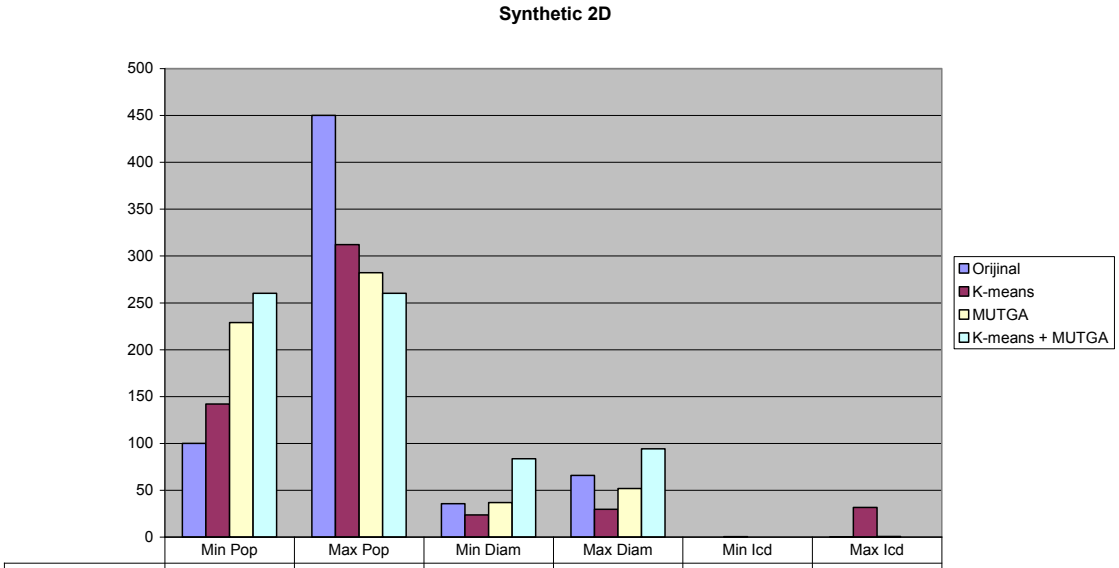
Çizelge 6.7 Veri kümeleri üzerinde kısıtların sağlanma değerleri -2

Wine	**Constraint File**	**Result**	**Total**
K-Means + EKGA 1	SoftWineCons.txt	1,8	3
2	SoftWineConsJustPop.txt	0,5	1
3	SoftWineConsJustDiam.txt	0,63	1
4	SoftWineConsJustDiam.txt	0,65	1
5	SoftWineConsJustIcd.txt	1	1
Soybean (Small)	**Constraint File**	**Result**	**Total**
K-Means + EKGA 1	ConsSoybeanSmall.txt	0,8	3

2	ConsSoybeanSmallJustPop.txt	0,8	1
3	ConsSoybeanSmallJustDiam.txt	0,3	1
4	ConsSoybeanSmallJustIcd.txt	0	1
5	ConsSoybeanSmallJustIcd.txt	0	1
Synthetic 2D	**Constraint File**	**Result**	**Total**
K-Means + EKGA			
1	Soft2dCons.txt	2,43	3
2	Soft2dCons.txt	2,42	3
3	Soft2dCons.txt	2,43	3
4	Soft2dConsJustDiam.txt	1	1

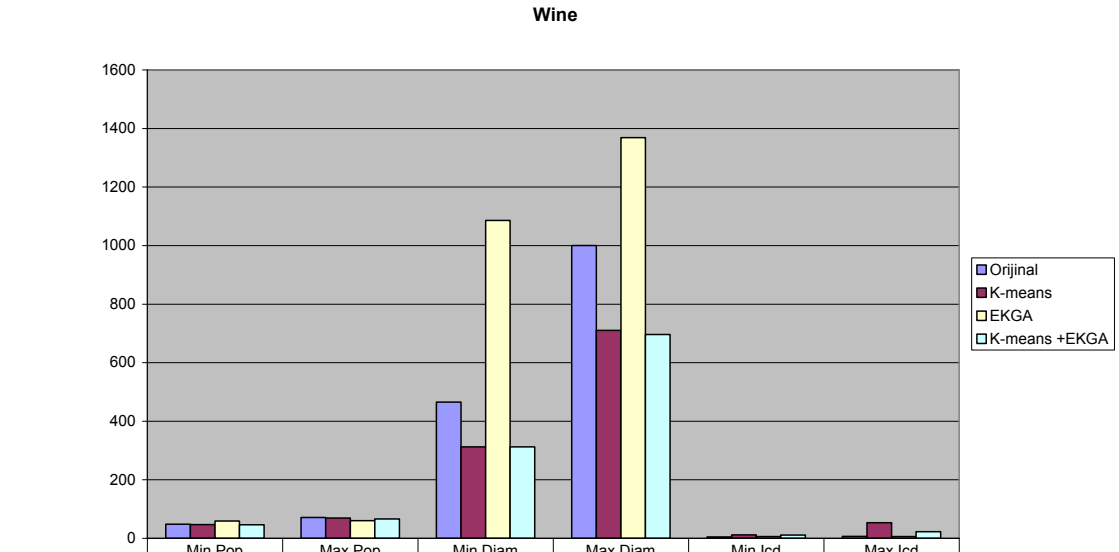


Şekil 6.11. Soybean (Small) veri kümesinde K-means ve MUTGA karşılaştırması



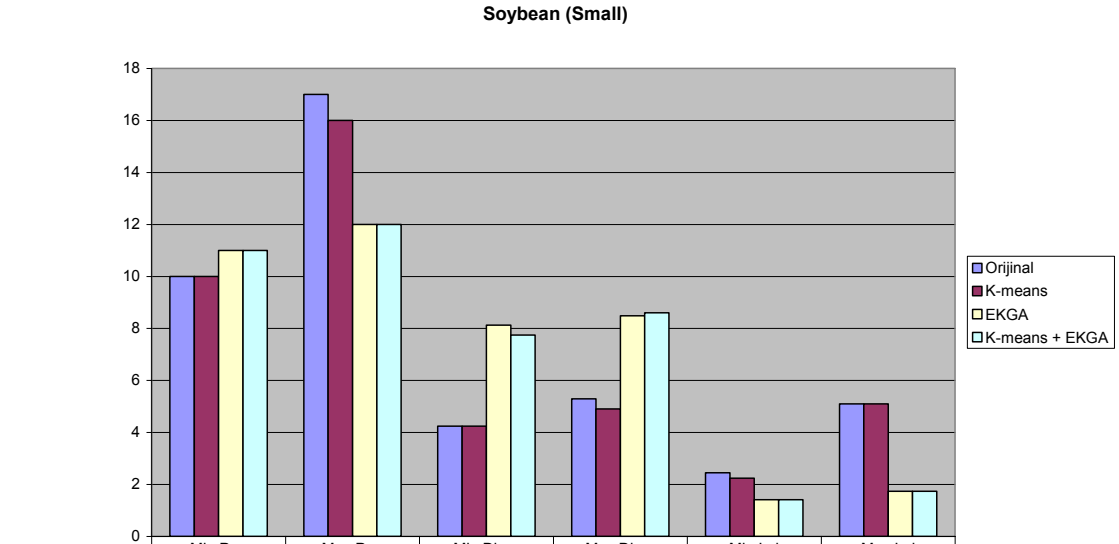
	Min Pop	Max Pop	Min Diam	Max Diam	Min Icd	Max Icd
Orijinal	100	450	35,656	65,859	0,053	0,151
K-means	142	312	23,672	29,498	0,387	31,587
MUTGA	229	282	36,752	51,877	0,033	0,574
K-means + MUTGA	260	260	83,555	94,245	0,017	0,077

Şekil 6.12. Synthetic 2D veri kümesinde K-means ve MUTGA karşılaştırması



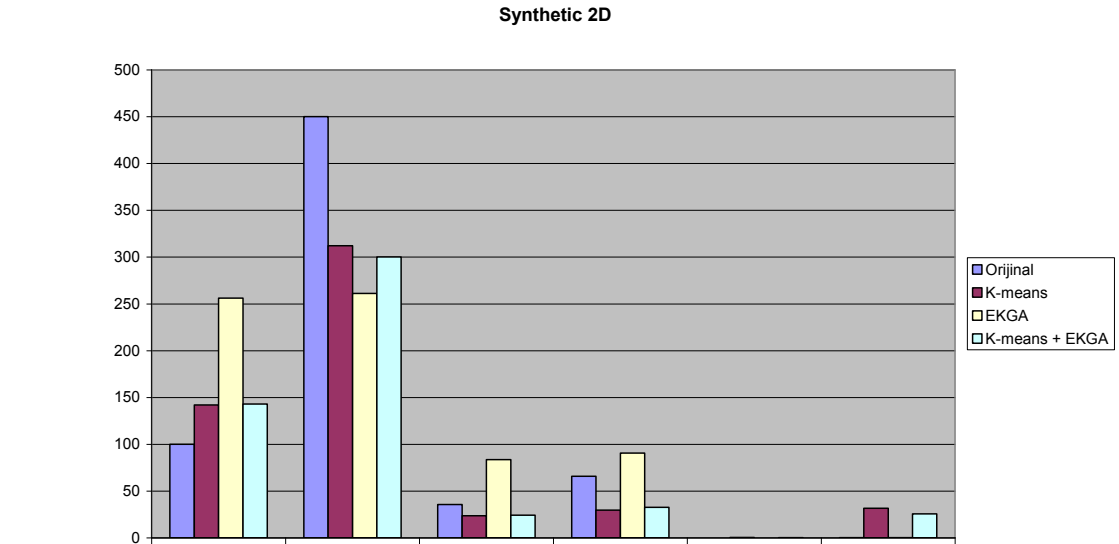
	Min Pop	Max Pop	Min Diam	Max Diam	Min Icd	Max Icd
Orijinal	48	71	465,378	1000,026	4,784	6,302
K-means	47	69	312,425	710,075	11,546	53,33
EKGA	59	60	1085,346	1368,112	5,805	5,976
K-means +EKGA	46	66	312,425	695,844	10,823	22,682

Şekil 6.13. Wine veri kümesinde K-means ve EKGA karşılaştırması



	Min Pop	Max Pop	Min Diam	Max Diam	Min Icd	Max Icd
Orijinal	10	17	4,242	5,291	2,449	5,099
K-means	10	16	4,242	4,898	2,236	5,099
EKGA	11	12	8,124	8,485	1,414	1,732
K-means + EKGA	11	12	7,745	8,602	1,414	1,732

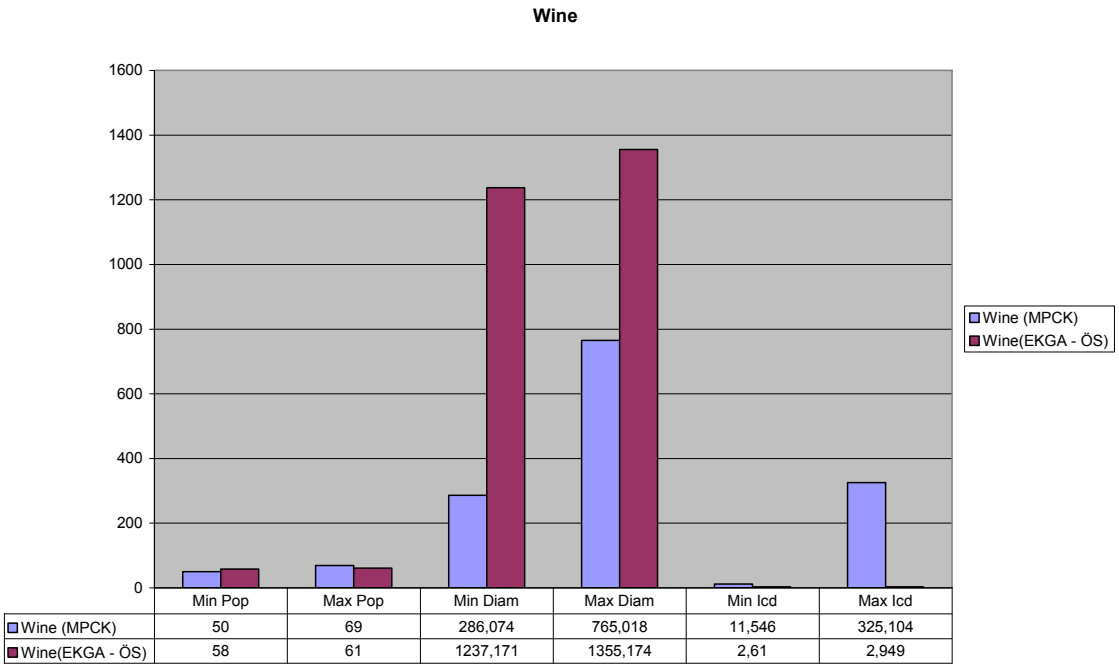
Şekil 6.14. Soybean (Small) veri kümesinde K-means ve EKGA karşılaştırması



	Min Pop	Max Pop	Min Diam	Max Diam	Min Icd	Max Icd
Orijinal	100	450	35,656	65,859	0,053	0,151
K-means	142	312	23,672	29,498	0,387	31,587
EKGA	256	261	83,484	90,471	0,077	0,11
K-means + EKGA	143	300	24,252	32,67	0,226	25,643

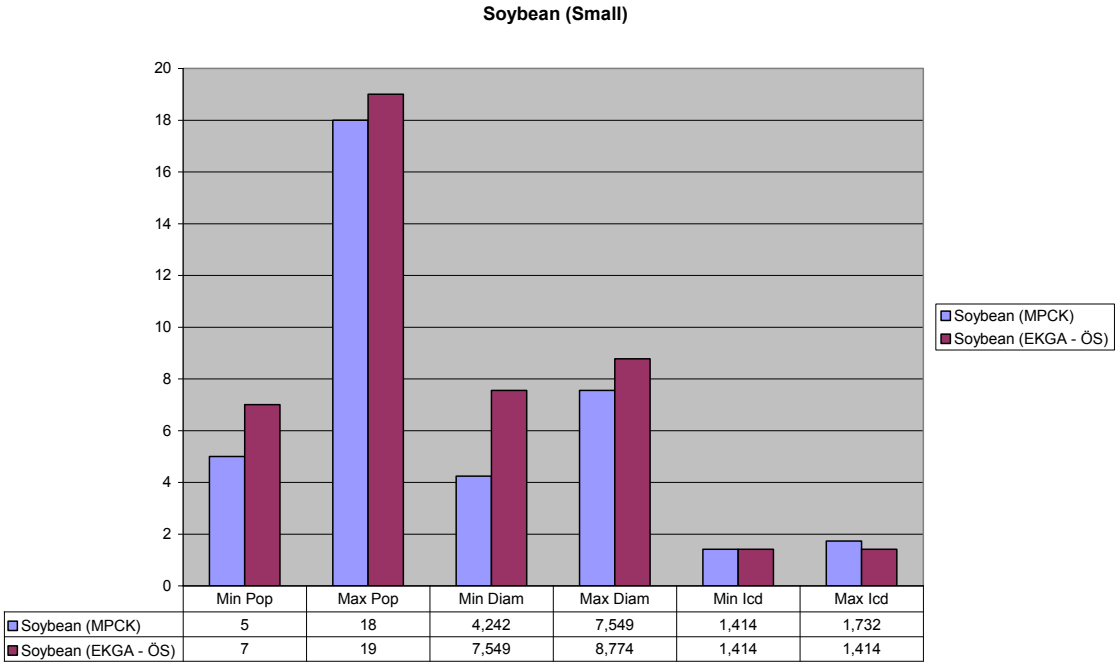
Şekil 6.15.Synthetic 2D veri kümesinde K-means ve EKGA karşılaştırması

Örnek seviyesinde öbekleme yapan algoritmalarla yaptığımız çalışmalar da bulunmaktadır. Gerçek veri kümesi olan Wine ve Soybean (Small) üzerinde MPCK-means ve EKGA – ÖS algoritmaları sırasıyla Şekil 6.16 ve Şekil 6.17'deki gibi davranışlar göstermektedir.



	Min Pop	Max Pop	Min Diam	Max Diam	Min Icd	Max Icd
Wine (MPCK)	50	69	286,074	765,018	11,546	325,104
Wine(EKGA - ÖS)	58	61	1237,171	1355,174	2,61	2,949

Şekil 6.16.Wine veri kümesinde MPCK-means EKGA - ÖS karşılaştırması



	Min Pop	Max Pop	Min Diam	Max Diam	Min Icd	Max Icd
Soybean (MPCK)	5	18	4,242	7,549	1,414	1,732
Soybean (EKGA - ÖS)	7	19	7,549	8,774	1,414	1,414

Şekil 6.17.Soybean (Small) veri kümesinde MPCK-means EKGA - ÖS karşılaştırması

Burada göz önünde bulundurulması gereken bir başka değer de ihlal edilen kısıt sayısı yani penaltı skorudur. EKGA - ÖS algoritması her iki veri kümesinde de daha az ihlale yol açmıştır. İki algoritmanın kullandığı kısıtlar ve penaltı skorları Çizelge 6.8'de gösterilmiştir. Bununla birlikte MPCK-means algoritması yeterli sayıda kısıt verilmediğinde boş öbekler oluşturma eğilimindedir.

Çizelge 6.8 MPCK-means ve EKGA - ÖS penaltı skorları

Wine	**Constraint File**	**Penalty**
MPCK-means	InsConsWine.txt	5
EKGA - ÖS	SoftWineConsJustIns.txt InsConsWine.txt	0
Soybean (Small)	Constraint File	Penalty
MPCK-means	InsConsSoybean2.txt	3
EKGA - ÖS	ConsSoybeanSmallJustIns.txt InsConsSoybean2.txt	1

Yukarıda bahsi geçen çalışmalardan farklı olarak elde edilen öbeklemelerin iyiliğini kıyaslamak için kullanılan algoritmalarla da çalışılmıştır. Bu algoritmalar literatürde öbek doğrulama algoritmaları olarak anılmaktadır. Bir öbeklemenin iyiliğini bir öbekleme algoritmasıyla başka bir öbekleme algoritmasını veya aynı algoritmanın farklı parametrelerle uygulanmasını karşılaştırarak ölçmeye öbek doğrulama denmektedir. Öbek doğrulamanın çeşitli yöntemleri bulunmaktadır. Dunn doğrulama indeksi , Davies-Bouldin doğrulama indeksi, Silhouette doğrulama metodu[41], C indeksi [42], Goodman-Kruskal indeksi [43], izolasyon indeksi [44], Jaccard indeksi [45] ve Rand indeksi [46] bu öbek doğrulama yöntemlerine örnek olarak verilebilir. Bu çalışmada Jaccard ve Rand indeksleri kullanılmıştır. Bu iki yönteme ilave olarak F-measure değerinden de yararlanılmıştır.

$$J(C,K) = \frac{a}{a+b+c} \tag{6.1}$$

Jaccard indeksi, Denklem 6.1'de gösterilen şekilde kullanılmaktadır. Bu denklemde görülen K, elde edilen öbeklemeyi; C, orijinal veri kümesindeki sınıflandırmayı; a, C'de aynı öbekte olup K'da da aynı öbeğe atanan veri çifti sayısını; b, C'de aynı öbekte olup K'da farklı öbeğe atanan veri çifti sayısını; c, C'de farklı öbekte olup K'da aynı öbeğe atanan veri çifti sayısını göstermektedir.

$$J(C,K) = \frac{a+d}{a+b+c+d} \tag{6.2}$$

Rand indeksi, Denklem 6.2'de gösterilen şekilde kullanılmaktadır. Bu denklemde görülen K, elde edilen öbeklemeyi; C, orijinal veri kümesindeki sınıflandırmayı; a, C'de aynı öbekte olup K'da da aynı öbeğe atanan veri çifti sayısını; b, C'de aynı öbekte olup K'da farklı öbeğe atanan veri çifti sayısını; c, C'de farklı öbekte olup K'da aynı öbeğe atanan veri çifti sayısını; d, C'de farklı öbekte olup da K'da da farklı öbeğe atanan veri çifti sayısını göstermektedir.

$$precision(C,K) = \frac{a}{a+c} \tag{6.3}$$

$$recall(C,K) = \frac{a}{a+b} \tag{6.4}$$

$$F-measure(C,K) = \frac{2 \times precision \times recall}{precision + recall} \tag{6.5}$$

F-measure, Denklem 6.5'de gösterilen şekilde kullanılmaktadır. Denklem 6.3 ve Denklem 6.4'de ise F-measure tanımında kullanılan precision ve recall değerleri hesaplanmaktadır. Bu denklemlerde görülen K, elde edilen öbeklemeyi; C, orijinal veri kümesindeki sınıflandırmayı; a, C'de aynı öbekte olup K'da da aynı öbeğe atanan veri çifti sayısını; b, C'de aynı öbekte olup K'da farklı öbeğe atanan veri çifti sayısını; c, C'de farklı öbekte olup K'da aynı öbeğe atanan veri çifti sayısını göstermektedir.

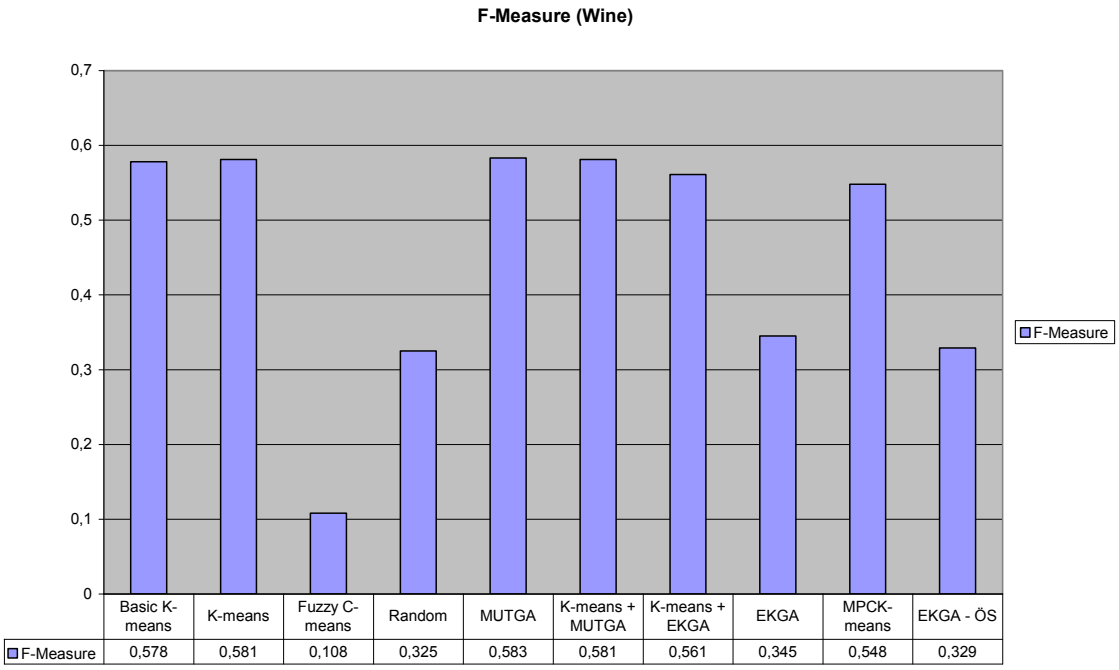
Çizelge 6.9 Wine veri kümesi için öbek doğrulama değerleri

					Cluster Validity Algorithms/indices		
Algoritma / Özellikler	***Fuzz.***	**Iter.**	**Pop.**	**Constraint File**	**F-Measure**	**Jaccard**	**Rand**
Basic K-means	NaN	100	NaN	NaN	0,578	0,412	0,719
K-means	NaN	100	NaN	NaN	0,581	0,414	0,72
	NaN	1000	NaN	NaN	0,581	0,414	0,72
Fuzzy C-means	2	Acc.: 0,00001	NaN	NaN	0,108	0,362	0,522
	20	Acc.: 0,00001	NaN	NaN	0,108	0,362	0,522
Random	NaN	1	NaN	NaN	0,325	0,198	0,552
	NaN	1	NaN	NaN	0,129	0,71	0,555
MUTGA	NaN	10000	5	NaN	0,581	0,414	0,72
	NaN	20000	5	NaN	0,581	0,414	0,72
	NaN	50000	5	NaN	0,583	0,414	0,72
	NaN	10000	20	NaN	0,581	0,414	0,72
K-means + MUTGA	NaN	1000+1000	20	NaN	0,581	0,414	0,72
K-means + EKGA	NaN	1000 + 100	30	SoftWineCons.txt	0,559	0,393	0,705
	NaN	1000 + 100	30	SoftWineCons JustPop.txt	0,329	0,197	0,551
	NaN	1000 +	30	SoftWineCons	0,546	0,38	0,693

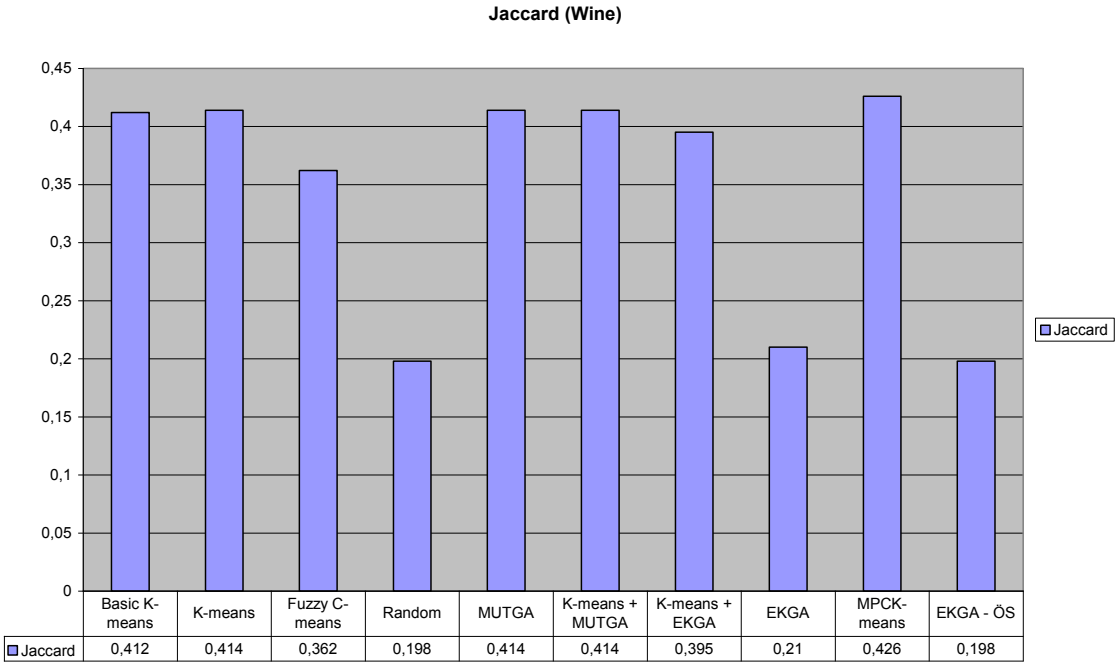
		100		JustDiam.txt			
	NaN	1000 + 1000	30	SoftWineCons JustDiam.txt	0,526	0,362	0,675
	NaN	1000 + 100	100	SoftWineCons JustIcd.txt	0,561	0,395	0,706
EKGA	NaN	10000	20	SoftWineDiam Icd.txt	0,335	0,201	0,555
	NaN	10000	20	SoftWineCons JustDiam.txt	0,33	0,199	0,553
	NaN	10000	20	SoftWineCons .txt	0,328	0,199	0,553
	NaN	100	500	SoftWineCons .txt	0,334	0,2	0,555
	NaN	100	500	SoftWineDiam Icd.txt	0,345	0,21	0,542
	NaN	100	750	SoftWineCons .txt	0,329	0,199	0,553
MPCK-means	NaN	1000	NaN	InsConsWine.t xt	0,548	0,426	0,729
EKGA - ÖS	NaN	100	30	SoftWineCons JustIns.txt InsConsWine.t xt	0,329	0,198	0,553

Çizelge 6.9'da Wine veri kümesinde yapılan çalışmalar sonucunda elde edilen öbek doğrulama sonuçları görülmektedir. Her algoritmadaki en iyi F-measure, Jaccard ve Rand indekslerinin kıyaslaması sırasıyla Şekil 6.18, Şekil 6.19 ve Şekil 6.20'de görülmektedir.

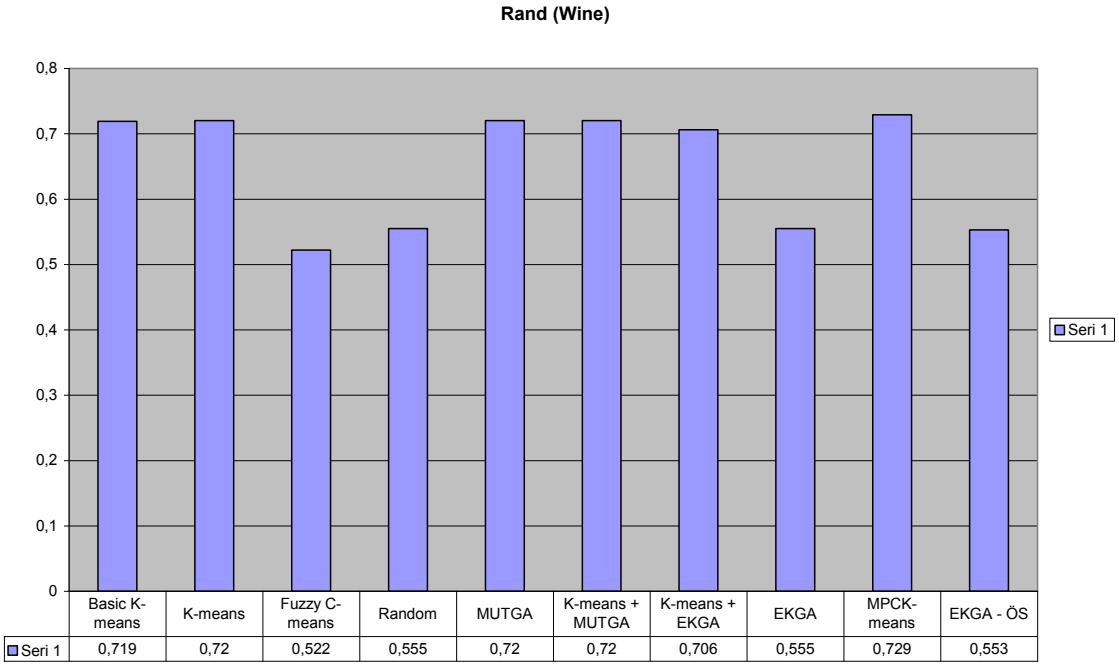
Wine veri kümesi için en iyi F-measure değerini MUTGA, en iyi Jaccard ve Rand değerlerini ise MPCK-means vermektedir.



Şekil 6.18.Wine veri kümesi için F-measure değerleri



Şekil 6.19.Wine veri kümesi için Jaccard değerleri



Şekil 6.20.Wine veri kümesi için Rand değerleri

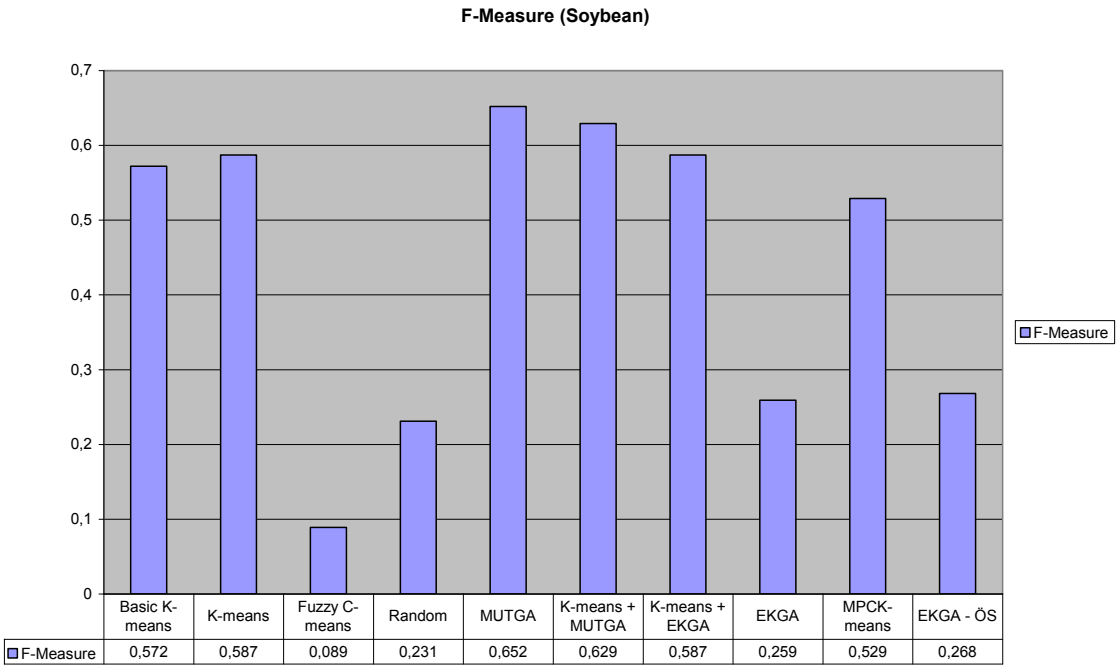
Çizelge 6.10 Soybean (Small) veri kümesi için öbek doğrulama değerleri

					Cluster Validity Algorithms		
Algoritma/ Özellikler	***Fuzz.***	**Iter.**	**Constraint File**	**Pop.**	**F-Measure**	**Jaccard**	**Rand**
Basic K-means	NaN	100	NaN	NaN	0,572	0,432	0,805
	NaN	1000	NaN	NaN	0,496	0,334	0,742
K-means	NaN	100	NaN	NaN	0,587	0,534	0,852
	NaN	100	NaN	NaN	0,587	0,534	0,852
Fuzzy C-means	2	Acc.: 0,00001	NaN	NaN	0,089	0,317	0,658
	20	Acc.: 0,00001	NaN	NaN	0,089	0,317	0,658
Random	NaN	1	NaN	NaN	0,224	0,124	0,626
	NaN	1	NaN	NaN	0,231	0,144	0,639
MUTGA	NaN	10000	NaN	5	0,534	0,498	0,837
	NaN	20000	NaN	5	0,652	0,498	0,837
	NaN	50000	NaN	5	0,534	0,498	0,837
	NaN	10000	NaN	20	0,568	0,498	0,837
K-means + MUTGA	NaN	1000 + 1000	NaN	20	0,629	0,498	0,837
K-means + Soft GA	NaN	1000 + 100	ConsSoybeanSmall.txt	30	0,261	0,15	0,645
	NaN	1000 + 100	ConsSoybeanSmallJustP	100	0,264	0,158	0,651

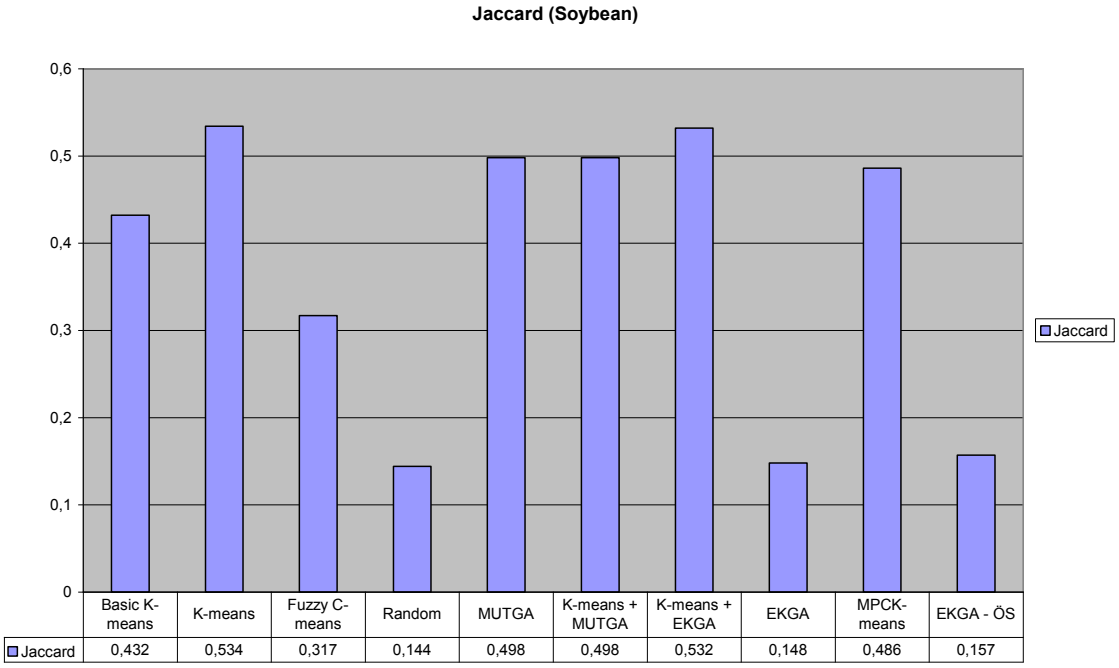
			op.txt				
	NaN	1000 + 100	ConsSoybeanSmallJustDiam.txt	100	0,587	0,532	0,852
	NaN	1000 + 100	ConsSoybeanSmallJustIcd.txt	100	0,26	0,15	0,622
	NaN	1000 + 1000	ConsSoybeanSmallJustIcd.txt	100	0,213	0,14	0,634
EKGA (No K-means)	NaN	10000	ConsSoybeanSmallDiamIcd.txt	20			
	NaN	10000	ConsSoybeanSmallJustDiam.txt	20	0,259	0,148	0,639
	NaN	10000	ConsSoybeanSmall.txt	20	0,209	0,116	0,62
	NaN	100	ConsSoybeanSmall.txt	500	0,211	0,121	0,624
MPCKmeans	NaN	1000	InsConsSoybean.txt	NaN	0,243	0,486	0,764
	NaN	1000	InsConsSoybean2.txt	NaN	0,529	0,36	0,749
My Instance Soft GA	NaN	100	ConsSoybeanSmallJustIns.txt InsConsSoybean2.txt	30	0,268	0,157	0,622

Çizelge 6.10'da Soybean (Small) veri kümesinde yapılan çalışmalar sonucunda elde edilen öbek doğrulama sonuçları görülmektedir. Her algoritmadaki en iyi F-measure, Jaccard ve Rand indekslerinin kıyaslaması sırasıyla Şekil 6.21, Şekil 6.22 ve Şekil 6.23'de görülmektedir.

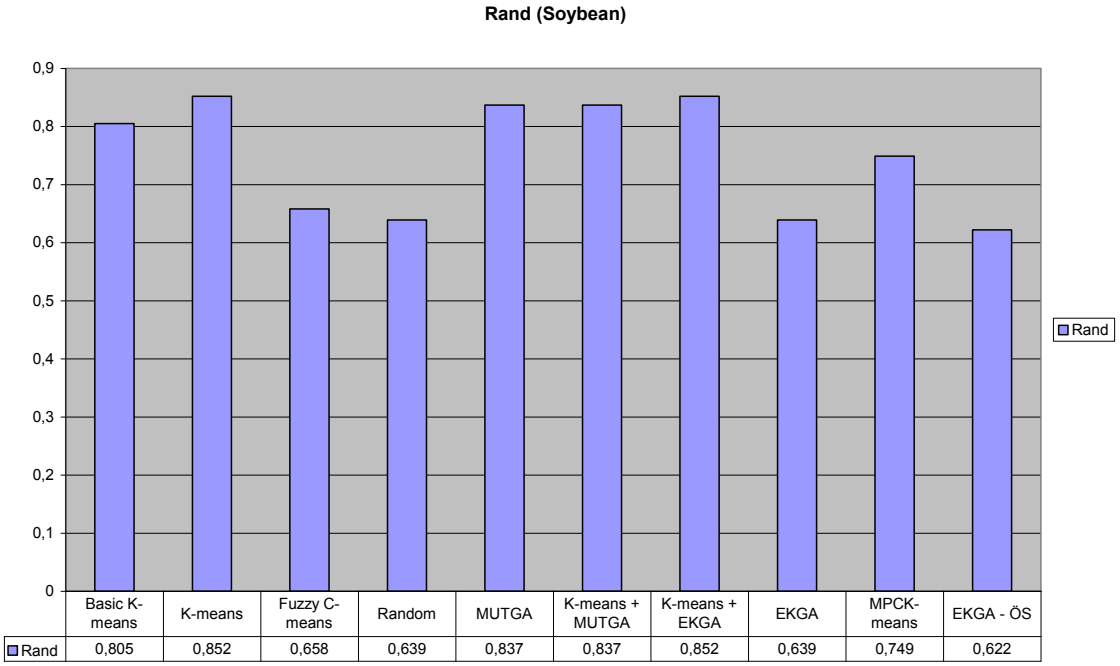
Soybean (small) veri kümesi için en iyi F-measure değerini k-means ve MUTGA, en iyi Jaccard değerini k-means ve EKGA, en iyi Rand değerini ise k-means ile k-means ve EKGA birlikte vermektedir.



Şekil 6.21.Soybean(Small) veri kümesi için F-measure değerleri



Şekil 6.22.Soybean (Small) veri kümesi için Jaccard değerleri



Şekil 6.23.Soybean (Small) veri kümesi için Rand değerleri

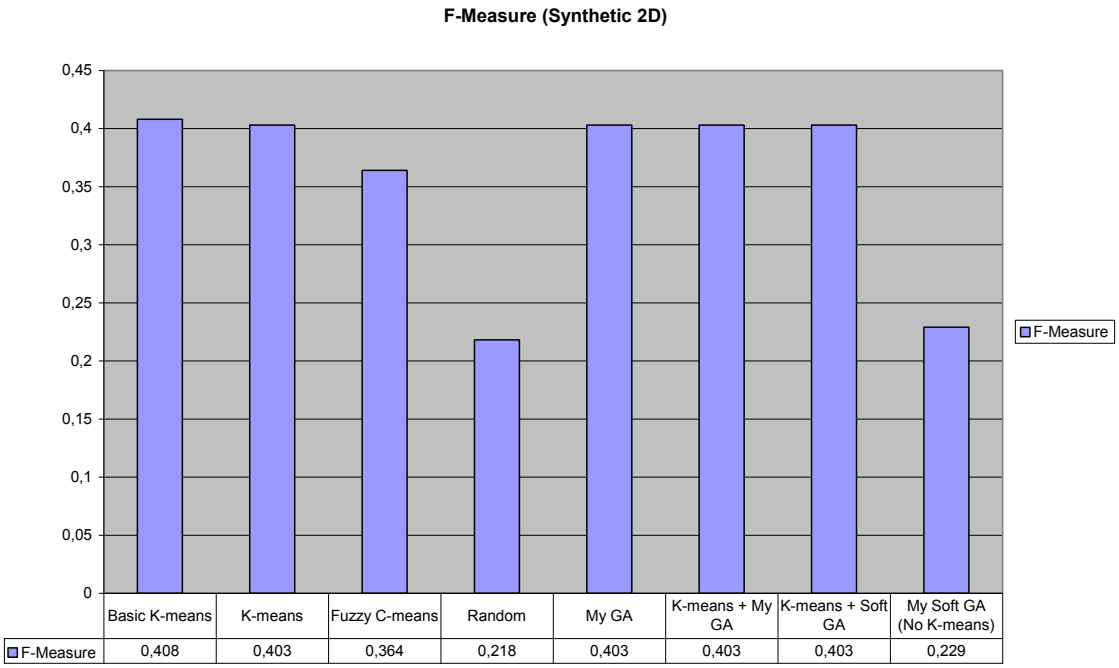
Çizelge 6.11 Synthetic 2D veri kümesi için öbek doğrulama değerleri

					Cluster Validity Algorithms/indices		
Algoritma/ Özellikler	***Fuzz.***	**Iter.**	**Constraint File**	**Pop.**	**F-Measure**	**Jaccard**	**Rand**
Basic K-means	NaN	100	NaN	NaN	0,4	0,26	0,736
	NaN	1000	NaN	NaN	0,408	0,266	0,74
K-means	NaN	100	NaN	NaN	0,403	0,262	0,737
	NaN	100	NaN	NaN	0,403	0,262	0,737
Fuzzy C-means	2	Acc.: 0,00001	NaN	NaN	0,364	0,223	0,566
	20	Acc.: 0,00001	NaN	NaN	0,364	0,223	0,566
Random	NaN	1	NaN	NaN	0,218	0,122	0,656
	NaN	1	NaN	NaN	0,216	0,121	0,655
MUTGA	NaN	10000	NaN	5	0,255	0,171	0,689
	NaN	20000	NaN	5	0,368	0,244	0,728
	NaN	50000	NaN	5	0,357	0,24	0,729
	NaN	10000	NaN	20	0,403	0,277	0,744
K-means + MUTGA	NaN	1000 + 1000	NaN	20	0,403	0,262	0,737
K-means + Soft GA	NaN	1000 + 100	Soft2dCons.txt	30	0,399	0,259	0,736

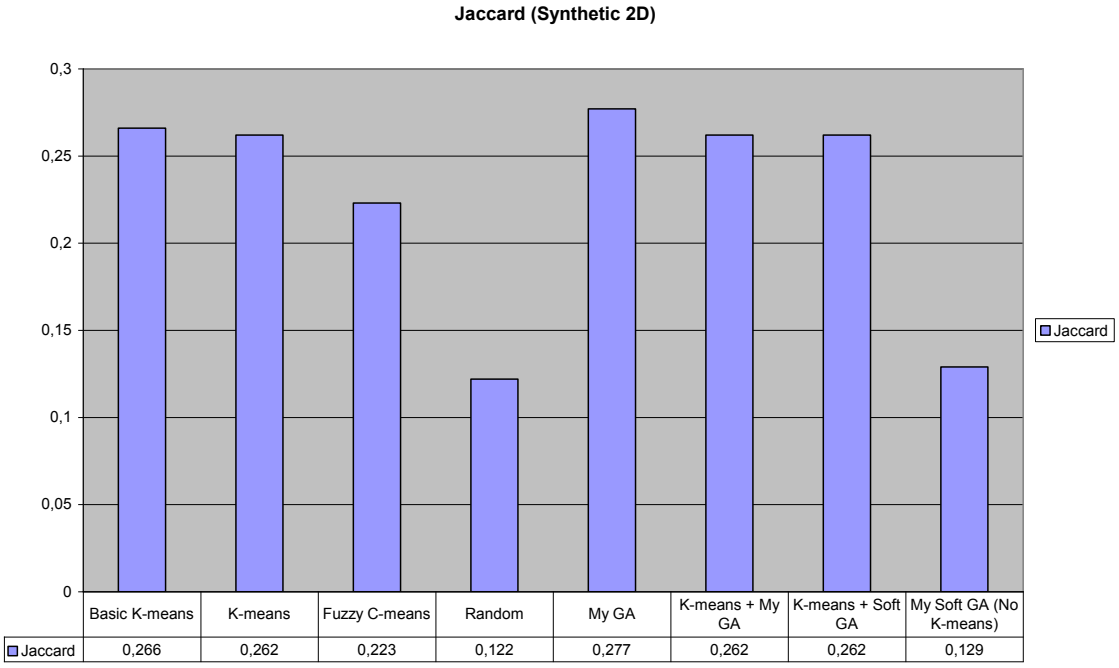
	NaN	1000 + 200	Soft2dCons.txt	30	0,398	0,258	0,735
	NaN	1000 + 100	Soft2dCons.txt	100	0,399	0,257	0,734
	NaN	1000 + 100	Soft2dConsJustDiam.txt	30	0,403	0,262	0,737
EKGA (No K-means)	NaN	100	Soft2dCons.txt	20	0,217	0,121	0,656
	NaN	200	Soft2dCons.txt	20	0,215	0,121	0,656
	NaN	100	Soft2dConsJustPop.txt	20	0,217	0,121	0,656
	NaN	100	Soft2dConsJustIcd.txt	20	0,228	0,129	0,647
	NaN	100	Soft2dCons.txt	100	0,217	0,122	0,657
	NaN	100	Soft2dConsJustIcd.txt	100	0,229	0,129	0,645

Çizelge 5.11'de Synthetic 2D veri kümesinde yapılan çalışmalar sonucunda elde edilen öbek doğrulama sonuçları görülmektedir. Her algoritmadaki en iyi F-measure, Jaccard ve Rand indekslerinin kıyaslaması sırasıyla Şekil 6.24, Şekil 6.25 ve Şekil 6.26 'de görülmektedir.

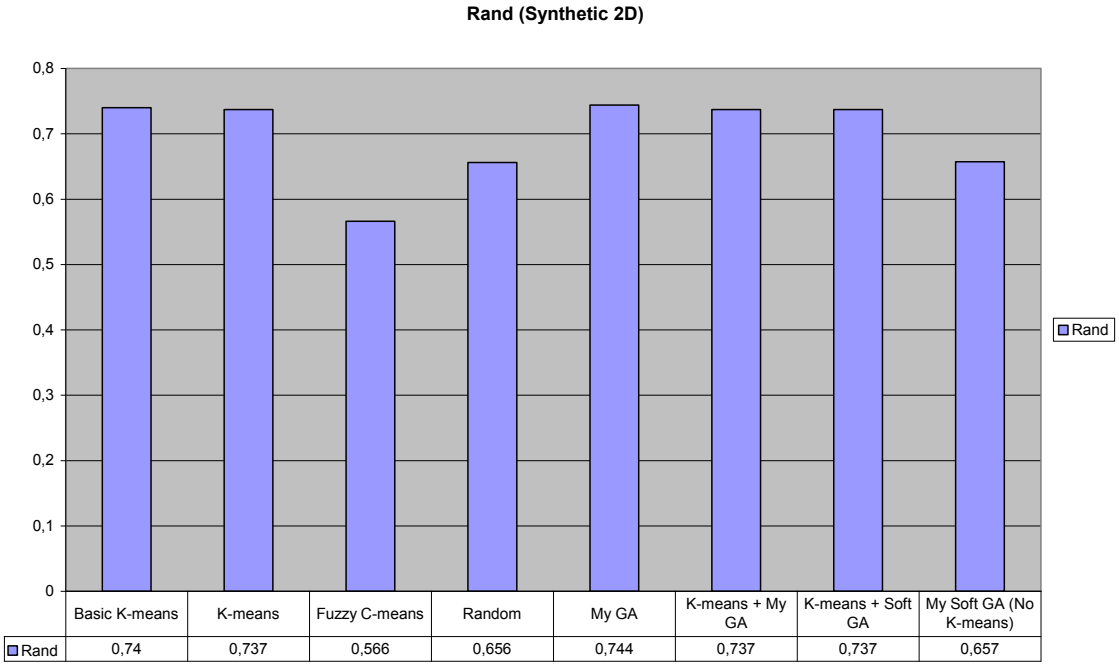
Synthetic 2D veri kümesi için en iyi F-measure değerini temel k-means, en iyi Jaccard ve Rand değerlerini ise MUTGA vermektedir. Synthetic 2D verilen öbek merkezleri etrafında belirli bir varyans girilerek dağıtılan noktaların kümesi olduğundan, öbekleme yaparken noktaları mümkün olduğunca öbek merkezleri etrafında toplayan k-means ve MUTGA bu veri kümesi üzerinde yapılan çalışamalarda iyi sonuç vermiştir.



Şekil 6.24.Synthetic 2D veri kümesi için F-measure değerleri



Şekil 6.25.Synthetic 2D veri kümesi için Jaccard değerleri



Şekil 6.26.Synthetic 2D veri kümesi için Rand değerleri

7.SONUÇ

Bu çalışmada kesişmeyen kısmi öbeklemeler elde etmek amacıyla esnek kısıtlar kullanan algoritmalar bulunması hedeflenmiştir. Verilen kısıtlar katı kısıtlarla ifade edildiğinde tüm kısıtların aynı anda sağlanması genellikle mümkün olmamaktadır. Bu nedenle öbeklemeler elde edilirken esnek kısıtlardan faydalanılmıştır. Esnek kısıtları matematiksel olarak tanımlamak için yarı halka modelleri kullanılmaktadır. Kısıt kombinasyonlarının sağlanma değeri ise bulanık, olasılıksal ve ağırlıklı yarı halka modellerinden herhangi birisi kullanılarak hesaplanmaktadır.

Kısıtların yaratılarak algoritmalarda kullanılması ve elde edilen öbekleme sonuçlarının hesaplanabilmesi için NetBeans IDE kullanılarak Java programlama diliyle bir araç geliştirilmiştir. Bu araç kullanılarak esnek kısıtlarla öbekleme yapan algoritmalar çalıştırılmaktadır. K-means, fuzzy c-means gibi geleneksel algoritmalar, Bilenko ve diğerlerinin çalışmasında bahsedilen ve örnek seviyesinde öbekleme yapan MPCK-means algoritmasının yanısıra genetik algoritmalardan faydalanılarak elde edilen algoritmalar araçta bulunmaktadır.

Gelişirilen bu araç kullanılarak ikisi UCI veri kümelerinden alınmış birisi de yapay olarak oluşturulmuş değişik özelliklere sahip üç veri kümesi üzerinde çeşitli denemeler yapılmıştır. Denemelerin sonuçları kullanılan algoritmanın amaç fonksiyonu (benzerlik faktörü, uzaklık birimi, uzaklığın hesap edilme şekli vb.), veri kümesinin boyutu, veri örnek sayısı, veri değerleri (dağılımı), eğer varsa kullanılan genetik operatörler, algoritmaya verilen kısıtlar, kromozom sayıları, kullanılan yarım halka çeşidi ve algoritmanın çalıştırılma sayısı gibi birçok değişkene bağımlı olduğundan birebir kıyaslama yaparak net sonuçlar elde etmek mümkün görünmemektedir.

Bununla birlikte yapılan çalışmalarda görülmüştür ki öngörülen algoritmalar gerçekleşmesi istenen şekilde veri kümeleri elde etme amacına hizmet etmektedirler. Ayrıca öngörülen algoritmalar kısıt kullanmayan algoritmalarla kıyaslandığında istenilen kısıtlardaki öbeklemeyi sağlamaya daha yakın sonuçlar vermektedir.

Kısıtlar olmasına rağmen öbek doğrulama algoritmalarında verdikleri sonuçlar kısıt kullanmayan algoritmalarla kıyaslandığında kötü değildir.

Gelecekte yapılacak çalışmalarda daha fazla veri kümesi kullanılarak daha net sonuçlar elde edilmesi planlanmaktadır. Genetik algoritmaların amaç fonksiyonları değiştirilerek daha hızlı ve daha yüksek sonuçlar elde edilebilir. Bu çalışmada kullanılan ağırlıklı yarı halka modeli yerine bulanık ve/veya olasılıksal yarı halka modelleri kullanılabilir. Kullanılan kısıtların önem dereceleri birbirinden farklı olacak şekilde verilerek daha iyi öbeklemeler bulunabilir. Gerçekleştirilen algoritmalar paralelleştirilerek çok daha büyük veri kümeleri üzerinde iyi sonuçlar alınması mümkün gözükmektedir.

Bununla birlikte kısmi öbeklemeler elde etmek amaçlandığından çöp öbeklerini çeşitli şekillerde oluşturan algoritmalar, örnek ve öbek seviyesinden başka bu çalışmada da bahsi geçen model seviyesinde kısıtlarla öbekleme yapan algoritmalar ve öbeklemenin sonuçlarını iyileştirecek öbek sayısını hesaplayabilen algoritmalar tasarlanması hedeflenmektedir.

KAYNAKLAR

[1] P. Tan, M.Steinbach, V. Kumar, Introduction to Data Mining, *Addison-Wesley*, 2006.

[2] J. Han, M. Kamber, Data Mining Concepts and Techniques, *Elsevier*, 2006.

[3] M.H. Dunham, Data Mining Introductry and Advanced Topics, *Prentice Hall*, Upper Saddle River, New Jersey, 2003.

[4] A.K. Jain, M.N. Murty, P.J. Flynn, Data Clustering: A Review, ACM Computing Surveys, 31(3), 264-323, 1999.

[5] D. Cohn, R. Caruana, A. McCallum, Semi-supervised Clustering with User Feedback, Constrained Clustering: Advances in Algorithms, Theory and Applications, S.Basu, I. Davidson, K. Wagstaff, *Chapman & Hall*, 2009.

[6] J. B. MacQueen (1967): "Some Methods for classification and Analysis of Multivariate Observations, Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability", Berkeley, University of California Press, 1:281-297.

[7] Öbekleme Analizi erişim adresi: http://en.wikipedia.org/wiki/Cluster_analysis.

[8] P.S. Bradley, K.P. Bennett, A. Demiriz, Constrained K-Means Clustering, MSR-TR-2000-65, Microsoft Research, Mayıs 2000.

[9] K. Wagstaff, C. Cardie, S. Rogers, S. Schroedl, Constrained K-Means Clustering with Background Knowledge, 18th Annual International Conference on Machine Learning(ICML ’2001), Williamstown, MA, U.S.A., 577-584, 2001.

[10] K.L. Wagstaff, S. Basu, I. Davidson, When is Constrained Clustering Beneficial and Why?, 21st Annual National Conference on Artificial Intelligence(AAAI ‘06), Boston, Massachusetts, U.S.A., Temmuz 2006.

[11] M. Bilenko, S. Basu, R.J. Mooney, Integrating Constraints and Metric Learning in Semi-Supervised Clustering, 21st Annual International Conference on Machine Learning(ICML ’2004), Banff, Alberta, Canada, 2004.

[12] S. Basu, A. Banerje, A. Mooney, Semi-supervised Clustering by Seeding, 19th International Conference on Machine Learning(ICML ‘2002), Sydney, NSW, Australia, 27-34, Temmuz 2002.

[13] M. H. C. Law, A. Topchy, A. K. Jain, Clustering with Soft and Group Constraints, Joint IAPR International Workshop on Syntactical and Structural Pattern Recognition and Statistical Pattern Recognition, 2004.

[14] K. Wagstaff, C. Cardie, Clustering with Instance-Level Constraints, 17th Annual International Conference on Machine Learning(ICML ’2000), Stanford, CA, U.S.A., 1003-1010, 2000.

[15] R. Ge, 2008, Clustering with Cluster-Level Constraints, *Doktora Tezi*, Simon Fraser University, School of Computing Science, Vancouver.

[16] O. Abul, F. Bonchi, R. Pensa, Soft Constraint Based Clustering. (Draft)

[17] A. Demiriz, K.P. Bennett, M.J. Embrechts, Semi-Supervised Clustering Using Genetic Algorithms, Artificial Neural Networks in Engineering,(ANNIE ’1999), Missouri, U.S.A., 809-814,1999.

[18] Y. Chiou, L.W. Lan, Theory and Methodolgy Genetic Clustering Algorithms, European Journal of Operational Research, 135, 413- 427, 2001.
[19] M.C. Cowgill, R.J. Harvey, L.T. Watson, A Genetic Algorithm Approach to Cluster Analysis, Computers and Mathematics with Applications, 37, 99-108, 1999.
[20] L.Y. Tseng, S.B. Yang, A Genetic Clustering Algorithm for Data with Non-Spherical-Shape Clusters, Pattern Recognition, 33, 1251-1259, 2000.
[21] R. Piccarreta, Clustering Large Categorical Data Set via Genetic Algorithms, 41th Scientific Meetings, Milan, Italy, 2002.
[22] G. Gan, J. Wu, Z. Yang, A Genetic Fuzzy k-Modes Algorithm for Clustering Categorical Data, Expert Systems with Applications, 36, 1615-1620, 2009.
[23] Y. Marinakis, M. Marinaki, M. Doumpos, N. Matsatsinis, C. Zopounidis, A Hybrid Stochastic Genetic-GRASP Algorithm for Clustering Analysis, Operational Research, 8(1), 33-46, Şubat 2008.
[24] M.C. Naldi, A.C.P.L.F. deCarvalho, Clustering Using A Genetic Algorithm Combining Validation Criteria, European Symposium on Artificial Neural Networks (ESANN'2007), 139-144, Bruges, Belgium, Nisan 2007.
[25] S. Bandyopadhyay, U. Maulik, Nonparametric Genetic Clustering: Comparison of Validity Indices, IEEE Transactions on Systems, Man and Cybernetics-Part C: Applications and Reviews, 31(1), 120-125, Şubat 2001.
[26] D.L. Davies, D. W. Bouldin, A Cluster Separation Measure, IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), volume PAMI-1 (2) , 224-227, 1979.
[27] J.C. Dunn, A fuzzy relative of the ISODATA process and its use in detecting compacted well-separated clusters, J.Cybern, volume 3, 32-57, 1973.
[28] S.R Thangiah, S. Salhi, Genetic Clustering: An Adaptive Heuristic for the Multidepot Vehicle Routing Problem, Applied Artificial Intelligence, 15(4), 361-383, 2001.
[29] R. Entriken, S. Vössner, Genetic Algorithms with Cluster Analysis for Production Simulation, 29th Confernce on Winter Simulation, Atlanta, Georgia, U.S.A, 1307-1314, Aralık 1997.
[30] C.M. Ringle, R.Schlittgen, A Genetic Algorithm Segmentation Approach for Uncovering and Separating Groups of Data in PLS Path Modeling, 5th International Symposium on PLS and Related Methods(PLS '2007), Matforsk, Aas, Norway, Eylül 2007.
[31] N.A. Rosenberg, T. Burke, K. Elo, M.W. Feldman, P.J. Friedlin, M.A.M. Groenen, J. Hillel, A. Maki-Tanila, M. Tixier-Boichard, A. Vignal, K. Wimmers, S. Weigend, Empirical Evaluation of Genetic Clustering Methods Using Multilocus Genotypes from 20 Chicken Breeds, Genetics, 159, 699-713, Ekim 2001.
[32] R.L. Haupt, S.E. Haupt, Practical Genetic Algorithms, *Wiley*, 2004.
[33] E. Falkenauer, Genetic Algorithms and Grouping Problems, *Wiley*, 1998.
[34] "Java Genetic Algorithms Package" erişim adresi: http://jgap.sourceforge.net/.
[35] D. Gondek, S. Vaithyanathan, A. Garg, Clustering with Model-Level Constraints, SIAM Conference on Data Mining(SDM '2005), Newport Beach, CA, U.S.A., 126-137, Nisan 2005.

[36] A free Java chart library "JFreeChart" erişim adresi: http://www.jfree.org/jfreechart/.
[37] UCI Machine Learning Repository Wine veri kümesi erişim adresi: http://archive.ics.uci.edu/ml/datasets/Wine/.
[38] Temel bileşenler analizi (Principal Componenet Analysis - PCA) erişim adresi: http://en.wikipedia.org/wiki/Principal_component_analysis.
[39] JavaStatSoft İstatistiksel Yazılım Programı erişim adresi: http://sourceforge.net/projects/javastatsoft/.
[40] UCI Machine Learning Repository Soybean (Small) veri kümesi erişim adresi: http://archive.ics.uci.edu/ml/datasets/Soybean+%28Small%29.
[41] P.J. Rousseeuw, Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis, Journal of Computational and Applied Mathematics, 20, 53-65, 1987.
[42] L. Hubert, J. Schultz, Quadratic Assignment as a General Data-Analysis Strategy, British Journal of Mathematical and Statistical Psychologie, 29, 190-241, 1976.
[43] L. Goodman, W. Kruskal, Measures of Associations for Cross-Validations, J. Am. Stat. Assoc., 49, 732-764, 1954.
[44] E.J. Pauwels, G. Frederix, Finding Salient Regions in Images: Nonparametric Clustering for Image Segmentation and Grouping, Computer Vision and Image Understanding, 75, 73-85, 1999.
[45] P.Jaccard, The Distribution of Flora in the Alpine Zone, New Phytologist, 11, 37-50, 1912.
[46] W.M. Rand, Objective Criteria for the Evaluation of Clustering Methods, Journal of the American Statistical Association, 846-850, 1971.

EKLER

Ek 1: Örnek kısıt dosyası içeriği

```
pop (P) >= 45 [0.2]  (42,5, 47,5)
pop (P) <= 80 [-0.3]  (78,33, 81,67)
diam (P) <= 5 [-0.05]  (0, 15)
diam (P) >= 3 [0.4]  (1,75, 4,25)
icd (P) <= 4 [-0.8]  (3,38, 4,62)
icd (P) >= 1 [0.9]  (0,44, 1,56)
trash (P) <= 4 [-1]  (3,5, 4,5)
size (P) >= 5 [0.7]  (4,29, 5,71)
ICML (P) <= 3 [-0.5]  (2, 4)
ICCL (P) <= 4 [-0.6]  (3,17, 4,83)
```

Ek 2: Yapay veri kümesi oluşturan MATLAB kodu.

```
mu = [20 50]
sigma = [2 1 ; 2 1]
r = mvnrnd(mu,sigma,50)
plot(r(:,1),r(:,2),'+')
hold on;
mu = [40 10]
sigma = [2 1 ; 2 1]
r = mvnrnd(mu,sigma,20)
plot(r(:,1),r(:,2),'o')
hold on;
mu = [60 30]
sigma = [2 1 ; 2 1]
r = mvnrnd(mu,sigma,30)
plot(r(:,1),r(:,2),'x')
hold on;
mu = [40 30]
sigma = [2 1 ; 2 1]
r = mvnrnd(mu,sigma,10)
plot(r(:,1),r(:,2),'*')
hold on;
mu = [25 15]
sigma = [2 1 ; 2 1]
r = mvnrnd(mu,sigma,25)
plot(r(:,1),r(:,2),'h')
```

Ek 3: Kısıt dosya içerikleri.

1) SoftWineConsJustPop.txt

pop (P) <= 60 [-0.04] (47,5, 72,5)

2) SoftWineConsJustDiam.txt

diam (P) <= 720 [-0.0017] (425,88, 1014,12)

3) SoftWineConsJustIcd.txt

icd (P) >= 5 [0.7] (4,29, 5,71)

4) SoftWineConsDiamIcd.txt

diam (P) <= 720 [-0.0017] (425,88, 1014,12)

icd (P) >= 5 [0.7] (4,29, 5,71)

5)SoftWineCons.txt

pop (P) <= 60 [-0.04] (47,5, 72,5)

diam (P) <= 720 [-0.0017] (425,88, 1014,12)

icd (P) >= 5 [0.7] (4,29, 5,71)

6) SoftWineConswithTrash.txt

pop (P) <= 60 [-0.04] (47,5, 72,5)

diam (P) <= 720 [-0.0017] (425,88, 1014,12)

icd (P) >= 5 [0.7] (4,29, 5,71)

trash (P) <= 8 [-0.5] (7, 9)

7)SoftWineConsJustIns.txt

ICML (P) <= 2 [-0.5] (1, 3)

ICCL (P) <= 5 [-0.5] (4, 6)

8) SoftWineConsAll.txt

ICML (P) <= 2 [-0.5] (1, 3)

ICCL (P) <= 5 [-0.5] (4, 6)

pop (P) <= 60 [-0.04] (47,5, 72,5)

diam (P) <= 720 [-0.0017] (425,88, 1014,12)

icd (P) >= 5 [0.7] (4,29, 5,71)

9) SoftWineConsAllwithTrash.txt

ICML (P) <= 2 [-0.5] (1, 3)

ICCL (P) <= 5 [-0.5] (4, 6)

pop (P) <= 60 [-0.04] (47,5, 72,5)

diam (P) <= 720 [-0.0017] (425,88, 1014,12)

icd (P) >= 5 [0.7] (4,29, 5,71)

trash (P) <= 8 [-0.5] (7, 9)

10) InsConsWine.txt

22 44 1

157 172 1

25 158 -1

73 172 -1

40 79 -1

66 154 -1

11) ConsSoybeanSmallJustPop.txt

pop (P) <= 15 [-0.1] (10, 20)

12) ConsSoybeanSmallJustDiam.txt

diam (P) <= 4.5 [-0.5] (3,5, 5,5)

13) ConsSoybeanSmallJustIcd.txt

icd (P) >= 3.5 [0.5] (2,5, 4,5)

14) ConsSoybeanSmallDiamIcd.txt

diam (P) <= 4.5 [-0.5] (3,5, 5,5)

icd (P) >= 3.5 [0.5] (2,5, 4,5)

15) ConsSoybeanSmall.txt

pop (P) <= 15 [-0.1] (10, 20)

diam (P) <= 4.5 [-0.5] (3,5, 5,5)

icd (P) >= 3.5 [0.5] (2,5, 4,5)

16) ConsSoybeanSmallwithTrash.txt

pop (P) <= 15 [-0.1] (10, 20)

diam (P) <= 4.5 [-0.5] (3,5, 5,5)

icd (P) >= 3.5 [0.5] (2,5, 4,5)

trash (P) <= 8 [-0.5] (7, 9)

17) InsConsSoybean.txt

1 10 1

10 11 -1

22 23 1

30 41 -1

18) InsConsSoybean2.txt

1 10 1

10 11 -1

22 23 1

30 41 -1

42 46 1

30 35 -1

19) ConsSoybeanSmallJustIns.txt

ICML (P) <= 2 [-0.5] (1, 3)

ICCL (P) <= 2 [-0.5] (1, 3)

20) ConsSoybeanSmallAll.txt

pop (P) <= 15 [-0.1] (10, 20)

diam (P) <= 4.5 [-0.5] (3,5, 5,5)

icd (P) >= 3.5 [0.5] (2,5, 4,5)

ICML (P) <= 2 [-0.5] (1, 3)

ICCL (P) <= 2 [-0.5] (1, 3)

21) ConsSoybeanSmallAllwithTrash.txt

trash (P) <= 8 [-0.5] (7, 9)

pop (P) <= 15 [-0.1] (10, 20)

diam (P) <= 4.5 [-0.5] (3,5, 5,5)

icd (P) >= 3.5 [0.5] (2,5, 4,5)

ICML (P) <= 2 [-0.5] (1, 3)

ICCL (P) <= 2 [-0.5] (1, 3)

22) Soft2dConsJustPop.txt

pop (P) <= 275 [-0.003] (108,33, 441,67)

23) Soft2dConsJustDiam.txt

diam (P) <= 50 [-0.03] (33,33, 66,67)

24) Soft2dConsJustIcd.txt

icd (P) >= 0.1 [10] (0,05, 0,15)

25) Soft2dCons.txt

icd (P) >= 0.1 [10] (0,05, 0,15)

diam (P) <= 50 [-0.03] (33,33, 66,67)

pop (P) <= 275 [-0.003] (108,33, 441,67)

Ek 4: Synthetic 2D veri kümesinde Ek 3, 24 numaralı kısıtın sağlanmasına yönelik sırasıyla 20 ve 100 kromozomluk populasyonlarla yapılmış denemelerde genetik algoritma amaç fonksiyonunun değerindeki değişim.

Populasyon: 20 iken

```
0.Best fitness so far:0.0
1.Best fitness so far:0.0
2.Best fitness so far:0.0
3.Best fitness so far:0.0
4.Best fitness so far:0.032242559929226644
5.Best fitness so far:0.10735429956491815
6.Best fitness so far:0.10735429956491815
7.Best fitness so far:0.10735429956491815
8.Best fitness so far:0.10735429956491815
9.Best fitness so far:0.1926034919057087
10.Best fitness so far:0.1926034919057087
11.Best fitness so far:0.24631683332215726
12.Best fitness so far:0.24631683332215726
13.Best fitness so far:0.24631683332215726
14.Best fitness so far:0.25420982922523583
15.Best fitness so far:0.27717801101163864
16.Best fitness so far:0.277625942275594
17.Best fitness so far:0.277625942275594
18.Best fitness so far:0.277625942275594
19.Best fitness so far:0.28710640106658125
20.Best fitness so far:0.28710640106658125
21.Best fitness so far:0.28710640106658125
22.Best fitness so far:0.28710640106658125
23.Best fitness so far:0.30884776596092656
24.Best fitness so far:0.30884776596092656
25.Best fitness so far:0.30884776596092656
26.Best fitness so far:0.3307558190587836
27.Best fitness so far:0.3307558190587836
28.Best fitness so far:0.3307558190587836
29.Best fitness so far:0.3307558190587836
30.Best fitness so far:0.3307558190587836
```

31.Best fitness so far:0.3307558190587836
32.Best fitness so far:0.3988079147404142
33.Best fitness so far:0.3988079147404142
34.Best fitness so far:0.3988079147404142
35.Best fitness so far:0.4856792338281337
36.Best fitness so far:0.4856792338281337
37.Best fitness so far:0.4856792338281337
38.Best fitness so far:0.4856792338281337
39.Best fitness so far:0.4856792338281337
40.Best fitness so far:0.4856792338281337
41.Best fitness so far:0.4856792338281337
42.Best fitness so far:0.4856792338281337
43.Best fitness so far:0.4856792338281337
44.Best fitness so far:0.507576609196559
45.Best fitness so far:0.507576609196559
46.Best fitness so far:0.507576609196559
47.Best fitness so far:0.507576609196559
48.Best fitness so far:0.507576609196559
49.Best fitness so far:0.522387793843414
50.Best fitness so far:0.522387793843414
51.Best fitness so far:0.522387793843414
52.Best fitness so far:0.522387793843414
53.Best fitness so far:0.522387793843414
54.Best fitness so far:0.522387793843414
55.Best fitness so far:0.522387793843414
56.Best fitness so far:0.522387793843414
57.Best fitness so far:0.522387793843414
58.Best fitness so far:0.5426347355138348
59.Best fitness so far:0.5426347355138348
60.Best fitness so far:0.5426347355138348
61.Best fitness so far:0.5426347355138348
62.Best fitness so far:0.5426347355138348
63.Best fitness so far:0.5426347355138348
64.Best fitness so far:0.5426347355138348
65.Best fitness so far:0.5426347355138348
66.Best fitness so far:0.5426347355138348
67.Best fitness so far:0.5426347355138348
68.Best fitness so far:0.5426347355138348
69.Best fitness so far:0.5426347355138348

```
70.Best fitness so far:0.5700832687692723
71.Best fitness so far:0.5700832687692723
72.Best fitness so far:0.5700832687692723
73.Best fitness so far:0.5700832687692723
74.Best fitness so far:0.5700832687692723
75.Best fitness so far:0.5700832687692723
76.Best fitness so far:0.5700832687692723
77.Best fitness so far:0.5700832687692723
78.Best fitness so far:0.5700832687692723
79.Best fitness so far:0.5713684207592036
80.Best fitness so far:0.5713684207592036
81.Best fitness so far:0.5713684207592036
82.Best fitness so far:0.5713684207592036
83.Best fitness so far:0.5713684207592036
84.Best fitness so far:0.5713684207592036
85.Best fitness so far:0.5713684207592036
86.Best fitness so far:0.5713684207592036
87.Best fitness so far:0.5736743761495091
88.Best fitness so far:0.5736743761495091
89.Best fitness so far:0.5736743761495091
90.Best fitness so far:0.5736743761495091
91.Best fitness so far:0.5736743761495091
92.Best fitness so far:0.5736743761495091
93.Best fitness so far:0.6010475501539532
94.Best fitness so far:0.6010475501539532
95.Best fitness so far:0.6010475501539532
96.Best fitness so far:0.6010475501539532
97.Best fitness so far:0.6010475501539532
98.Best fitness so far:0.6010475501539532
99.Best fitness so far:0.6010475501539532
The best solution contained the following:
Result = 0.6010475501539532
```

Populasyon: 100 iken

```
0.Best fitness so far:0.032242559929226644
1.Best fitness so far:0.10735429956491815
2.Best fitness so far:0.1926034919057087
```

3.Best fitness so far:0.1926034919057087
4.Best fitness so far:0.25420982922523583
5.Best fitness so far:0.25420982922523583
6.Best fitness so far:0.27717801101163864
7.Best fitness so far:0.27717801101163864
8.Best fitness so far:0.277625942275594
9.Best fitness so far:0.277625942275594
10.Best fitness so far:0.2847867410322429
11.Best fitness so far:0.2847867410322429
12.Best fitness so far:0.30884776596092656
13.Best fitness so far:0.30884776596092656
14.Best fitness so far:0.3307558190587836
15.Best fitness so far:0.4856792338281337
16.Best fitness so far:0.4856792338281337
17.Best fitness so far:0.4856792338281337
18.Best fitness so far:0.4856792338281337
19.Best fitness so far:0.507576609196559
20.Best fitness so far:0.507576609196559
21.Best fitness so far:0.507576609196559
22.Best fitness so far:0.5219394711038619
23.Best fitness so far:0.5219394711038619
24.Best fitness so far:0.5219394711038619
25.Best fitness so far:0.5426347355138348
26.Best fitness so far:0.5426347355138348
27.Best fitness so far:0.5426347355138348
28.Best fitness so far:0.5426347355138348
29.Best fitness so far:0.5426347355138348
30.Best fitness so far:0.5426347355138348
31.Best fitness so far:0.5426347355138348
32.Best fitness so far:0.5713684207592036
33.Best fitness so far:0.5713684207592036
34.Best fitness so far:0.5736743761495091
35.Best fitness so far:0.5736743761495091
36.Best fitness so far:0.5840965216252827
37.Best fitness so far:0.5840965216252827
38.Best fitness so far:0.5840965216252827
39.Best fitness so far:0.5840965216252827
40.Best fitness so far:0.5840965216252827
41.Best fitness so far:0.5840965216252827

42.Best fitness so far:0.5840965216252827
43.Best fitness so far:0.6010475501539532
44.Best fitness so far:0.6010475501539532
45.Best fitness so far:0.6010475501539532
46.Best fitness so far:0.6010475501539532
47.Best fitness so far:0.6010475501539532
48.Best fitness so far:0.6010475501539532
49.Best fitness so far:0.6068909381235543
50.Best fitness so far:0.6068909381235543
51.Best fitness so far:0.6068909381235543
52.Best fitness so far:0.6068909381235543
53.Best fitness so far:0.6068909381235543
54.Best fitness so far:0.6114415738130266
55.Best fitness so far:0.6114415738130266
56.Best fitness so far:0.633540581540851
57.Best fitness so far:0.633540581540851
58.Best fitness so far:0.633540581540851
59.Best fitness so far:0.633540581540851
60.Best fitness so far:0.633540581540851
61.Best fitness so far:0.6384577362818395
62.Best fitness so far:0.6384577362818395
63.Best fitness so far:0.6384577362818395
64.Best fitness so far:0.6384577362818395
65.Best fitness so far:0.6384577362818395
66.Best fitness so far:0.6384577362818395
67.Best fitness so far:0.6384577362818395
68.Best fitness so far:0.6400081852776327
69.Best fitness so far:0.6400081852776327
70.Best fitness so far:0.6400081852776327
71.Best fitness so far:0.6400081852776327
72.Best fitness so far:0.6569933645877368
73.Best fitness so far:0.659839269424858
74.Best fitness so far:0.659839269424858
75.Best fitness so far:0.659839269424858
76.Best fitness so far:0.659839269424858
77.Best fitness so far:0.659839269424858
78.Best fitness so far:0.659839269424858
79.Best fitness so far:0.659839269424858
80.Best fitness so far:0.659839269424858

```
81.Best fitness so far:0.659839269424858
82.Best fitness so far:0.659839269424858
83.Best fitness so far:0.659839269424858
84.Best fitness so far:0.659839269424858
85.Best fitness so far:0.659839269424858
86.Best fitness so far:0.659839269424858
87.Best fitness so far:0.659839269424858
88.Best fitness so far:0.659839269424858
89.Best fitness so far:0.659839269424858
90.Best fitness so far:0.659839269424858
91.Best fitness so far:0.659839269424858
92.Best fitness so far:0.659839269424858
93.Best fitness so far:0.659839269424858
94.Best fitness so far:0.659839269424858
95.Best fitness so far:0.659839269424858
96.Best fitness so far:0.659839269424858
97.Best fitness so far:0.659839269424858
98.Best fitness so far:0.659839269424858
99.Best fitness so far:0.659839269424858
The best solution contained the following:
Result = 0.659839269424858
```

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı	: ÖRS, Elif Tuğçe
Uyruğu	: T.C.
Doğum tarihi ve yeri	: 03.11.1985, Eskişehir
Medeni hali	: Bekar
Telefon	: 0 (312) 291 72 80
Faks	: 0 (312)
e-mail	: etors@etu.edu.tr

Eğitim

2011	**TOBB Ekonomi ve Teknoloji Üniversitesi, Ankara**
	Yüksek Lisans, Bilgisayar Mühendisliği
2008	**Eskişehir Osmangazi Üniversitesi, Eskişehir**
	Lisans, Bilgisayar Mühendisliği

İş Deneyimi

Yıl	**Yer**	**Görev**
2008-2010	TOBB Ekonomi ve Teknoloji Üniversitesi	Araştırma Görevlisi
2010-	Bilim, Sanayi ve Teknoloji Bakanlığı	San. ve Tek. Uzman Yrd.
2011-	Türk Standardları Enstitüsü	Raportör

Yabancı Dil

İngilizce, Almanca

Yayınlar

E.T. ÖRS, G.D. KURT, A. YAZICI, Kapsayan Ağaç Taraması Yöntemi ile İç Ortamların Taranması, Bilimde Modern Yöntemler Sempozyumu BMYS 2008, 15-17 Ekim 2008, Eskişehir.