T.C.

İSTANBUL AYDIN ÜNİVERSİTESİ

LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

GELİŞTİRİLMİŞ SPEA2 İLE ENVANTER PROBLEMİNİN ÇÖZÜMÜ

YÜKSEK LİSANS TEZİ

Ali BAYRAKDAR

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

NİSAN, 2020

İSTANBUL AYDIN ÜNİVERSİTESİ

LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

GELİŞTİRİLMİŞ SPEA2 İLE ENVANTER PROBLEMİNİN ÇÖZÜMÜ

YÜKSEK LİSANS TEZİ

Ali BAYRAKDAR

(Y1713.010017)

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Tez Danışmanı: Doç. Dr. İlham Hüseyinov

NİSAN, 2020

YEMİN METNİ

Yüksek Lisans / Doktora tezi olarak sunduğum “Geliştirilmiş SPEA2 ile Envanter Optimizasyonu” adlı çalışmanın, tezin proje safhasından sonuçlanmasına kadarki bütün süreçlerde bilimsel ahlak ve geleneklere aykırı düşecek bir yardıma başvurulmaksızın yazıldığını ve yararlandığım eserlerin Bibliyografya’da gösterilenlerden oluştuğunu, bunlara atıf yapılarak yararlanılmış olduğunu belirtir ve onurumla beyan ederim. (27/01/2020)

Aday / İmza

ÖNSÖZ

Temelde optimizasyon problemlerinin çözümünü ele alan bu çalışmanın, envanter problemi dışındaki alanlarda da yararlı olacağını umuyorum. Tez boyunca bana destek olan aileme ve daha iyisini yapmam için beni motive eden sayın hocam Doç. Dr. İlham Hüseyinov'a sonsuz teşekkürlerimi sunarım. Ayrıca Nilgün Haciu-Ayşe Teyze'nin Çantaları şirketine algoritmanın testi için bize güvendiğinden dolayı teşekkür ederim.

Nisan 2020 Ali Bayrakdar

Yazılım Geliştirici

İÇİNDEKİLER

Sayfa

KISALTMALAR

SPEA	**:** Strength Pareto Evolutionary Algorithm
SPEA2	**:** Strength Pareto Evolutionary Algorithm 2
NSGA	**:** Non-Dominated Sorting Genetic Algorithm
NSGA-II	**:** Non-Dominated Sorting Genetic Algorithm-II
NSGA-III	**:** Non-Dominated Sorting Genetic Algorithm-III
OMOPSO	**:** Optimized Multi-Objective Particle Swarm Optimizer
TOPSIS	**:** Technique for Order of Preference by Similarity to Ideal Solution
IBEA	**:** Indicator Based Evolutionary Algorithm
HypE	**:** Hypervolume Estimation Algorithm
MO-CMA-ES	**:** Multi-Objective Covariance Matrix Adaptation Evolution Strategy
MOEA/D	**:** Multi-Objective Evolutionary Algorithm Based on Decomposition
ISIC	**:** International Standart Industrial Classification of All Economic Activities
UNEP	**:** United Nations Environment Programme
VEGA	**:** Vector Evaluated Genetic Algorithm
GHz	**:** Gigahertz
GB	**:** Gigabyte
SSD	**:** Solid State Disk

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

GELİŞTİRİLMİŞ SPEA2 İLE ENVANTER PROBLEMİNİN ÇÖZÜMÜ

ÖZET

Optimizasyon problemleri hemen hemen her ticari ve akademik disiplini ilgilendirmektedir. Optimizasyon probleminin tanımı, belirli kısıtlar altında belirli karar değişkenlerinin değerlerini tayin ederek bir amaç fonksiyonunun optimize etmektir. Bu problemlerin bir türü de envanter optimizasyonudur. Envanter optimizasyonu, herhangi bir envantere sahip olan tüm ticari kuruluşlar için büyük önem taşımaktadır. Öyle ki, envanter optimizasyonu direkt olarak finansal kazanca ve müşteri memnuiyetine etki etmektedir. Bunu yanı sıra envanter optimizasyonu tedarik zincirinin diğer alanlarıyla etkileşim halindedir. Üretim biriminden ham madde temin etmek, ambar yerleri ile sayısını belirlemek, ulaşım türleri, tedarikçi seçimi, genel orta vade üretim planı, promosyon seçimi bu alanların bazılarıdır. Matematiksel olarak modellenmiş olan farklı envanter problemlerinin farklı amaç fonksiyonları, kısıtlamaları ve karar değişkenleri vardır. Giderleri minimize etmek, karı maksimize etmek, envanter yatırımlarında gelen kar dönüş oranını maksimize etmek, bir parametre için tek bir mükemmel sonuç elde etmek bunlardan bazılarıdır. Ayrıca envanter üzerindeki insan kontrol faktörünün optimal değerini bulmak, belirsiz bir gelecekte zarara uğramamak için belirli bir esneklik derecesine sahip olmak, kurum içindeki politik anlaşmazlıkları minimize etmek, bazı yöneticilerin kurum içindeki yerini garanti altına almak gibi amaç fonksiyonlarıda olabilir. Envanter optimizasyonundaki kısıtlamalar ise tedarikçi, pazarlama veya şirket iç politikalarından kaynaklanmaktadır. Maksimum sipariş miktarı, optimum müşteri memnuniyeti, maksimum iş gücü bu kısıtlamara bazı örneklerdir. Karar değişkenleri ile kısıtlamar arasında mutlak bir çizgi yoktur. Bazı envanter modellerindeki kısıtlamalar diğer modellerde karar değişkeni olabilmektedir. Bir ürünün fiyatına veya sipariş miktarına karar vermek, karar değişkeni olarak modellenebilir.

Envater problemleri matematiksel olarak modellenirken amaç sayısı, dönem sayısı, ürün sayısı gibi parametreleri dikkate alınır. Bu tez çalışmasında üç farklı envanter modeli üstünde deneyler yapılmıştır. İlk model çok amaçlı tek dönemli çok ürünlü bir modeldir. İkinci modelin çok amacı tek dönemi ve tek ürünü vardır. Üçüncü model ikinci modelin çok ürünlü türevidir. Bu üç model gerçek hayatta, birden çok bozulabilir ürüne sahip olan süpermarket envanterlerini, bir sezon sonunda modası geçen ürünlere sahip butiklerin envanterlerini ve yeni teknolojilerin gelişmesiyle işe yaramaz hale gelen son teknoloji ürünlerinden oluşan envanterleri kapsamaktadır. Bu sebepten ötürü bu çalışmada modellenen envanterler, gerçek hayatta kullanılan envanterlerin büyük bir çoğunluğunu temsil etmektedir.

Bu modeller farklı veri setlerine göre, farklı parametrelerle, farklı optimizasyon algoritmlarına göre çözülmüşlerdir. Bu problemlerin çözümünde en popüler yöntemler, metasezgisel algoritmalar ailesine ait olan evrimsel algoritmalardır. Bilim dünyasında en güvenilen ve saygı duyulan evrimsel algoritmalar SPEA ve NSGA ailesine ait olan algoritmlardır. Bu sebepten dolayı, bu çalışmada bu ailelere ait olan

algoritmalar kullanılmıştır. Bu algoritmaların performansları belirli performans metrikleri ile karşılaştırılmıştır. Öncelikle ilk envanter modeli, bu ailelerin en yeni versiyonları olan SPEA2 ve NSGA-III algoritmları ile çözülmüştür. Algoritmların bu problem üstünde gösterdiği performans, üç performans metriği ile ölçülmüştür. Karşılaştırma sonucu SPEA2'nin daha iyi bir performans sergilediği gözlemlenmiştir. Bunun üzerine SPEA2'nin geliştirilip envanter optimizasyonunun çözümünde kullanılmasına karar verilmiştir. NSGA-II algoritması da yeni SPEA2 ile kıyaslanması amacıyla geliştirilip daha iyi hale getirilmiştir. İkinci ve üçüncü envanter modelleri yeni SPEA2, yeni NSGA-II, SPEA2, NSGA-II ve bir sürü algoritması olan OMOPSO ile çözülmüştür. Çözüm işlemleri farklı iterasyon ve popülasyon sayıları için 10'ar kez gerçekleştirilmiştir.Bu üç problemin yanı sıra algoritmalar üç farklı test problemi üzerinde denemiştir. Algoritmların gösterdiği performansın ölçülmesi için aynı performans metrikleri kullaılmıştır. Karşılaştırma sonuncunda geliştirilmiş SPEA2 algoritmasının her problemin her çözümü için diğer algoritmalarda daha iyi performans sergilediği gözlenmiştir. Geliştirilmiş NSGA-II ise geliştirilmiş SPEA2 algoritmasından sonra en iyi performansı sergilemiştir. Bu deney sonucunda geliştirilmiş SPEA2 algoritmasının çok amaçlı envanter optimizasyonu için en uygun çözüm olduğu görülmüştür.

En sonunda, geliştirilmiş olan SPEA2 algoritması gerçek bir problem üzerinde denenmiştir. Bu problem çok amaçlı tek dönemli çok ürünlü bir modeldir. Bu envantere sahip olan firma herhangi bir optimizasyon yöntemi kullanmamaktadır. Optimizasyon firmanın bir satış dönemi için yapılmıştır. Dönem başında optimizasyon yapılmadan elde edilecek karar değişkeni değerleri kaydedilmiştir ama uygulamaya konmamıştır. Optimizasyon yapılarak bulunan karar değişkenleri gerçek hayatta uygulamaya konulmuştur. Dönem sonunda optimize edilen envanterin karı ile envanter optimize edilmemiş olsaydı elde edilecek olan kar karşılaştırılmıştır. Bu karşılaştırma sonucunda gözlenen, geliştirilmiş SPEA2 algoritması ile yapılan optimizasyonun, optimizasyon yapılmamasından daha karlı olduğudur. Elde edilen bu kar oranı günümüz ticari şartları bakımından büyük bir orandır.

Anahtar Kelimeler : *Envanter optimizasyonu, SPEA2, NSGA-II, NSGA-III.*

INVENTORY OPTIMIZATION WITH A NOVEL SPEA2 ALGORITHM

ABSTRACT

Optimization problem holds importance for almost all scientific and trade disciplines. By definition, optimization problem is maximizing or minimizing an objective function under constraints with certain decision variables. Optimization problems may or may not have constraints and they may have one or more objective functions. Optimization problems with more than one objective function are known as a multi-objective optimization problems. Obtimization problems with more than or equal to three objectives are known as many-objective optimization problems. While it is possible for a single objective problem to have a single best answer, solving a multi-objective function is a more complicated task. Multi-objective optimization problems generally have more than one correct solution for each decision varibale. Therefore, Pareto optimization concept is used for deciding the correct solutions for the optimization problem.

A Pareto solution is a solution which cannot be further improved for an objective without detoriating another one. Typically, while solving a multi-objective problem the algorithm yields numerous Pareto solutions. Then the decision maker decides which solution or solutions are going to be used. Statistical methods like TOPSIS are common techniques used by the decision maker to order the Pareto solutions. The algorithm which is used to solve the problem generates the values for the decision variables and the objective functions. The values yielded for the decision variables are known as the Pareto optimal set. The values yielded for the objective functions are known as the Pareto front. Multi-objective inventory problems are no exception to the structure described above. They have multiple objective functions, decision variables and constraints. When they are solved, multiple Pareto solutions are yielded.

Inventory problems hold great significance to inventory managers because they are directly tied to financial gains and customer satisfaction. Moreover the inventory problem often interacts with other areas of operational research. Investing in raw materials, production rate, service and maintenance activities, warehouse specifications, trasportation, pricing and choosing the suppliers are among these areas. The objective functions for inventory problems include but are not limited to are maximizing the profit, minimizing losses, maximizing service rate and customer satisfaction. The constraints may depend on the supplier, internal politics, or marketing limitations. Limits to order sizes are supplier constraints. Tolerable service level is the most important marketing constraint. Budget and workload constraints are some of the most important internal constraints.

While designing a model for the inventory problem at hand, there are some classifications to be considered. The inventory model may have single or multiple objectives. Products may be perishable or their shelf-life may be unlimited. There may be multiple products or a single product. The model may have a single period or it may

span multiple periods. In a multiple period model, the items in stock may carry over to the next period or they may become obsolote at the end of a single period. The demand for products may be deterministic or stochastic. There may be single or multiple warehouse locations. The order cost may or may not depend on the size of the order and so on.

In this study, three different inventory models are designed. The first model is a multi-objective single-period multi-item inventory model. Its objectives are maximizing the profit and warehouse occupancy level. Warehouse lavel and ordering budget are constraints. Order size is the decision variable. The objectives for the second model are maximizing the profit and service level. Order amount and selling price are the decision variables. There are constraints on the order amount and the selling price. The third model is identical to the second model except it has multiple items. These three models correspond to retail stores with multiple perishable items or garment stores with items which goes out of style at the end of each period. It can also cover some specific high technology items which can become obsolote with the emerging of new technologies. Therefore, the models presented in this paper cover most of the inventory models encountered in real life.

Inventory problems and other optimization problems are traditionally solved with metaheuristic optimization algorithms. They can also be solved with exact algorithms or specific heuristics but metaheuristics, especially evolutionary algorithms are known to perform better.

Metaheuristic algorithms can be classified into solution based or population based algorithms. Solution based algorithms include local search and simulated annealing methods. Local search algorithms may have a hard time finding the globally optimum solutions while simulated annealing methods may not yield a diverse set of solutions. Simulated annealing also has an asymptotic time complexity Therefore as the number of objectives and decision variables increase, time complexity increases exponentially. Swarm based algorithms and tabu search algorithms are the two other main subdivisions of population based algorithms other than evolutionary algorithms. Tabu search displays poorer performance as the number of objectives increases and has a hard time searching continuous search spaces. It also has a higher time and space complaxity compared to other methods. This makes it a poor candidate for hybridization with other algoirthms. Swarm based algorithms do not allow for much parameter manipulation. They also need to run multiple times for yielding a good solution and even then they may not reach the true Pareto front. These reasons has lead to evolutionary algorithms to be preferred over other optimization algorithms.

There are three kinds of evolutionary algorithms: Pareto based, indicator based and decomposition based algorithms. Indicator based algorithms search towards the parts of the solution plane with the guidance of a performance indicator based metric. Their most important disadvantage is their fastly growing time complexity as the number of the objective functions increases. Since most of the metrics use a reference set to compare the performance of the algorithm, indicator based algoirhtms may perform poorly according to the choice of the reference points. IBEA, HypE, MO-CMA-ES are some of the indicator based algorithms. Decomposition based algorithms require a priori based knowledge. Since they use aggregation techniques, the number of weights used increase exponentially. NSGA-III and MOEA/D are some of the decomposition based algorithms. Pareto based algorithms are very practical beacuse they require only a few paramteres and can work with large number of objectives without any problems.

The main working mechanism of all the Pareto based algorithms are similar. They use a two step selction mechanism. First, the solutions are chosen according to their Pareto dominance and then they are selected based on the density of solutions on the search plane. NSGA-II and SPEA2 are regarded as the two benchmark Pareto based algorithms. Therefore, in this study algorithms belonging to SPEA and NSGA family are researched.

SPEA2 algorithm has a strength based selection mechanism A solution's strength is the number of other solutions it dominates. The fitness of a solution is calculated with regards to the number of solutions it dominates and the strength of the solutions it is dominated by. Also SPEA2 keeps an archive of nondominated solutions for carrying them on to the next generation. The archive size is fixed. If the number of nondominated solutions exceed the archive limit, a truncation operator eliminates some of the solutions. If there are vacant spots in the archive, then they are filled with the best dominated solutions. Moreover, SPEA2 uses a spreading preservation mechanism which eliminates solutions form the most crowded parts of the solution space. This causes the algorithm to have a better spread value.

The main distinct feature of NSGA-II is its nondominated sorting mechanism. After the evaluation of solutions with regards to the objective functions, the algotihm sorts the solutions according to their fitness values. Then the solutions are sorted into layers according to their fitness values. The nondominated solutions form the first layer, the best group of dominated solutions form the second layer, the second best group of nondominated solutions form the third layer and so on. Selection is made with bias towards the better layers. The selection mechanism is based on binary tournament selection. As in other evolutionary algorithms, NSGA-II has recombination and mutation operators.

The performance of multi-objective evolutionary algorithms are measured using specialized metrics. These metrics usually work on the basis of the convergence or diversity of the obtained solutions. For a better performance assesment, metrics which measure the diversity and the convergence of the solutions should be used together. The metrics can further be classified as unary and binary metrics. Unary metrics are measured indepenedently of the other algorithm being compared. Binary metric compares two or more algorithm based on their performance for solving a single problem. Using one unary metric for each objective function is enough for a good performance estimation. Hypervolume, generational distance and spacing are used in this study. Hypervolume works by calculating the size of the solution space dominated by the obtained solutions. It measures both the convergence and the diversity of the solutions. Generational distance measures the Euclidian distance between solutions and therefore it is a convergence metric. Spacing measures the spread of solutions across the solution space. It is a diversity metric. All three of these metrics ar unary metrics.

In the first part of the study a multi-objective singe-period multi-item objective function is modeled. Then the model is solved with SPEA2 and NSGA-III for 10000 iterations for 10 consecutive times. The two algorithms are compared according to their hypervolume, generational distance and spacing values.The results display that SPEA2 perfoms better than NSGA-III. Therefore, the second half of the study is focused on SPEA2.

In the second part of the study, SPEA2 algorithm is improved with modifications made on its truncation, selection and reproduction schemes. Similar improvements on NSGA-II have been made for comparison purposes. The improved algorithms along with SPEA2, NSGA-II and OMOPSO which is a swarm based algorithm are compared. The comparative analysis is made based upon the performance they display on three test problems and two inventory models. The test problems used are Fonseca, Binh and WFG problems. The first inventory model is a multi-objecive single-period single-item inventory model with profit and service level maximization as objectives. There are constraints on both of these objectives and the decision variables are order amount and selling price. The second model is the same as the first model but the latter has multiple items. The problems where solved with the novel SPEA2, the novel NSGA-II, SPEA2, NSGA-II and OMOPSO algorithms. The problems were solved for 1000, 5000, 10000 population sizes for 10 times with 1000 iterations. Hypervolume, generational distance and spacing metrics are measured for each algorithm. The results display that the novel SPEA2 outperforms the other four algorithms in every problem instance. Also, the novel NSGA-II perfoms worse than the novel SPEA2 but performs better than the other three algorithms.

In the last part of the study, the novel SPEA2 algorithm, with its proven success, is applied to a real world inventory optimization problem. It is a multi-objective single-period multi-item inventory problem. The problem has two objectives, one constraint and one decision variable. The inventory managers aim to find the optimal order amount without exceeding the warehouse capacity for maximizing the profit and the service rate. The optimization algortithm is applied for one period. At the beginning of the period the order amount is decided with the help of an improved SPEA2 based application. The order amount which would have been made without the help of the optimization application is also recorded. At the end of the period the real profit and the would have been profit are compared. It is observed that the optimized profit is greater than the unoptimized profit by a large margin.

Keywords : *Inventory optimization, SPEA2, NSGA-II, NSGA-II*

1. GİRİŞ

1.1 Çalışma Konusu

Bu tez çalışmasında bir optimizasyon problemi olan envanter optimizasyonu için en uygun çözüm yönteminin bulunması ve geliştirilmesi konusu işlenmiştir. Optimizasyon probleminin bir veya birden çok amaç fonksiyonu olabilir. Bir amaç fonksiyonu olan optimizasyon problemine tek amaçlı optimizasyon problemi denir. Birden çok amaç fonksiyonu olan optimizasyon problemine çok amaçlı optimizasyon problemi denir (Coello, 2007). Tek amaçlı optimizasyon probleminin tanımı Tanım 1'de verilmiştir:

Tanım 1 : Tek amaçlı optimizasyon probleminde *f(x)* maksimize veya minimize edilir. Bu fonksiyon $gi(x)\leq 0,\ i=\{1,...,m\}$ ve $hj(x)=0,\ j=\{1,...,p\}$ kısıtşamasına tabiidir. $x\in\Omega$ olmak üzere $x,\ \Omega$ uzayından n-boyutlu bir karar değişkeni vektörüdür. Bu vektörler ayrık veya sürekli olabilir (Coello, 2007).

Çok amaçlı optimizasyon probleminde hedef, karar değişkenleri vektörünü bularak ve kısıtlamaları tatmin ederek bileşenleri amaç fonksiyonlarını olan bir fonksiyon vektörünü optimize etmektir (Coello, 2007). Çok amaçlı optimizasyon probleminin tanımı Tanım 2'de verilmiştir.

Tanım 2 : $x=[x1,x2...,xn]T$ karar değişkenleri ve $gi(x)\leq 0,\ i=\{1,...,m\}$ veya $hj(x)=0,\ j=\{1,...,p\}$ kısıtlamalarına için amaç fonksiyonları vektörü $f(x)=[f1(x),f2[x],...,fk(x)]$ bulunmaktadır. Rn n-boyutlu Öklid uzayıdır. $F:\Omega\rightarrow Y$ işlemi ile *f(x)* vektörünü minimize veya maksimize etmek çok amaçlı optimizasyon problemidir (Coello, 2007). Çok amaçlı optimizasyon problemleri çözülürüken genelde birden fazla optimum çözüm elde edilir. Bunlara Pareto çözümleri denir. Bu çözümleri anlayabilmek için Pareto terminolojisinin bilinmesi gereklidir. Pareto çözümü, bir veya birden çok amaç fonksiyonunu değerini kötüleştirmeden başka bir amaç fonksiyonunun değerinin iyileştirilemeyeceği sonuçlardır. Terminolojide Pareto çözümlerinin diğer çözümleri domine ettiği söylenir (Coello, 2007). Pareto çözümünün tanımı Tanım 3'te Pareto dominansının tanım Tanım 4'te verilmiştir.

Tanım 3 : *iff* $x' \in \Omega$ için $f(x')=(f1(x'),...,fk(x'))$ vektörünü domine eden bir $f(x)=(f1(x),f2[x],...,fk(x))$ vektörü yoksa x' bir Pareto çözümüdür (Coello, 2007).

Tanım 4 : İki vektör olan vi ve ui için eğer $\forall i \in \{1,...,k\}\ ui \leq vi \wedge \exists i \in \{1,...,k\}: ui < vi$ ise v vektörü u vektörünü domine etmektedir (Coello, 2007).

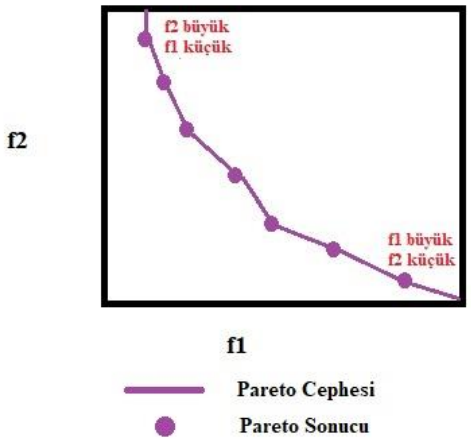
Amaç uzayında Pareto çözümlerini veren karar değişkeni değerlerine Pareto optimal kümesi denir (Coello, 2007). Pareto optimal kümesinin tanımı Tanım 5'te verilmiştir.

Tanım 5 : $P^*:=\{ x \in \Omega | \neg \exists x' \in \Omega\ f(x') \succ f(x)\}$ ise P^* Pareto optimal kümesidir. "$\succ$" Pareto dominansı sembolüdür (Coello, 2007).

Pareto optimal kümesinin Amaç fonksiyonuna uygulamasıyla elde edilen çözüm kümesine Pareto cephesi denir. Pareto cephesinin tanımı Tanım 6'da verilmiştir.

Tanım 6 : $PF^*:=\{u=f(x)|x \in P^*\}$ ise PF^* Pareto cephesidir (Coello, 2007).

Şekil 1.1'de iki amaç fonksiyonu olan bir optimizasyon problemi için iki farklı algoritma tarafından elde edilen sonuçlar ve Pareto cephesi görülmektedir.



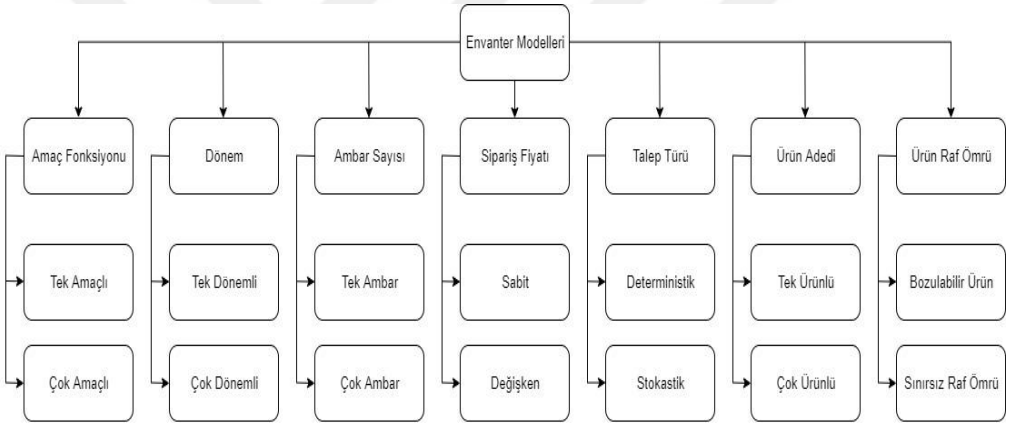
Şekil 1.1 : Pareto Cephesi

Bu çalışmanın konusu bir optimizasyon problemi olan envanter problemidir. Envanter probleminde diğer optimizasyon problemlerinde olduğu gibi amaç fonksiyonları, karar değişkenleri ve kısıtlamalar bulunur. Envanter problemi tek veya çok amaçlı olabilir. Bu çalışmadaki envanter modelleri çok amaçlıdır ve bu sebepten dolayı Pareto terminolojisine tabiidir. Karı maksimize etmek, zararı minimize etmek, servis seviyesini maksimize etmek gibi amaç fonksiyonları olabilir. Bu fonksiyonlar çözülürken karar değişkenleri toptancıya yapılan sipariş miktarı, ürünün satış fiyatı gibi değişkenler arasından seçilebilir. Depo hacmi, bütçe, sipariş miktarı gibi kısıtlamalar olabilir. Bir modelde kısıtlama olan bir parametre başka bir modelde karar

değişkeni olabilir (Silver, 2008). Envanter probleminin kara kutu olarak temsili **Şekil 1.2**'de verilmiştir.

Şekil 1.2 : Envanter Problemi

Envanter problemi modellenirken ürün sayısı, ürünlerin raf ömrü, talep türü, dönem sayısı gibi parametreler göz önüne alınır (Silver, 2008). Envanter probleminin modellenmesi için gereken parametreler **Şekil 1.3**'de görülebilir.



Şekil 1.3 : Envanter Modelleri

Envanter problemi tek amaçlı veya birden çok amaçlı olabilir. Tek amaçlı envanter modelleri tek amaçlı optimizasyon problemleri, çok amaçlı envanter modelleri çok amaçlı optimizasyon problemleridir. Envanter tek dönemlik planlanabilir veya önceki dönemden kalan ürünlerin devrettiği birden çok dönem olabilir. Birden çok dönem olması halinde envanterin gözden geçirilmesine ihtiyaç duyulur. Envanter periyodik olarak veya bir ürünün stok miktarı belirli bir seviyeye düştüğünde gözden geçirilebilir. Envanter bir veya birden çok ambarda saklanabilir. Toptancıdan sipariş fiyatı sabit olabilir veya sipariş miktarı ve sipariş zamanı gibi değişkenlere bağlı olarak farklılık gösterebilir. Ürünlere gösterilen talep deterministik, yani her zaman aynı olabilir. Talep stokastik de olabilir. Bu durumda modeli oluşturmak için dağılım türüne göre dağılım yoğunluk fonksiyonları modele dahil olabilir. Stokastik bir modelde dağılım türü de dönem içinde veya dönemler arasında farklılık gösterebilir. Bir

envanterde sadece bir çeşit ürün veya birden çok çeşit ürün bulunabilir. Envaterde bulunan ürünler sınırsız raf ömrüne sahip olabilir veya son kullanma tarihi bulunabilir. Son kullanma tarihi olan ürünler dönem sonunda veya dönemin içinde bozulabilir. Bozulan ürünler daha az bir fiyata satılabilir, toptancıya iade edilebilir veya hiç satılmadan imha edilebilir (Silver, 2008). Bu çalışmada üç tür çok amaçlı tek dönemli envanter modellenmiştir. Birincisi ve üçüncüsü çok ürünlü, ikincisi tek ürünlüdür. Talebin dönem boyunca değişmeden tek biçimli sürekli dağılımı olduğu kabul edilmiştir. Ürünler dönem sonunda atılamaz ve iade edilemez hale gelmektedirler. Tek bir ambar olduğu kabul edilmiştir.

Bu çalışmada optimizasyon problemi çözümü için çok amaçlı evrimsel algoritmalar kullanılacaktır. Optimizasyon konusunda bilim camiasının güvenini ve saygısını kazanan evrimsel algoritmalar, genelde aynı prensibe göre çalışır. Evrimsel algoritmalar çalışmaya başladığını ilk önce rassal olarak bir sonuçlar kümesi oluşturur. Bu kümeye popülasyon denir. Popülasyonu oluşturan çözümlere bireyler adı verilir. Belirli bir iterasyon sayısı kadar veya önceden belirlenen bir şart yerine getirilene kadar bir döngü yapısıyla bireyler bazı işlemlerle modifiye edilir. Öncelikle bireylerin amaç fonksiyonuna uygunluğu ve birbirilerine olan üstünlükleri hesaplanarak en iyi bireyler bir sonraki popülasyona aktarılmak üzere seçilir. Buna seleksiyon denir. Bu seçilen bireylerin bazı özellikleri birbirleri arasında takas edilir. Buna üreme veya rekombinasyon denir. Daha sonra, üremiş olan bireylerin bazı özellikleri rassal veya belirli bir kurala göre değiştirilir. Buna mutasyon denir. Bu işlem sonucunda yeni nesil elde edilmiş olur ve tekrar döngünün başına dönülür (Whitley, 1994). Farklı evrimsel algoritmaların farklı stratejileri vardır. Örneğin bazı algoritmalar çözüm kümesinin çeşitliliğini korumak için Pareto çözümlerini bir arşivde saklayarak yeni popülasyona katılmalarını garanti altına alır (Emerich, 2018). Evrimsel algoritmaların ortak pseudokodu **Algoritma 1**'de, temel yapısı **Şekil 1.4**'de görülmektedir.

Algoritma 1:

t=0;
initialize P(0):={a1(0),...,an(0)}∈In(0);
while(ı({P(0),...,P(t)})≠true) do
    rekombinasyon uygula: P'(t):=r(t)(P(t));
    mutasyon uygula: P''(t):=m(t)(P'(t));
    seleksiyon uygula:
    if X

then P(t+1)=s(t)(P’’(t);

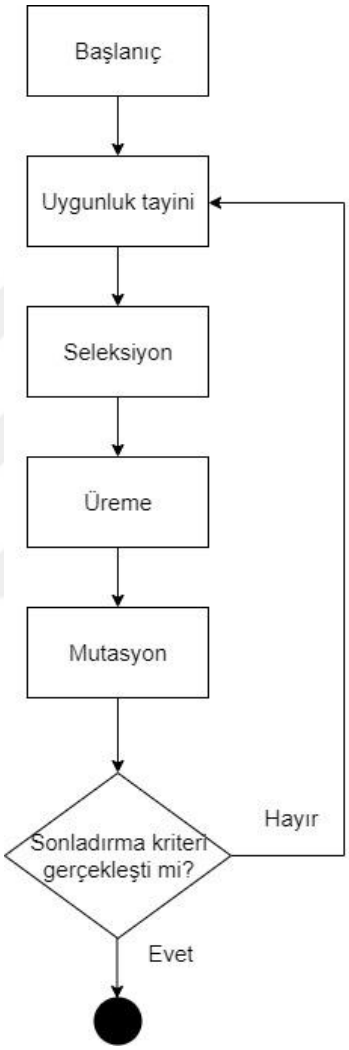
else P(t+1)=s(t)(P’’(t)∪P(t));

end if

t=t+1;

end while

return P;



Şekil 1.4 : Çok Amaçlı Evrimsel Algoritmaların Temel Yapısı

Çok amaçlı evrimsel algoritmaların ortak matematiksel tanımı **Tanım 7**’de verilmiştir. **Tanım 7**: *I* boş olmayan bireyler uzayıdır. *Z+* ebeveynler kümesinin boyutudur. Aynı zamanda *Z+* çocuklar kümesinin boyutudur. *{μ’(i)}i∈N Z+*’daki herhangi bir sıralı elemandır. *Φ:I→R* uygunluk fonksiyonudur. $\iota:\bigcup_1^\infty I\mu(i)\rightarrow$ {true, false} (sonlandırma şartı), *X∈{true,false},r(i) :Xr(i)→T(Ωr(i),T(Iμ(i), I’μ(i)))* rekombinasyon operatörü olmak üzere *r*, *{r(i)}*’nin bir sıralı elemanıdır. *m(i) :Xm(i)→T(Ωm(i),T(Iμ(i), I’μ(i)))*

mutasyon operatörü olmak üzere *m, {m(i)}*'nin bir sıralı elemanıdır. $s(i):Xs(i)\times T(I,R)\rightarrow T(\Omega s(i),T(I'\mu(i)+x\mu(i), I\mu(i+1))$ seleksiyon operatörü olmak üzere *s, {s(i)}*'nin bir sıralı elemanıdır. $\Theta r(i) \in Xr(i)$ rekombinasyon parametreleri, $\Theta m(i) \in Xm(i)$ mutasyon parametreleri ve $\Theta s(i) \in Xs(i)$ seleksiyon parametreleridir (Coello, 2007).

Evrimsel algoritmaların optimizasyon amaçlı kullanılması 1984 yılında VEGA'nın bulunmasıyla başlar. VEGA'nın uygunluk fonksiyonu sonuç olarak bir vektör döndürmektedir. Çalışma prensibi olarak popülasyonu alt popülasyonlara böler ve her alt popülasyon vektörün bir bölümünü iyileştirmekten sorumluluğu altına alınır (Schaffer, 1986). VEGA'nın Pareto cephesinin içbükey bölümlerini bulmakta başarısız olmasından dolayı çeşitli stratejileri olan çeşitli evrimsel algoritmalar geliştirilmiştir. Sonuçların çeşitliliğini koruyan seyreltme operatörleri olan, paralel popülasyonlar ve algoritmanın karar vericiyle etkileşim içinde bulunması bu stratejilerden bazılarıdır. Ama SPEA2 ve NSGA-II'ninde bulunduğu Pareto bazlı algoritmalar diğerlerinden daha popüler olmuşlardır. Bu algoritmaların temelinde sonuçların diğer sonuçlara olan Pareto dominansları göz önüne alınır (Emerich, 2018). Bu çalışmaya konu olan evrimsel algoritma strength Pareto evolutionary algorithm (SPEA2)'dir. SPEA2'de uygunluk puanları verilirken bir bireyin hangi bireyler tarafında domine edildiği ve hangi bireyleri domine ettiği göz önüne alınır. Ayrıca domine edilmeyen sonuçlar ayrı bir arşivde tutulur. Mutasyon ve rekombinasyon işlemleri standarttır (Zitzler, 2001). SPEA2'nin temel yapısı şöyledir:

Popülasyonun birey sayısı Pmax, arşiv boyutu Pmax* ve kuşak sayısı K önceden belirlenmiştir.

Adım 1 : Başlangıç:

P0 popülasyonu ve P0* arşivi oluşturulur.

Adım 2 : Uygunluk hesaplanır:

Hem P0'de hem P0*'da bulunan bireylerin uygunlukları domine ettikleri birey sayısı temel alınarak hesaplanır.

Adım 3 : İlk Seleksiyon:

Hem P0'de hem P0*'da bulunan dominant bireyler P1* arşivine taşınır. Eğer taşınacak birey sayısı Pmax*'e eşitse bir sonraki adıma geçilir. Eğer taşınacak birey sayısı fazlaysa, budama işlemleri ile fazla bireyler silinir. Eğer taşınacak birey sayısı azsa, bu bireylere ek olarak arşive domine edilmiş sonuçlar de eklenir.

Adım 4 : Son:

Eğer kuşak sayısı K'ye ulaşmışsa, arşivdeki sonuçlar cevap olarak verilir ve Algoritma sonlanır.

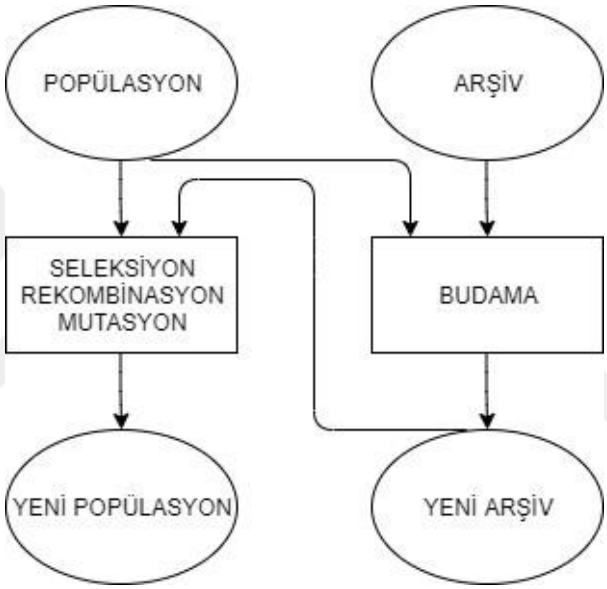
Adım 5 : İkinci Seleksiyon:

Her birey farklı iki bireyle bastırdıkları birey sayısı göz önüne alınark karşılaştırılır. Yeni bir P oluşturulur.

Adım 6 :

Adım 5'te oluşturulan P'ye mutasyon ve rekombinasyon uygulasnır. Elde edilen küme yeni popülasyonu oluşturur ve Adım1'geri dönülür.

SPEA2'nin ana yapısının grafiksel gösterimi **Şekil 1.5**'te görülmektedir. Ayrıca pseudocode'u **Algoritma 2**'de sunulmuştur.



Şekil 1.5 : SPEA2'nin Temel Yapısı

Algoritma 2:

```
Procedure SPEA2
Input: Y;
Initialize Z0;
Set W0=Ø,T= Ø, u=0, nds=0;
While u<Y
        Zu ve Wu için uygunluğu hesapla;
        nds= Zu ∪ Wu'dan gelen domine edilmeyen çözümler;
        if nds>T then
                Budama operatörünü çalıştır;
        Else if nds<T
                Wu 'daki boş yerleri en iyi domine edilen çözümlerle doldur;
        Else
                Wu'yu domine edilmeyen sonuçlarla doldur;
        End if
```

Rekombinasyon ve mutasyon gerçekleştir;
Set u=u+1;
End while;
Return wy

Çalışmada kullanılacak olan diğer algoritma ise NSGA-II algoritmasıdır. NSGA-II popülasyonda domine edilmeyen sonuçları seçip katmanlara ayırarak çalışmaya başlar. Daha sonra algoritma, seleksiyon işlemini daha uygun katmanlara pozitif ayrımcılık yapacak şekilde gerçekleştirir. Mutasyon ve rekombinasyon işlemleri standart bir biçimde yapılır (Deb, 2000). NSGA-II algoritmasının temel yapısı şöyledir:

Adım1 : Ebeveynler, çocuklar ve referans noktaları oluşturulur:

P0 adında rassal bir ebeveyn popülasyonu oluşturulur. Amaç fonksiyonu sayısına göre ZS referans noktaları oluşturulur. P0'daki bireylerin uygunluğu baskınlık derecesine göre seçilir. Herbir birey iki kere olacak şekilde başka bir bireyle karşılaştırılır. Kazananların bazı özellikleri mutasyon ile değiştirilir ve Q0 çocuk kümesini oluştururlar. Bu kümenin eleman sayısı S'dir.

Adım2 : Ebeveyn ve çocuk kümesi birleştirilir:

Ebeveynler ve çocuklar birleştirilerek eleman sayısı 2S olan r kümesi oluşturulur.

Adım3 : Non-dominated sort ile r sınıflandırılır:

Non-dominated sort yani baskılandırılmamaış sıralama ile rt kümesi farklı cephelere ayrılır. En az bastırılan cephe en ayrıcalıklı olan cephedir. Bu şekilde en ayrıcalıklı cepheler, toplamı S'i geçmeyecek şekilde seçilir ve yeni popülasyon P1 kümesini oluşturur.

Adım4 : Crowding distance yani kalabalık mesafesi ile P1 kümesine bir seçilmiş cephe eklenir:

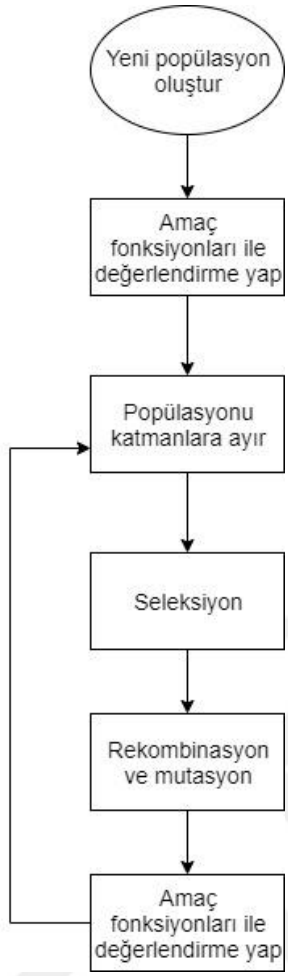
Her bireyin en yakın komşu iki bireye olan uzaklıklarının ortalaması bulunarak kalabalık mesafesi hesaplanır. Yüksek kalabalık uzaklığına sahip bireylerden oluşan cephe, P1 kümesine dahil edilir.

Adım5 : P1, ZS referanslarına göre tekrar düzenlenir:

P1'den toplam sayısı S olacak kadar, ZS referans noktalarına en yakın bireyler seçilir.

Adım6 : Adım5'te seçilen bireyler yeni P1 kümesini oluşturur.

NSGA-II'nin ana yapısının grafiksel gösterimi **Şekil 1.6**'te görülmektedir. Ayrıca pseudokod'u **Algoritma 3**'de sunulmuştur.



Şekil 1.6 : NSGA-II'nin Temel Yapısı

Algoritma 3:

Procedure: NSGA-II

Input: Y

Initialize Z0

Set W0=Ø, e=0, u=0

While e<Y

Set u=0

While u<length Z0

We∪Ze kümesine seleksiyon uygula

Rekombinasyon ve mutasyon uygula

Yeni bireyleri değerlendir

Set u=u+1

End while

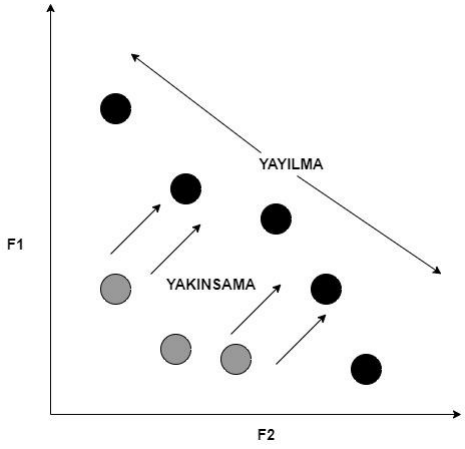
Ebeveynleri ve çocukları sırala ve katmanlara ayır

Set z=z+1

End while

Return Ze

Evrimsel algoritmaları performansları hesaplanırken özelleştirilmiş performans metrikleri kullanılır. Performans metriklerinin amacı bulıunan sonuçların gerçek Pareto cephesine yakınsaklığı ve çözüm uzayına ne kadar iyi yayıldığını ölçmektir (Laumanns, 2002). Bazı performans metrikler hem yakınsaklığı hem yayılmayı ölçer. Ama genelde her metrik ya sadece yakınsaklığı ya da sadece yayılmayı ölçer. Yayılma ve yakınsama grafiksel olarak Şekil 1.7'de gösterilmiştir. Bilim dünyasında birçok perfromans metriği kullanılmaktadır ve hangisnin daha iyi olduğuna dair bir konsensus yoktur. Fakat bunlardan bazılarına diğerlerinden daha çok güvenilmektedir. Ayrıca yakınsamayı ve yayılmayı ölçen metriklerin beraber kullanılması daha doğru karar verilmesini sağlar (Laumanns, 2002). Bunun yanı sıra performans metrikleri tekli veya çoklu olarak sınıflandırılabilir. Tekli metrikler her algoritma için birbirinden bağımsız teker teker sayısal değerler verir. Çoklu metrikler ise algoritmaları birbiriyle kıyaslayarak sayısal değerler denir (Zitzler, 2003). Bilim dünyasında kabul gören kural bir problemdeki amaç fonksiyonu kadar tekli metrik kullanmak iyi bir performans analizi için yeterlidir (Zitzler, 2002). Bu çalışmada üst hacim, kuşaksal mesafe ve boşluk metrikleri kullanılmıştır.



Şekil 1.7 : Yayılma ve Yakınsama

Üst hacim hem yakınsama hem de yayılma hakkında bilgi veren bir performans metriğidir. Ayrıca tekli bir metriktir. Üst hacmi hesaplarken bulunan her sonuç için bir küp oluşturulur. Küp oluşturulurken en kötü amaç fonksiyonu değerlerini barındıran vektör, referans noktası seçilir. Her sonuç için bu küp hacimleri toplanarak üst hacim

metriği bulunur. Üst hacmin hesaplanması için **(1.1)** eşitliği kullanılır. Daha yüksek performans için daha büyük üst hacim değeri istenmektedir (Zitzler, 1998).

$$UH = hacim|\bigcup_{i=0}^{|Q|} Vi \tag{1.1}$$

Kuşaksal mesafeyi bulmak için, Pareto cephesine dahil olan veya olmayan her sonucun Pareto cephesine olan ortalama mesafesi bulunur.

$$KM = \frac{(\sum_{i=1}^{|Q|} öi^p)}{|Q|}^{1/p} \tag{1.2}$$

(1.2) eşitliği ile kuşaksal mesafe *KM* hesaplanır. Q hesaplanacak olan sonucu temsil eder. Sonuç ile cephe arasındaki Öklid mesafesi olan *ö*'yü bulmak için ise **(1.3)** eşitliği kullanılılır. Daha küçük kuşaksal mesafe değerine sahip olan algoritmanın daha iyi performans gösterdiği kabul edilir.Kuşaksal mesafe tekli bir yakınsama metriğidir (Van Veldhuizen, 1999).

$$ö_i = \min_{k \in |p^*|} \sqrt{\sum_{m=1}^{M} (fm^i - fm^{*i})^2} \tag{1.3}$$

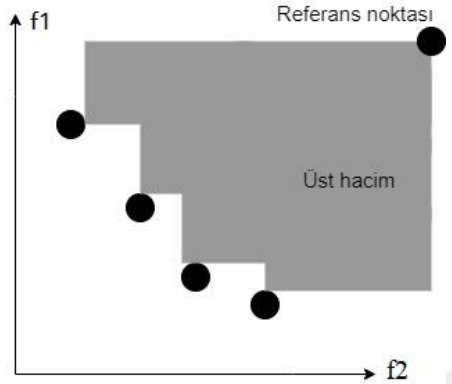
Boşluk metriği, Pareto cephesindeki çözümlerin birbirilerine olan uzaklıkları ile ilgilidir. Pareto cephesindeki ardışık sonuçların birbirilerine olan ilişkisel mesafelerinin ölçümleri ile bulunur. **(1.4)** eşitliği ile boşluk hesaplanır. Buradaki Öklid mesafesini hesaplamak için, kuşaksal mesafeden farklı olarak minimum değer kullanılır. Ayrıca ortalama hesplanarak uç değerlerin etkisi azaltılmıştır. Bu işlemler **(1.5)** ve **(1.6)** formülleri ile gerçekleşir. Algoritmanın performansının yüksek olması için boşluk değerinin olabildiğince küçük olması istenir. Boşluk tekli bir yayılma metriğidir (Schott, 1995).

$$B = \sqrt{\sum_{i=1}^{Q} (di - \overline{d})^2 \frac{1}{|Q-1|}} \tag{1.4}$$

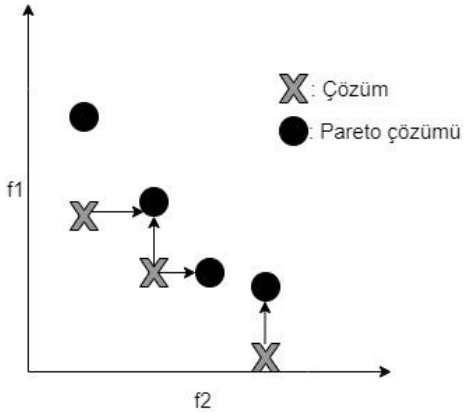
$$di = \min_{k \in Q \cap k \neq i} \{\sum_{m=1}^{M} (f_m^i - f_m^k)\} \tag{1.5}$$

$$\overline{d} = \sum_{i=1}^{|Q|} di/|Q| \quad \textbf{(1.6)}$$

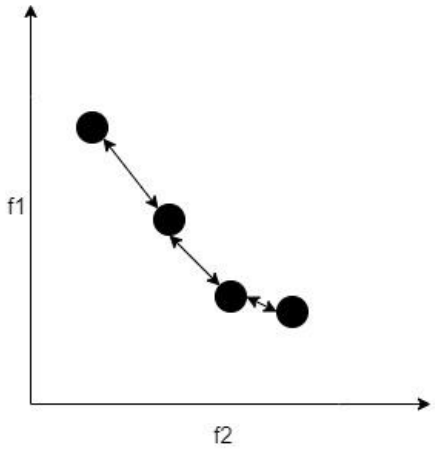
Bu üç metriğin grafiksel gösterimi **Şekil 1.8**, **Şekil 1.9** ve **Şekil 1.10**'da sunulmuştur.



Şekil 1.8 : Üst Hacim



Şekil 1.9 : Kuşaksal Mesafe

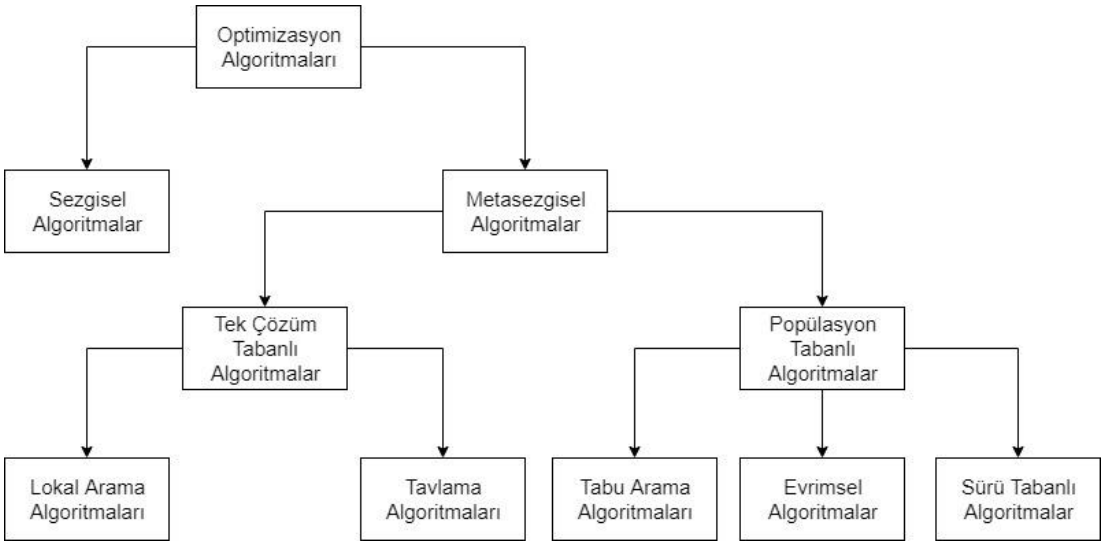


Şekil 1.10 : Boşluk

1.2 Tezin Amacı

Envanter problemi sadece envanter yönetcisini değil bütün tedarik zinciri halkalarını ilgilendirmektedir. Öyle ki, envanter yönetiminde yaşanan stoksuzluk veya ambar alanından taşma gibi sorunlar işletmenin bütün işleyişini olumsuz yönde etkileyebilir veya tamamen durdurabilir. Diğer taraftan iyi yönetilen bir envater sadece tedraik zincirinin sorunsuz işlemesine katkı sağlamak dışında finansal kazançlar ve müşteri memnuniyeti gibi uzun vadeli kazançlara sebep olur. Bu çalışmada incelenen çok amaçlı tek dönemli envanter modelleri gerçek hayatta bozulabilir ürünlere sahip marketlere, modası geçince satılamaz hale gelen ürünler satan moda dükkanlarına ve zamanın en son teknolojisini satmak zorunda olan yüksek teknoloji ürünleri satan işletmelere karşılık gelir. ISIC (International Standard Industrial Classification of All Economic Activities) bu sayılan alanları Grup 6 aktiviteleri olarak sınıflandırmaktadır. UNEP (United Nations Environment Programme) tarafından sunulan bilgiler, Grup 6'ya dahil olan ekonomik aktivitelerinin küresel çapta en büyük ikinci endüstrisi olduğunu göstermektedir. Bu bilgiler ışığında çok amaçlı tek dönemli envanter modellerinin yönetimi sadece işletmeye değil küresel ekonomiye de etki ettiği görülebilir (UNSD, 2019).

Envanter problemi genelde evrimsel algoritmalar başta olmak üzere metasezgisel algoritmalarla çözülür. Optimizasyon algoritmaların türleri **Şekil 1.11**'de sunulmuştur.



Şekil 1.11 : Optimizasyon Algoritmaları

Şekil 1.11'de görülen metasezgisel algoritmaların bazı dezavantajları yüzünden evrimsel algoritmalar çoğunlukla bu algoritmalara tercih edilmektedir. Lokal arama algoritmaları lokal yapılarından dolayı lokal optimumlara yakınsayabilir. Tavlama

algoritması çözümlerin yayılmasını sağlamak açısından başarılı olamamaktadır. Ayrıca asimptotik zaman karmaşıklığına sahip olduğundan işlenen veri büyüdükçe karmaşıklık üssel olarak artmaktadır. Tabu arama algoritmaları sürekli çözüm uzaylarında ve birden çok amaç fonksiyonu bulunana problemlerde başarılı olamamaktadır. Ayrıca yüksek zaman ve yer karmaşıklığı yüzünden diğer algoritmalarla hibrit olarak kullanılmaya elverişsizdir. Sürü tabanlı algoritmalar diğerlerine kıyasla daha popülerdir fakat önemli dezavantajları vardır. İyi çözümlere ulaşabilmesi için birden çok kez çalıştırılması gereklidir. Ayrıca çözüm kümesinde yayılmayı sağlamakta zorluk çekmektedir. Evrimsel algoritmalara karşı bir diğer zayıf noktası da, evrimsel algoritmaların aksine parametre ayarları kısıtlı olarak yapılabilmektedir (Coello, 2007).

Evrimsel algoritmalar; gösterge tabanlı, dağılma tabanlı ve Pareto tabanlı olmak üzere üçe ayrılır. Bunlardan en popüler ve kabul görenleri Pareto tabanlı algoritmalardır. Pareto tabanlı algoritmlara seleksiyon işlemini Pareto dominasyonu ilkesine dayanark yapar ve yayılmayı korumak için özel operatörler kullanır. Gösterge ve dağılma tabanlı algoritmaların Pareto tabanlı algoritmalara karşı önemli zayıflıkları vardır. Gösterge tabanlı algoritmalar bir performans metriğinin kılavuzluğunda arama yapar. Bu sebeple problemin boyutları arttıkça zaman karmaşıklığı hızlı bir şekilde artar. Ayrıca ilk başta metriğin değerini hesaplamak için seçilen referans noktasının yerinin performans üzerinde önemli etkileri vardır. Dağılma tabanlı algoritmalar amaç fonksiyonlarına ağırlıklar atayarak problemi alt problemlere böler. Bunu yapabilmek için karar vericinin çözüm uzayı hakkında bilgi sahibi olması gerekir. Ayrıca amaç fonksiyonları arttıkça ağırlık sayısının üssel olarak artması önemli bir karmaşıklık artışına neden olur. Bu sebepler dolayısıyla Pareto tabanlı algoritmalar diğer algoritmalara tercih edilmektedirler (Emmerich, 2018).

Pareto tabanlı algoritmaların en güvenilir ve saygın üyeleri SPEA ve NSGA ailesine ait olanlarıdır. SPEA ailesinin son üyesi SPEA2 (Zitzler, 2001) ve NSGA ailesinin son üyesi NSGA-III'tür (Deb, 2014). Bundan dolayı envanter probleminin çözümü için bu algoritmaların kullanılması idealdir. Bu iki algoritmanın hangisinin daha iyi olduğu karşılaştırılmalı ve en iyi olan daha da geliştirilerek envanter problemi daha verimli şekilde çözülmelidir.

Bu deneylerde elde edilen bilgiler doğrultusunda geliştirilmiş olan optimizasyon algoritması gerçek dünyadaki problemleri çözmeye hazır olacaktır. Nitekim,

çalışmanın en son bölümünde geliştirilmiş olan algoritma gerçek bir envanter problemine uygulanıp elde edilen kazanç gözlenecektir.

1.3 Literatür Araştırması

Envanter problemi bilimsel dünyada popülerlik kazanan bir konudur ve bu alanda bir çok çalışma yapılmıştır. En çok kullanılan çözüm yöntemleri evrimsel algoritmalar ailesine ait algoritmalardır. NSGA-II ve NSGA bu algoritmalar arasındadır. Ayrıca parçacık sürü algoritması kullanılan diğer bir popüler yöntemdir.

Cholodowicz vd.'nin yaptığı çalışmaya göre SPEA2 ve NSGA-II envanter probleminin çözümünde etkili olan algoritmalardır (Cholodowicz, 2017). Ayrıca Azuma vd. SPEA2'yi envanter optimizasyonu üzerinde denemiştir (Azuma, 2011). 2010 yılında Sanchez vd. tarafından yapılan bir araştırmada SPEA2, bir SPEA2 türevi, NSGA-II ve MOPSO envanter optimizasyonu için kullanılmıştır. Yapılan bütün bu araştırmalarda SPEA ve NSGA ailesine ait olan algoritmaların envanter çözümü için iyi birer seçenek olduğu sonucuna varılmıştır (Sanchez, 2010). Fakat SPEA2 ve NSGA-III daha önce hiç karşılaştırılmamıştır. Bu noktada aydınlanması gereken bir belirsizlik mevcuttur.

Bunun yanı sıra envanter problemi bulanık simülasyon ve hedef programlama ile çözülmüştür (Hosseini,2009). Kesin sonuç bulan analitik çözüm yöntemleri kullanılan diğer algoritmalar arasındadır (Chen, 2013). Lokal arama algoritması (Hopp, 1997), dal-sınır algoritması (Hnainen, 2016), indirgenmiş gradyan yöntemi (Panda, 2008), memetik algoritma (Pasendideh, 2013) ve tavlama algoritması (Pasendideh, 2017) ile de envanter problemi çözülmüştür.

Orijinal SPEA algoritmasının zayıf yönleri vardır. SPEA bireyin seleksiyonu ve uygunluk atamasını domine edildiği çözüm sayısına göre yapar. Arşiv boyutunun bir olduğu bir durumda ise bütün çözümler aynı tek çözüm tarafından domine edildiği için hepsinin uygunluk derecesi aynı olur. Ayrıca bir yoğunluk hesaplama operatörü kullanılmadığından, bazı çözümler tek noktaya yakınsayıp yayılmayı düşürebilir. Bir diğer zayıf noktası ise, arşiv budama operatörünün uç çözümleri eleyebilmesidir (Zitzler, 1999).

Bu dezavantajları yüzünden SPEA'nın geliştiricisi Zitzler vd., 2001 yılında SPEA2 algoritmasını geliştirmişlerdir. SPEA2'de her çözüm için uygunluk hesaplanırken domine edildiği birey sayısının yanı sıra domine ettiği bireyler de hesaba

katılmaktadır. Ayrıca budama operatörü geliştirilerek arşivin boyutunun sabit kalması sağlanmıştır (Zitzler, 2001).

İlerleyen yıllarda SPEA2 algoritmasının üzerine inşa edilen en önemli algoritma SPEA2+ algoritmasıdır. Bu çalışmada genetik değişim operatörü iyileştirilmiştir. Ayrıca arşiv işlemlerine yenilikler eklenmiştir. Bunları yaparak daha iyi dağılma performansı elde edilmiştir (Kim, 2004).

SPEA2 algoritmasına diğer bilim insanları tarafından başka yenilikler de getirilmiştir. Bir çalışmada ikili genetik değişim stratejisi izleyen, polinom mutasyon operatörüne sahip diferansiyel evrimsel operatöre sahip bir SPEA2 algoritması geliştirilmiştir. Diferansiyel evrim, çözüm vektörlerinin parçalarının çözüm uzayında birbirileriyle birleşerek daha iyi çözümler bulmaları esasına dayanır (Zhao, 2016). Kim vd. SPEA2 için daha verimli bir genetik değişim stratejisi ve arşiv mekanizması geliştirmişlerdir (Kim, 2004). Zheng vd. SPEA2'yi paralel genetik algoritma ile hibrit olarak kullanmışlardır (Zheng, 2009). Wu vd.'nin geliştirdiği bir algoritma SPEA2 benzeri bir uygunluk operatörü kullanmaktadır (Wu, 2009). Li vd. SPEA2 ile lokal arama algoritmasını hibrit olarak kullanmıştır (Li, 2010). Başka bir çalışmada quasigradyan lokal arama algoritması ile SPEA2 beraber kullanılmıştır (Belgasmi, 2011). Al-Hajiri ve Abido'nun geliştirdği SPEA2 türevinde budama operatörü arşivin boyutunu değiştirmektedir. Aynı zamanda en iyi sonuç domine edilmeyen sonuçlar arasından bulanık mantık kullanılarak seçilmektedir (Al-Hajiri, 2011). Bir diğer çalışmada kısıtlamaların ceza fonksiyonu olarak kullanıldığı ve evrimsel operatörlerin duruma göre uyum sağlayarak değiştiği bir SPEA2 uygulaması yapılmıştır (Sheng, 2012). Maetha ve Dabhi çalışmalarında yakınsama ve yayılmaya aynı anda iyileştirmeyi amaçlayarak farklı bir genetik değişim operatörü geliştirmişlerdir (Maetha, 2014).

SPEA ailesinin en yakın rakibi olan NSGA ailesinin en eski üyesi NSGA'dır (Srinivas, 1994). NSGA-II hızlı elitist bir sıralama stratejisi kullanarak ve yayılmayı arttırmak amacıyla çözümlerin kalabalıklaştığı noktaları seyrekleştiren bir operatörle NSGA algoritmasını önemli ölçüde geliştirmiştir (Deb, 2000). NSGA-III, NSGA ailesinin en son üyesidir. 2014 yılında geliştirilen NSGA-III, NSGA-II'ye göre temelde aynı çalışma prensibine sahiptir. Aralarındaki en önemli fark, NSGA-III'de, sayıları amaç fonksiyonlarının sayısına göre değişen ve algoritmanın başlamasıyla oluşturulan referans noktalarıdır. Referans noktaları ilk çözümler olarak kabul edilir. Çözüm kümesi oluşturulurken ise bu referans noktalarına daha yakın olan çözümlerin daha uygun olduğu kabul edilir (Deb, 2014).

2016 yılında yapılan bir çalışmada, daha az uygun olan sonuçları koruyarak yayılmayı arttırmayı hedefleyen ve geliştirilmiş bir uygunluk fonksiyonu kullanan yeni bir NSGA-III türevi ortaya konmuştur (Bhesdadiya, 2016). Bi ve Wang çözüm uzayını parçalara bölerek, algoritmayı aynı anda farklı parçalarda çalıştırıp parçaların birbirileriyle bilgi alışverişinde bulunduğu bir NSGA-III türevi önermişlerdir (Bi, 2017). 2014 yılında bölgesel dominans esasına dayanan bir NSGA-III türevi geliştirilmiştir (Yuan, 2014). NSGA-III'ün bir diğer versiyonu olan u-NSGA-III ise eldeki problemi eşit bir biçimde eşit bir biçimde alt problemlere dağıtan bir algoritmadır (Seada, 2014).

1.4 Hipotez

Envanter probleminin en başarılı çözüm yöntemleri olan NSGA ve SPEA ailesine bağlı algoritmalar yeterli düzeyde test edilmelidir. Bu testlerin sonunda hangi algoritmanın daha başarılı olduğu ortaya çıkacaktır. Bu testleri gerçekleştirmek için bir envanter problemi modellenmeli ve gerekli metrikler kullanılarak hangisinin daha iyi olduğu kesin olarak ortaya konulmalıdır. Bu sebepten dolayı model ve metrikler doğru seçilmelidir. Bu test için seçilen model, en çok karşılaşılan model olan çok amaçlı tek dönemli çok ürünlü envanter problemidir. Ayrıca iki amaç fonksiyonu olan bu model için iki adet tekli metrik gerekmesine rağmen üç adet tekli metrik kullanılacaktır. Hem yakınsamayı hem de dağılmayı hesaplayan farklı metrikler kullanılacaktır. Çıkan sonuç bir algoritmanın en iyi olduğunu ortaya koysada, diğer algoritmanın saygınlığı ve güvenilirliği ortalamanın çok üstünde olacaktır. Bu sebeple çalışmanın ikinci bölümünde bu algoritmaya de yer verilecektir.

Tez çalışmasının ikinci bölümünde, başta performansı en iyi olan algoritma olmak üzere birinci bölümde kıyaslanan algoritmalar geliştirilip test edilecektir. Birinci bölümde test edilen algoritmalar Pareto bazlı evrimsel algoritmalardır. Bu algoritmaların yayılma değeri çok iyidir ve yakınsama değerini gölgede bırakmaktadır. Bu çalışmada amaçlanan hem yakınsama hem de yayılma değerlerini yükseltmek ve yakınsama başarısını yayılma başarısına yakınlaştırmaktır.

SPEA ailesine ait algoritmasının sonuçların belirli bir noktada toplanmasını önleyen seyreltme işlemine dayalı bir budama operatörü vardır. Budama operatörü iyileştirilerek, domine edilmeyen sonuçlara karşı pozitif ayrımcılık yapması sağlanacaktır. Yani popülasyona orantılı miktarda domine edilmeyen sonuç,

bulundukları bölgenin yoğunluğuna bakılmaksızın mutlak olarak gelecek popülasyon için saklanacaktır. Ayrıca bu algoritmanın seleksiyon mekanizmasında geliştirmeler yapılacaktır. Domine edilmeyen sonuçların uygunlukları bir kıdem sistemiyle zamana bağlı olarak artacaktır. Şöyle ki, her iterasyon sonunda domine edilmeyen sonuçların uygunlukları popülasyona orantılı bir miktar kadar iyileştirilecektir. Böylece yakınsamanın artması sağlanacaktır. Üçüncü yenilik ise yayılmayı arttırmayı hedeflemektedir. Bu noktada üreme politikası üzerinde iyileştirme yapılmıştır. Normal rekombinasyon gerçekleştikten sonra, popülasyon ve arşivde en kötü üyeler bulunup tekrar çiftleşttirilecektir. Elde edilen yeni bireyler popülasyonda aynı miktar kadar bireyin yerine konulacaktır. Bu bieylerin sayısı popülasyonun boyutu ile orantılıdır.

NSGA ailesine ait algoritmaya da benzer üç yenilik yapılacaktır. NSGA algoritmaları uygunluk hesaplarken sonuçları sıralı sınıflara ayırır. Domine edilmeyen sonuçların bulunduğu sınıfın uygunlukları benzer şekildeki bir kıdem sistemiyle arttırılacaktır. Burada uygunluğun arttırılacağı miktar yine popülasyon boyutuyla orantılıdır. İkinci yenilik, seleksiyon sonrası en iyi bireylerin kopyalanıp aynı miktardaki en kötü bireylerin yerine koyulmasıdır. Buradaki birey sayısı popülasyon boyutuyla orantılıdır. Bu iki yenilik yakınsamayı arttırmak için uygulanacaktır. Yayılmayı arttırmak için SPEA ailesine ait algoritmaya yapılan yeniliğe benzer bir yenilik getirilecektir. Rekombinasyon sonrası elde edilen sonuçlarda en kötüleri tekrar çiftleştirilip aynı miktarda rassal bireylerin yerini alması sağlanacaktır. Buradaki birey miktarı yine popülasyon boyutuyla orantılıdır.

Geliştirilen bu algoritmalar kapsamlı bir test sürecine tabi tutulacaktır. Öncelikle iç bükey ve dış bükey Pareto cepheleri olan Fonseca ve Binh problemleri üzerinde deneneceklerdir. Sonra ayrık bir Pareto cephesine sahip olan WFG problemini çözmeleri sağlanacaklardır. En sonunda iki adet envanter modeli üzerinde denenip amaca ulaştıkları gösterilecektir. Buradaki envanter problemleri çok amaçlı tek dönemli envanter problemleridir. Problemlerden biri çok ürünlü diğeri tek ürünlüdür. Aynı problemler geliştirilmiş algoritmaların yanı sıra bu iki algoritmanın orijinal versiyonları ve bir sürü algoritması olan OMOPSO ile çözüleceklerdir. Bu deneyler sonunda elde edilen verilerle çalışmanın bir önceki bölümünde olduğu gibi yeterli miktarda, yakınsama ve yayılmayı ölçen metrik değerleri hesaplanacaktır. Üst hacim, kuşaksal mesafe ve boşluk kullanılacak olan metriklerdir.

Bütün bu deneyler sonunda en iyi sonucu veren geliştirilmiş algoritma gerçek bir envanter problemine uygulanacaktır. Problem tek bir ambarı olan tek kademeli bir

envanter sistemine sahiptir. Bir dönemi iki hafta sürmektedir ve algoritma bir dönem için test edilecektir. Algoritma test problemine benzer bir problemi çözecektir. Bu sebepten dolayı başarılı olması beklenmektedir.

2. ÇOK AMAÇLI TEK DÖNEMLİ ÇOK ÜRÜNLÜ ENVANTER PROBLEMİNİN ÇÖZÜMÜNDE NSGA-III VE SPEA2 ALGORİTMALARININ KARŞILAŞTIRILMASI

2.1 Amaç

Bu bölümde yapılan çalışmada envanter probleminin çözümü için en uygun algoritmanın hangisi olduğu bulunmak istenmiştir. Buradaki deneyler bir sonraki bölümde yapılacak çalışmaya yön verecektir. Bu bölümde, öncelikle çok amaçlı tek dönemli çok ürünlü bir envanter problemi modellenecektir. Bu matematiksel model SPEA2 ve NSGA-III algoritmalarıyla çözülecektir. Elde edilen çözümler ile üsthacim, kuşaksal mesafe ve boşluk metrikleri hesaplanacaktır. Hesaplanan bu değerler hangi algoritmanın bu problem için daha iyi performans sergilediğini ortaya koyacaktır.

Her optimizasyon probleminin farklı özellikleri vardır. Amaç fonksiyonu sayısı, kısıtlama sayısı ve karar değişkeni sayısı problemden probleme farklılık gösterebilir. Bunun yanı sıra problem polinom olabilir veya olmayabilir. Ayrıca, çözüm uzayında da farklılıklar olabilir. Örneğin bir problemin gerçek Pareto cephesi içbükey şeklindeyken başka bir problemin gerçek Pareto cephesi dış bükey şeklinde olabilir. Bunun yanı sıra, gerçek Pareto cephesi sürekli veya ayrık olabilir. Parametrelere getirilen kısıtlamaların yanı sıra çözüm uzayı da kısıtlı olabilir. Bu sebepten dolayı bir optimizasyon algoritması bir problemi çözerken iyi performans gösteririken diğer bir problemi çözerken kötü performans gösterebilir.

Bu durum envanter optimizasyonu için de geçerlidir. Başka problemlerin çözümünde başarılı olan bir algoritma envanter probleminin çözümünde başarısız olabilir. Bunun yanı sıra, her envanter modeli aynı değildir. Bir önceki paragrafta sıralanan özellikler bir envanter problemi modelinden başka bir envanter problemi modeline farklılık gösterebilir. Sayısız envanter modeli mevcut olmasından dolayı her envanter modeli üzerinde deney yapmak imkansızdır. Ama amacına uygun ve kapsamlı bir envanter modeli seçerek başarılı bir deney gerçekleştirilebilir.

Dünya üzerindeki envanter modellerinin çoğu süpermarket zincirlerinin envanter modellerine uymaktadır. Bir süpermarketin envanterinde çoğunlukla son kullanma

tarihi bulunan ve son kullanma tarihinden sonra zayi olan birden çok ürün vardır. Ayrıca bu tür envanterlerin yöneticilerinin ulaşmak istedikleri en önemli hedefler arasında karı ve ambar doluluk oranını maksimize etmek bulunmaktadır. Ambar doluluk oranı genelde servis oranı ile hemen hemen aynı anlama gelmektedir. Süpermarket envanterlerinin envanter modelleri butikler ve teknoloji mağazaları gibi başka işletmelerin envanter modellerini kapsamaktadır. Ayrıca kullanılan çoğu model yapı olarak bu model benzemektedir. Bu sebepten dolayı bu çalışmada süpermarket zincirlerinin envanter modeli kullanılacaktır. Bu model çok amaçlı tek dönemli çok ürünlü envanter modelidir.

Evrimsel algoritmalar genelde problemden probleme farklılıklar gösterse de ve tek bir evrimsel algoritmaya en iyi algoritma denilemese de, bazı algoritmalar genelde diğer algoritmalarda daha başarılıdır. Pareto tabanlı algoritmaların farklı çeşit ve özellikteki Pareto cephelerini bulmada diğer evrimsel algoritmalardan genelde daha başarılı olduğu kanıtlanmıştır. Bunun yanı sıra, bazı Pareto tabanlı algoritmaların diğer algoritmalardan çoğunlukla daha iyi performans sergilediği bilinmektedir. En iyi performans gösteren algoritmalar SPEA ve NSGA ailesine ait olan algoritmalardır. SPEA ailesinin en son üyesi olan SPEA2 göreceli olarak eski bir algoritma olsa da günümüzde hala diğer algoritmalardan daha iyi performans gösterdiği bilinmektedir. Diğer taraftan 2014 yılında geliştirilen ve NSGA ailesinin son üyesi olan NSGA-III algoritması üzerinde SPEA2'ye göre daha az çalışma yapılmıştır. Yine de bu çalışmalarda NSGA-III algoritmasının başarılı bir algoritma olduğu gözlemlenmiştir. Ayrıca NSGA-III'ün bir önceki versiyonu olan NSGA-II algoritması bilim camiası tarafından en çok güvenilen ve saygı duyulan, başarıları kanıtlanmış bir algoritmadır. Bu sebeplerden dolayı bu deneyde SPEA2 ve NSGA-III algoritmaları kıyaslanmıştır.

2.2 Problem Modeli

Bu çalışmadaki çok amaçlı tek dönemli çok ürünlü olarak modellenen envanter probleminin amaç fonksiyonları karı ve ambar doluluk oranını maksimize etmektir. Ambar boyutu ve sipariş miktarı üzerinde kısıtlamalar bulunmaktadır. Sipariş miktarı karar değişkenidir. Modelde kullanılan kısaltmaların açılımları şu anlamları taşımaktadırlar:

i = 1, 2, 3....n için

ZK : Beklenen kar

ZD : Beklenen ambar doluluk oranı

D : Ambar kapasitesi

G : Beklenen gelir

Cb : Beklenen bulundurma maliyeti

Cs : Beklenen stoksuzluk maliyeti

Cm : Beklenen sipariş maliyeti

T : Ürün sayısı

P : Satış ücreti

Qi : i. ürünün sipariş miktarı

Xi : i. ürünün stokastik talebi

Fxi(xi) : i. ürünün talep probabilite fonksiyonu

λi : i. ürünün beklenen talep miktarı

μi : i. ürünün beklenen maksimum talep miktarı

πi : i. ürünün beklenen minimum talep miktarı

b : Bulundurma maliyetinin satış fiaytına olan yüzde oranı

s : Stoksuzluk maliyetinin satış fiyatına olan yüzde oranı

m : Sipariş maliyetinin satış fiyatına olan yüzde oranı.

Kar stokastik talebe ve sipariş miktarına bağlı olarak değişmektedir. Eğer sipariş miktarı talep miktarında daha az veya talep miktarına eşit ise satılan miktar talep miktarına eşit olur. Eğer talep miktarı sipariş miktarından daha az ise talep miktarı satış miktarı olur. Buna göre gelir **(2.1)** eşitliği ile hesaplanır.

$$G = \sum_{i=1}^{T} Pimin(Qi, Xi)Fxi(xi) \qquad \textbf{(2.1)}$$

Beklenen bulundurma, stoksuzluk ve sipariş maliyetleri **(2.2)**, **(2.3)** ve **(2.4)** eşitlikleri ile hesaplanır.

$$Cb = \sum_{i=1}^{T} (Q_i - X_i)Fx_i(x_i)P_i\, b \qquad \textbf{(2.2)}$$

$$Cs = \sum_{i=1}^{T} (Q_i - X_i)Fx_i(x_i)P_i\, s \qquad \textbf{(2.3)}$$

$$Cm = \sum_{i=1}^{T} (Q_i - X_i)Fx_i(x_i)P_i\, m \qquad \textbf{(2.4)}$$

Kar **(2.5)** eşitliği ile hesaplanabilir. Problem sürekli tek biçimli bir dağılıma sahip olduğundan λ_i ve $Fx_i(x_i)$ **(2.6)** ve **(2.7)** eşitliklerinden elde edilebilir.

$$Z_K = \sum_{i=1}^{T} P_i \min(Q_i, X_i) Fx_i(x_i) - \sum_{i=1}^{T} (Q_i - X_i) Fx_i(x_i) P_i b - \sum_{i=1}^{T} (X_i - Q_i) Fx_i(x_i) P_i s - \sum_{i=1}^{T} (Q_i - X_i) Fx_i(x_i) P_i m \quad \textbf{(2.5)}$$

$$\lambda_i = (\mu_i + \pi_i)/2 \quad \textbf{(2.6)}$$

$$Fx_i(x_i) = 1/(\mu_i - \pi_i) \quad \textbf{(2.7)}$$

İkinci amaç fonksiyonu olan Z_D, ambar kapasitesi ve toplam doluluk oranı arasındaki farktan bulunabilir. Bu, eşitlik **(2.8)**'de gösterilmiştir.

$$Z_D = \sum_{i=1}^{T} Q_i - D \quad \textbf{(2.8)}$$

Kısıtlamalar **(2.9)** ve **(2.10)**'da modellenmiştir.

$$Z_D = \sum_{i=1}^{T} Q_i - D \quad \textbf{(2.9)}$$

$$\pi_i \leq Q_i \leq \mu_i \quad \textbf{(2.10)}$$

Modelin son hali **(2.11)**'de gösterilmiştir.

Maksimize et:

$$Z_K = \sum_{i=1}^{T} P_i \min(Q_i, X_i)/(\mu_i - \pi_i) - \sum_{i=1}^{T} (Q_i - X_i)/(\mu_i - \pi_i) P_i b - \sum_{i=1}^{T} (X_i - Q_i)/(\mu_i - \pi_i) P_i s \sum_{i=1}^{T} (Q_i - X_i)/(\mu_i - \pi_i) P_i m$$

Kısıtlamalar:

$$\sum_{i=1}^{T} Q_i \leq D, \ \pi_i \leq Q_i \leq \mu_i \quad \textbf{(2.11)}$$

Problemin çözülmesi için gerekli olan veriler **Çizelge 2.1**'de verilmiştir.

Çizelge 2.1 : Problem kurulumu için gereken veriler

Ambar Kapasitesi : 150					
Ürün(i)	1	2	3	4	5
μ(i)	20	43	11	85	36
π(i)	5	24	2	29	30
P(i)	4	3	9	2	3
s(i)	0.85	0.85	0.85	0.85	0.85
b(i)	0.87	0.87	0.87	0.87	0.87
m(i)	0.55	0.55	0.55	0.55	0.55

2.3 Deney Ortamı

Deney için MOEA kütüphanesi kullanılmıştır. MOEA açık kaynaklı bir Java kütüphanesidir. MOEA ile optimizasyon algoritmaları geliştirilebilmekte, test edilebilmekte ve karşılaştırılabilmektedirler (Hadka, 2019). Deney için gerekli olan programlama Java 8 programlama diliyle Ecplise ortamında gerçekleştirilmiştir. Deney için modellenen problem SPEA2 ve NSGA-III algoritmalarıyla 10'ar defa çözülmüştür. Çıkan sonuçlar kayıt altına alınıp üst hacim, kuşaksal mesafe ve boşluk metriklerinin ortalama, maksimum ve minimum değerleri hesaplanmıştır. Sonuçların grafiksel gösterimi için plot.ly Python kütüphanesi kullanılmıştır (Plotly, 2020). Deneyin gerçekleştrildiği bilgisayarın 2.2GHz'lik 2 çekirdekli işlemcisi ve 16 GB Ram'i vardır.

2.4 Sonuçlar

Deney sonucu elde edilen üst hacim değerleri **Çizelge 2.2**'de, kuşaksal mesafe değerleri **Çizelge 2.3**'te, boşluk değerleri **Çizelge 2.4**'te sunulmuştur. Bütün çözümlerin birleşik Pareto cephesi **Şekil 2.1**'de görülmektedir.

Çizelge 2.2 : Üst hacim

Metrik Değeri	NSGA-III	SPEA2
Minimum	0	0
Ortalama	0.067221007	0.094919746

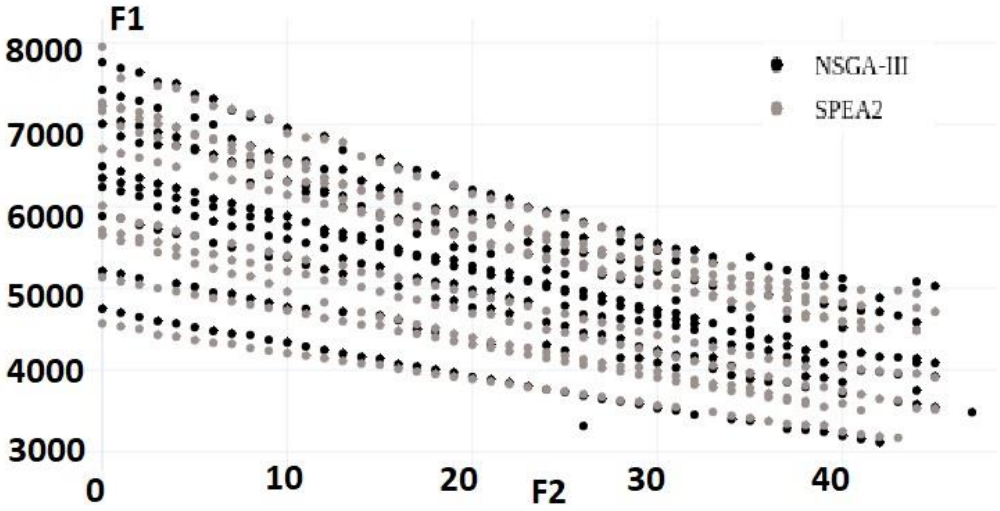
Maksimum	0.537886336	0.398168237

Çizelge 2.3 : Kuşaksal mesafe

Metrik Değeri	NSGA-III	SPEA2
Minimum	0	0.016821037
Ortalama	0.096327886	0.088705536
Maksimum	0.221913876	0.160462133

Çizelge 2.4 : Boşluk

Metrik Değeri	NSGA-III	SPEA2
Minimum	11.62307543	7.773394218
Ortalama	23.7473377063504	14.35630048
Maksimum	55.53887269	25.43976436



Şekil 2.1 : Ortak Pareto cephesi

Yapılan deney sonunda elde edilen performans metrikleri değerlendiridiğinde, ilk bakışta SPEA2 algoritmasının üst hacim metriklerinin NSGA-III algoritmasının değerlerine göre yüksek olmasına rağmen bu değerlerin birbirine yakın olduğudur. Boşluk değeri ele alındığında ise SPEA2 algoritmasının büyük bir farkla NSGA-III algoritmasından daha iyi performans sergilediği görülmektedir.

Üst hacim değerleri ele alındığında iki algoritmanında minimum değerlerinin sıfır olduğu görülmektedir. NSGA-III algoritmasının maksimum üst hacim değeri SPEA2

algoritmasının maksimum üst hacim değerinden fazladır. Ama bu NSGA-III algoritmasının bir uç sonuç elde etmesiyle ilgilidir. Çünkü genel olarak SPEA2 algoritmasının üst hacim değeri NSGA-III algoritmasının üst hacim değerinden daha iyiydir. Şöyle ki, SPEA2 algoritması %29.181 oranında daha yüksek bir ortalama değere sahiptir. SPEA2'nin ortalama kuşaksal mesafe değeri de NSGA-III için karşılık gelen değerden %27.691 oranında daha düşüktür.

İki algoritmanın kuşaksal mesafe değerleri üst hacim metriği değerlerinden daha yakındır. Genel olarak SPEA2'nin kuşaksal mesafe değerleri daha iyidir. NSGA-III sadece minimum değerler açısından SPEA2'den daha iyi performans göstermiştir. NSGA-III'ün minimum kuşaksal mesafe değeri 0'a yakındır. Ama SPEA2 için bu değer 0.016 civarındadır. SPEA2'nin maksimum kuşaksal mesafe değeri ise NSGA-III'ün maksimum kuşaksal mesafe değerinden %7.912 kadar daha düşüktür.

SPEA2 ve NSGA-III algoritmalarının boşluk değerleri birbirine daha uzaktır. Bu metrikte de SPEA2'nin üstünlüğü bulunmaktadır. SPEA2'nin minimum boşluk değeri NSGA-III'ün minimum boşluk değerinden %33.121 daha düşüktür. Ayrıca SPEA2'nın maksimum değeri NSGA-III'ün maksimum değerinden %39.545 daha düşüktür. Ortalama değerlere bakıldığında ise SPEA2'nin boşluk değeri NSGA-III'ün boşluk değerinden %54.194 daha düşüktür. Buradan yayılma özelliği açısından SPEA2'nin çok daha iyi olduğu görülmektedir.

Bu deneylerden çıkan başka bir sonuç ise Pareto tabanlı iki algoritma arasında farkın büyük bölümünü yaratanın yayılma özelliği olduğu gözlenmiştir. Buradan, bu algoritmaların yakınsama değerlerini iyileştirmenin geliştirmeye açık olduğu anlaşılmaktadır.

Çıkan sonuçlar doğrultusunda tezin bir sonraki bölümünde SPEA2 algoritması üzerinde yoğunlaşılacaktır. Ama kıyaslama açısından NSGA ailesinden bir algoritma yine çalışmada kullanılacaktır. Bu deneyin sonucu olarak envanter problemlerinin çoğunun çözümünde SPEA2 algortimasının daha başarılı olduğu bulgusuna varılmıştır. Yine de her problem için SPEA2 algoritmasının en iyi seçenek olduğu kanaatine varılamaz. SPEA ve NSGA ailesine ait algoritmaların farklı problemlerin çözümünde kıyaslanması ayrı bir çalışma konusudur.

3. GELİŞTİRİLMİŞ SPEA2 İLE ENVANTER OPTİMİZASYONU

3.1 Amaç

Envanter optimizasyonu için en iyi çözüm yolu olan evrimsel algoritmalardan en iyi ikisi bir önceki bölümde test edilmiştir. Bu algoritmalar, yani SPEA2 ve NSGA-III yayılmayı ve yakınsamayı ölçen metrikleri ile performans analizine tabi tutulmuşlardır. SPEA2'nin daha iyi performans sergilediği sonucuna varılmıştır. Envanter probleminin önemi önceki bölümde anlatılmıştır. Anlatılan sebeplerden dolayı SPEA2'nin geliştirilmesi aynı oranda önem taşımaktadır. Bu bölümde envanter optimizasyonunda zaten başarılı olan bu algoritmanın daha iyi hale getirilmesi amaçlanmaktadır.

Bir önceki bölümde elde edilen bulgular, SPEA2 ve NSGA-III'nin yakınsama değerlerinin birbirilerine yakın olduğu ve yayılma değerlerinin arasında daha büyük bir fark olduğu doğrultusundandır. Pareto tabanlı algoritmaların en büyük avantajı yayılmayı koruyan seyreltme mekanizmaları kullanmalarıdır. Bu sebeplerden ötürü bu algoritmaların yakınsama özelliklerinin gelişmeye açık olduğu görülmektedir.

Bu bölümde hedeflenen asıl yenilik SPEA2 algoritmasının yakınsama performansını iyileştirmektir. İkinci olarak yayılma değerinin de iyileştirilmesi hedeflenmektedir. Bunun yanı sıra NSGA-II' de başarıları kanıtlanmış saygı gören bir algoritma olduğu için benzer yeniliklere tabi tutulacaktır.

Yenilenmiş olan SPEA2 algoritmasına kapsamlı deneyler uygulanacaktır. Öncelikle, bilim camiası tarafından evrimsel alogirtmaları test etmek amacıyla kullanılan üç test fonksiyonu SPEA2' ile çözülecektir. Daha sonra iki envanter problemi matematiksel olarak çözümlenecektir. Bu problemler çok amaçlı tek dönemli problemlerdir. Deneylerin daha kapsamlı olması için modellerden birinde tek ürün diğerinde çok ürün olacaktır. Bu test problemleri ve envanter modelleri geliştirilmiş SPEA2 ile karşılaştırma amacıyla diğer dört farklı algoritma ile de çözülecektir. Geliştirilmiş SPEA2 ile kıyaslanan ilk algoritma benzer değişiklikler geliştirilmiş olan NSGA-II algoritması olacaktır. Bu iki algoritmanın yeniliklerinin birer gelişme olduğunu kanıtlamak amacıyla, bu algoritmaların orijinal versiyonları olan SPEA2 ve NSGA-II

karşılaştırmaya dahil edilecektir. Karşılaştırılacak olan diğer algoritma ise optimize edilmiş parçacık sürü optimizasyonu olan OMOPSO olacaktır. Deneyin daha kapsamlı olması amacıyla evrimsel algoritma olmayan bu algoritma da deneye dahil edilecektir. Bu problemler farklı iterasyon sayıları ve farklı popülasyon boyutlarıyla çözülecektir. Elde edilen çözümlerin Pareto sonuçları kaydedilecektir. Bu sonuçlar ile her algoritma için üst hacim, kuşaksal mesafe ve boşluk metrikleri hesaplanacaktır.

3.2 Geliştirilmiş SPEA2

Öncelikle, geliştirilmiş SPEA2 algoritması arşivdeki domine edilmemiş sonuçlar karşı pozitif ayrımcılık yapmaktadır. Bireyin arşivde kaldığı her iterasyon için, bireyin uygunluk değerinden popülasyona orantılı olarak bir miktar çıkarılacaktır. SPEA2 algoritmasının çalışma prensibine göre uygunluk değerinin düşük olması daha iyi bir uygunluk değerini göstermektedir. Çıkarılacak miktar, d, popülasyon boyutu, P'nin 0.0002 ile çarpımıyla elde edilir. Bu miktar uygunluğu 0'ın altına düşürürse uygunluk derecesi 0.0001 olarak ayarlanır. Çıkarılacak bu miktar kümülatiftir. Ayrıca popülasyondaki değişiklikler nedeniyle arşivdeki bir bireyin uygunluğu değişse de bu, üstündeki toplam d değerini etkilemeyecektir.

İkinici yenilik ise seleksiyon sonrası, eğer budama operatörü çalışacaksa devreye girecektir. Budama operatörü budamayı gerçekleştirdikten sonra, arşivdeki en iyi sonuçların kopyalarını alıp, arşivdeki en kötü sonuçların yerine koyacaktır. Bu işlem için kopyalancak ve silinecek birey sayısı yine P'nin 0.0002 ile çarpımından elde edilir. Bu miktar en az 1'dir.

Son yenilik algoritmanın üreme operatörüyle ilgilidir. Normal rekombinasyon gerçekleştikten sonra en az uygun bireyler çiftleştirilip aynı sayıda rassal seçilen bireylerin yerine konur. Burada çiftleştirilecek birey sayısı P'nin 0.0002 ile çarpımından elde edilir. Bu yenilik yayılmayı iyileştirmek amacıyla yapılmıştır. Önceki diğer iki yenilik yakınsamayı iyileştirmeyi hedeflemektedir. Üç yenilikte de P'nin 0.0002 ile çarpımından elde edilen sayı en yakın birler basamağına yuvarlanır. Geliştirilmiş SPEA2'nin pseudokodu **Algoritma 4**'te görülmektedir.

Algoritma 4:

Procedure Geliştirilmiş SPEA2

Input: Y

Initialize Z0

Set W0=Ø,T= Ø, u=0, nds=0

While u<Y

Zu ve Wu'nun uygunluklarını hesapla

Domine edilmemiş bireylerin uygunluklarını Z'nin boyutunun 0.0002 ile çarpımı kadar düşür

nds=Zu ∪ Wu'dan gelen domine edilmemiş sonuçlar

if nds>T then

Budama operatörünü kullan

Else if nds<T

Wu 'daki boş yerleri en uygun domine edilen sonuçlarla doldur

Else

Wu'yu domine edilmemiş sonuçlarla doldur.

End if

0.0002XZ en kötü bireyi sil ve en iyi 0.0002XZ kopyala

Rekombinasyon ve mutasyon işlemlerini gerçekleştir

En kötü komşu bireyler arasında rekombinasyon gerçekleştir

Set u=u+1

End while

Return Wy

3.3 Geliştirilmiş NSGA-II

Geliştirilmiş SPEA2'de olduğu gibi, geliştirilmiş NSGA-II algoritmasında yakınsama ve yayılma değerlerinin iyileştirilmesi hedeflenmiştir. Yayılmayı iyileştirmek için benzer bir hücresel üreme işlemi de geliştirilmiş NSGA-II'de bulunmaktadır. Geliştirilmiş NSGA-II'de de 0.0002XP miktarda en kötü komşu bireyler, standart rekombinasyon operasyonu sonunda çiftleştirilir. Elde edilen yeni bireyler aynı miktarda rassal bireylerin yerine konulur.

Yakınsama performansını iyileştirmek için iki yenilik yapılmıştır. Bunlardan birincisi, seleksiyon sonrası elde edilen tüm popülasyonun bireylerini kapsamaktadır. Tüm bireyler arasından 0.0002XP kadar en az uygun birey çıkarılıp, aynı miktarda en uygun birey kopyalanmaktadır. Kopyalanan bireyler çıkarılan bireylerin yerine konur.

Yakınsamayı ilgilendiren ikinci yenilik, bir kıdem sistemiyle domine edilmemiş bireylere pozitif ayrımcılık yapmayı kapsamaktadır. Bir birey domine edilmemiş

katmanda bulunduğu her iterasyon için uygunluk değeri 0.0002XP kadar iyileşir. Bu iyileşme kümülatiftir ve popülasyonda olan değişiklikler bu kümülatif değeri etkilemez.

Son yenilik SPEA2'de olduğu gibi üreme mekanizmasıyla ilgilidir. Standart üreme sonunda en az uygun komşu 0.0002XP birey seçilir ve çiftleştirilir. Elde edilen yeni bireyler 0.0002XP kadar rassal seçilen bireyin yerini alır. Bu yenilik yayılmayı iyileştirmek için yapılmıştır. Geliştirilmiş NSGA-II'nin pseudokodu Algoritma 5'te sunulmuştur.

Algoritma 5:

Procedure: Geliştirilmiş NSGA-II

Input: Y

Initialize Z0

Set W0=Ø, e=0, u=0

While e<Y

Set u=0

Ebeveynleri ve çocukları sırala

Domine edilmeyen katmanda bulununa bireylerin uygunluğunu 0.0002×Z kadar iyileştir.

While u<length Z0

We∪Ze için seleksiyon işlemini gerçekleştir.

Üreme ve mutasyon işlemlerini gerçekleştir.

Çocukların uygunluğunu değerlendir

Set u=u+1

End while

Set z=z+1

En kötü 0.0002×Z sonucu çıkar ve en iyi 0.0002×Z sonucu bunların yerine koy

En kötü 0.0002×Z bireye hücresel üreme uygula

End while

Return Ze

3.4 Test Problemleri

Evrimsel algoritmaları test etmek için birçok test problemi ve test problemleri grubu bulunmaktadır. Bu problemlerin hiçbiri gerçek hayatta karşılaşılan problemlerin yerini

almamaktadır. Fakat bu problemler algoritmaları birbiriyle karşılaştırmak için idealdir. Aynı zamanda, gerçek hayatta çözülmek istenilen bir problem için, algoritma o probleme benzeyen bir test problemiyle sınanabilir. Böyle bir durumda test probleminden elde edilen sonuçlar algoritmanın gerçek hayat performansı hakkında bir fikir vermektedir (Coello, 2007).

Bu çalışmada kullanılacak üç problemden ikisi olan Fonseca ve Binh problemlerinin iki amaç fonksiyonu ve iki karar değişkeni vardır (Fonseca, 1995) (Binh, 1997). Bu bakımdan algoritmanın gerçekte uyugulanacağı envanter problemlerine benzemektedirler. Ayrıca Fonseca probleminin içbükey (Fonseca, 1995), Binh probleminin dışbükey gerçek Pareto cepheleri vardır (Binh, 1997). Böylece algortima iki tür Pareto cephesi için test edilmiş olacaktır. Üçüncü problem olan WFG'nin ise bir amaç fonksiyonıu vardır. Karar değişkenlerinin sayısı 10'dur. Bu problemin teste dahil edilmesinin sebebi gerçek Pareto cephesinin hem içbükey hem de dışbükey kısımlarının olmasıdır (Huband, 2006). Fonseca problemi eşitlik (**3.1)**'de, Binh problemi eşitlik **(3.2)**'de, WFG problemi eşitlik **(3.3)**'de verilmiştir.

Minimize et:

$$F_1(x,y)=1-exp[-\sum_{1=1}^{n}(x_i-\frac{1}{\sqrt{n}})^2]$$

$$F_2(x,y)=1-exp[-\sum_{1=1}^{n}(x_i+\frac{1}{\sqrt{n}})^2]$$

Arama alanı:

$$-4\leq x_i\leq 4$$

$$1\leq i\leq n \qquad \textbf{(3.1)}$$

Minimize et:

$$F_2(x,y)=(x-5)^2-(y-5)^2$$

Kısıtlamalar:

$$G_1(x,y) = (x-5)^2 + y^2 \leq 25$$

$$G_2(x,y) = (x-8)^2 + (y-3)^2 \geq 7$$

Arama alanı:

$$-4 \leq x_i \leq 4$$

$$1 \leq i \leq n \qquad \textbf{(3.2)}$$

Mimimize et:

$$F(x_1, \ldots x_n) = \left(1 - x - \frac{\cos(2\pi a x + \frac{\pi}{2})}{2A\pi}\right)^{\alpha}$$

Arama alanı:

$$-10 \leq x_i \leq 10$$

$$1 \leq i \leq 10 \qquad \textbf{(3.3)}$$

3.5 Envanter Problemleri

Bu bölümde kullanılacak olan ilk envanter probleminin amaç fonksiyonları karı ve servis seviyesi maksimize etmektir. Sipariş miktarı ve satış fiyatı karar değişkenleridir. İkisi üstünde de kısıtlamalar vardır. Talep tek biçimli sürekli bir dağılıma sahiptir. Modelde kullanılacak değişkenler şunlardır:

Zk : Beklenen kar

Zd : Beklenen servis seviyesi

P : Satış fiyatı

Pmin : Minimum satış fiyatı

Pmax : Maksimum satış fiyatı

Q : Sipariş miktarı

Qmin : Minimum sipariş miktarı

Qmax : Maksimum sipariş mişktarı

G : Beklenen gelir

X : Stokastik talep

λ : Beklenen talep

μ : Beklenen maksimum talep

π : Beklenen minmum talep

b : Bulundurma maliyetinin satış fiyatına yüzde oranı

m : Sipariş maliyetinin satış fiyatına yüzde oranı

s : Sipariş stoksuzluk satış fiyatına yüzde oranı

Cb : Beklenen bulundurma maliyeti

Cm : Beklenen sipariş maliyeti

Cs : Beklenen stoksuzluk maliyeti

Bekleren gelir, beklenen bulundurma maliyeti, beklenen sipariş maliyeti ve beklenen stoksuzluk sırasıyla maliyeti eşitlik **(3.4),** eşitlik **(3.5)**, eşitlik **(3.6)** ve eşitlik **(3.7)**'görülmektedir. Bekelenen kar eşitlik **(3.8)** ile hesaplanabilir. Beklenen servis seviyesi eşitlik **(3.9)**'da verilmiştir. Kısıtlamaların da katılmasıyla model eşitlik **(3.10)**'da son halini almaktadır.

$$G=XPF(X) \quad \textbf{(3.4)}$$

$$C_b=(Q-X)P_bF(X) \quad \textbf{(3.5)}$$

$$C_m=(Q)P_mF(X) \quad \textbf{(3.6)}$$

$$Cs=(X-Q)P_sF(X) \quad \textbf{(3.7)}$$

$$Z_k=G-C_m-max(C_b, C_s) \quad \textbf{(3.8)}$$

$$Z_d=Q/X \quad \textbf{(3.9)}$$

Maksimize et:

$$Z_k, Z_d$$

Arama alanı:

$$Z_d = Q/X$$

$$Q_{min} < Q < Q_{max} \quad \textbf{(3.10)}$$

İkinci envanter modeli eşitlik (**3.10)**'daki envanter modeline benzemektedir. Aralrındaki tek fark bu modelde *x1,...,xn* adet ürün bulunmasıdır. Gelir **(3.11)**'deki eşitlikle hesaplanmaktadır. Maliyetler eşitlik **(3.12)**, eşitlik **(3.13)** ve eşitlik **(3.14)** ile hesaplanmaktadır. Kar ve servis seviyesi eşitlik **(3.15)** ve eşitlik **(3.16)**'da görülmektedir. Modelin son hali eşitlik **(3.17)**'de verilmiştir.

$$G = \sum_{i=1}^{T} P_i X_i Fx_i(x_i) \quad \textbf{(3.11)}$$

$$C_b = \sum_{i=1}^{T} (X_i - Q_i) Fx_i(x_i) P_i b \quad \textbf{(3.12)}$$

$$C_s = \sum_{i=1}^{T} (X_i - Q_i) Fx_i(x_i) P_i s \quad \textbf{(3.13)}$$

$$C_m = \sum_{i=1}^{T} (Q_i - X_i) Fx_i(x_i) P_i m \quad \textbf{(3.14)}$$

$$Z_k = G - C_m - max(C_b, C_s) \quad \textbf{(3.15)}$$

$$Z_d = Q/X \quad \textbf{(3.16)}$$

Maksimize et:

$$Z_k,$$

$$Z_d$$

Arama alanı:

$$P_{min} < P < P_{max}$$

$$Q_{min} < Q < Q_{max} \quad \textbf{(3.17)}$$

3.6 Deney Kurulumu

Deneyde, çok amaçlı evrimsel algoritmaları geliştirip test emeye yardımıcı bir Java kütüphanesi olan MOEA framework kullanılmıştır (Hadka, 2019). Programlama Java 8 programlama diliyle yapılmıştır. Programlama arayüzü olaraka Eclipse kullanılmıştır. Deneyin yapıldığı bilgisayarın işlemcisi 2.2 Ghz'lik iki çekirdeğe sahiptir. Bilgisayarın 16 GB RAM'i vardır. Bu bölümde anlatılan üç test problemi ve iki envanter modeli; geliştirilmiş SPEA2, geliştirilmiş NSGA-II, SPEA2, NSGA-II ve OMOPSO algoritmalarıyla 1000, 5000 ve 10000 popülasyon değerleriyle 10'ar kez çözülmüştür. Her çözüme 1000 iterasyon ile ulaşılmıştır. Deneyler sonucunda her çözüm ve her algoritma için Pareto sonuçları kaydedilmiştir. Bu sonuçlardan her algoritmanın üst hacim, kuşaksal mesafe ve boşluk değerleri hesaplanmıştır. İlk envanter modelinin çözümü için gereken bilgiler **Çizelge 3.1**'de, ikinci envanterin çözümü için gereken bilgiler **Çizelge 3.2**'de verilmiştir.

Çizelge 3.1 : Tek Ürünlü Envanter Problemi İçin Kurulum Verileri

Qmin(i)	Qmax(i)	Pmin(i)	Pmax(i)	π(i)	μ(i)	b(i)	m(i)	s(i)
100	480	100	300	1	500	0.55	0.6	0.6

Çizelge 3.2 : Çok Ürünlü Envanter Problemi İçin Kurulum Verileri

Ürün(i)	1	2	3	4	5
Qmin(i)	94	103	78	97	85
Qmax(i)	300	275	295	437	320
Pmin(i)	90	94	93	75	40
Pmax(i)	254	200	196	378	320
π(i)	100	200	120	50	100
μ(i)	300	450	490	600	200
b(i)	0.4	0.5	0.2	0.1	0.4
m(i)	0.5	0.5	0.7	0.5	0.5
s(i)	0.3	0.5	0.8	0.3	0.3

Çizelge 3.3 : Fonseca Problemi İçin Metrik Değerleri

Popülasyon	Metrik	NSGA-II	OMOPSO	SPEA2	Yeni SPEA2	Yeni NSGA-II
1000	ÜH	0.394	0.39445	0.39441	0.41928	0.40228
	KM	0.001	0.00119	0.00162	0.00114	0.00011
	B	419.48	455.57	429.11	400.26	410.33
5000	ÜH	0.558	0.55806	0.55845	0.58277	0.58011
	KM	0.000	0.00026	0.00031	0.00019	0.00020
	B	316.40	166.20	380.72	100.47	200.358
10000	ÜH	0.54999	0.54810	0.55007	0.60993	0.59336
	KM	0.00032	0.00035	0.00032	0.00021	0.00029
	B	225.97	231.21	202.73	102.83	108.735

Çizelge 3.4 : Binh Problemi İçin Metrik Değerleri

P	Metrik	NSGA-II	OMOPSO	SPEA2	Yeni SPEA2	Yeni NSGA-II
1000	ÜH	0.684	0.684	0.684	0.727	0.694
	KM	0.0025	0.00234	0.00223	0.0022	0.00255
	B	799.10	649.92	1007.04	600.45	620.33
5000	ÜH	0.398	0.397	0.398	0.413	0.404
	KM	0.00088	0.00031	0.00088	0.00026	0.0003
	B	342.86	171.98	342.86	169.33	170.33
10000	ÜH	0.4684	0.467	0.468	0.498	0.4815
	KM	0.00016	0.00023	0.000157	0.00015	0.000154
	B	137.34	171.64	136.31	103.774	110.56

Çizelge 3.5 : WFG Problemi İçin Metrik Değerleri

P	Metrik	NSGA-II	OMOPSO	SPEA2	Geliştirilmiş SPEA2	Geliştirilmiş NSGA-II
1000	ÜH	0.485	0.484	0.485	0.524	0.5033818
	KM	0.00076	0.00089	0.00075	0.00023	0.0004529
	B	381.67	435.52	399.25	264.45	322.8846911
5000	ÜH	0.697	0.696	0.697	0.75	0.7344714
	KM	0.00147	0.00056	0.0016	0.0014	0.0013788
	B	989.24	346.74	867.94	345.77	300.335518
10000	ÜH	0.589	0.58	0.589	0.6388	0.602
	KM	0.00076	0.00064	0.0007	0.00054	0.00060
	B	899.65	568.71	854.79	439.75	692.55

Çizelge 3.6 : Tek Ürünlü Envanter Problemi İçin Metrik Değerleri

P	Metrik	NSGA-II	OMOPSO	SPEA2	Yeni SPEA2	Yeni NSGA-II
1000	ÜH	0.538	0.511	0.5518	0.592	0.569
	KM	0.0000019	0.0000019	0.000017	0.0000015	0.0000016
	B	1010.67	709.6279688	756.20	459.32	510.33
5000	ÜH	0.566	0.4005	0.616	0.663	0.638
	KM	0.0000011	0.0000017	0.0000017	0.0000011	0.0000011
	B	390.04	196.18	1757.63	237.46	389.75
10000	ÜH	0.680	0.486	0.529	0.562	0.590
	KM	0.000001	0.0000019	0.000014	0.000014	0.0000011
	B	654.038	239.16	105.532	104.44	99.225

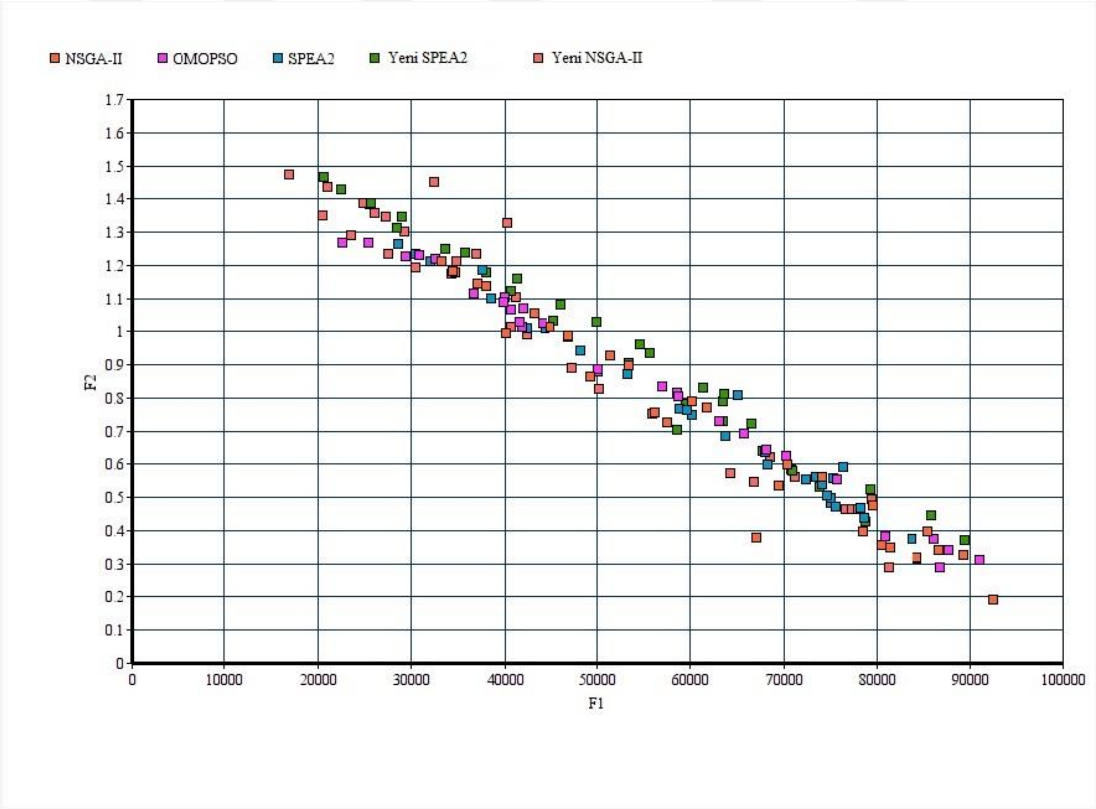
Çizelge 3.7 : Çok Ürünlü Envanter Problemi İçin Metrik Değerleri

P	Metrik	NSGA-II	OMOPSO	SPEA2	Yeni SPEA2	Yeni NSGA-II
1000	ÜH	0.498	0.482	0.50118	0.53689	0.5133072

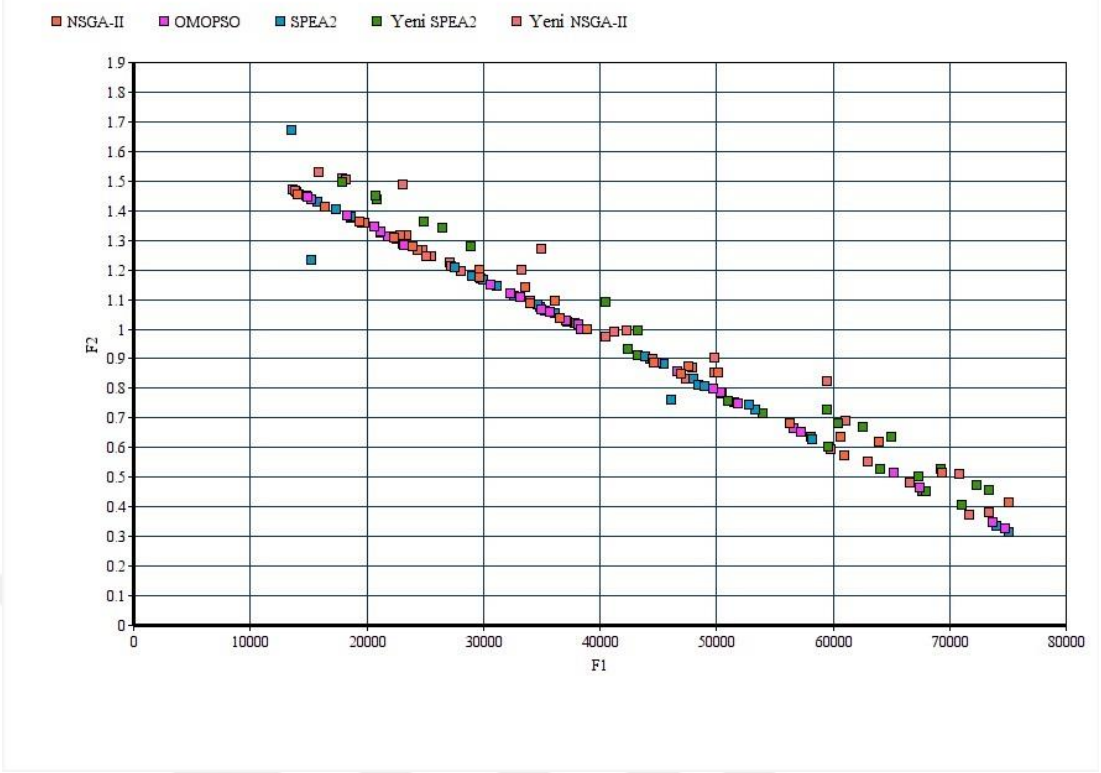
	KM	0.000012	0.000011	0.000010	0.000010	0.0000012
	B	1120.182	993.12	977.4591	699.3610	735.926134
	ÜH	0.528	0.5151	0.5189	0.529	0.528813
5000	KM	0.000012	0.000011	0.000010	0.000011	0.0000109
	B	911.46	1028.55	1082.310	762.9422	810.338420
	ÜH	0.558	0.5571	0.59152	0.639	0.607
10000	KM	0.000010	0.000011	0.000010	0.00001	0.00001
	B	730.174	289.1003	203.1667	202.44	210.66

3.7 Sonuçlar

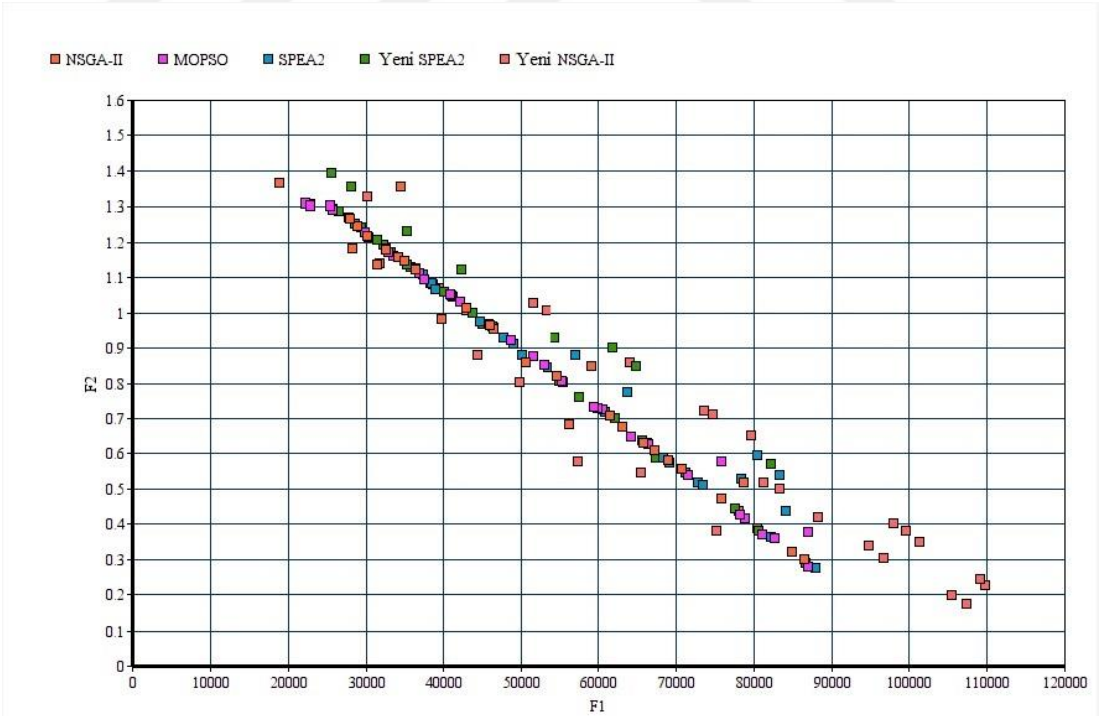
Problemlerin metrik değerleri **Çizelge 3.3**, **Çizelge 3.4**, **Çizelge 3.5, Çizelge 3.6** ve **Çizelge 3.7**'de verilmiştir.Tek ürünlü ve çok ürünlü envanter problemlerinin her popülasyon boyutu için 10'ar kez çözülmesiyle elde edilen birleşik Pareto cepheleri **Şekil 3.1**, **Şekil 3.2**, **Şekil 3.3**, **Şekil 3.4**, **Şekil 3.5** ve **Şekil 3.6**'da görülmektedir.



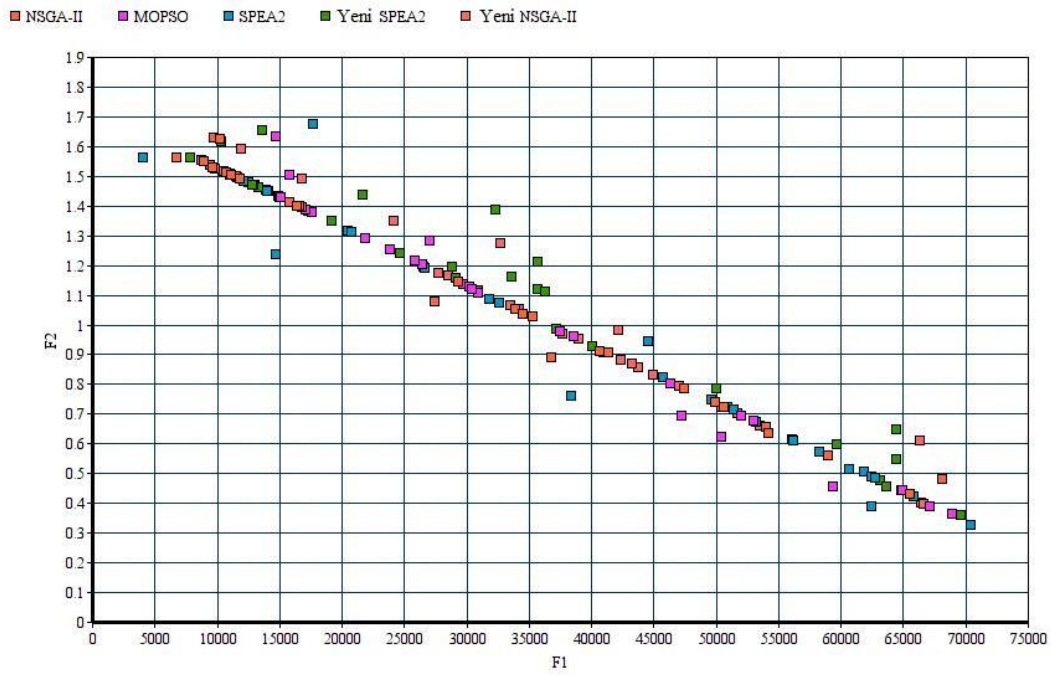
Şekil 3.1 : Tek Ürünlü Envanter Probleminin 1000 Popülasyon Boyutyla Elde Edilen Pareto Cephesi



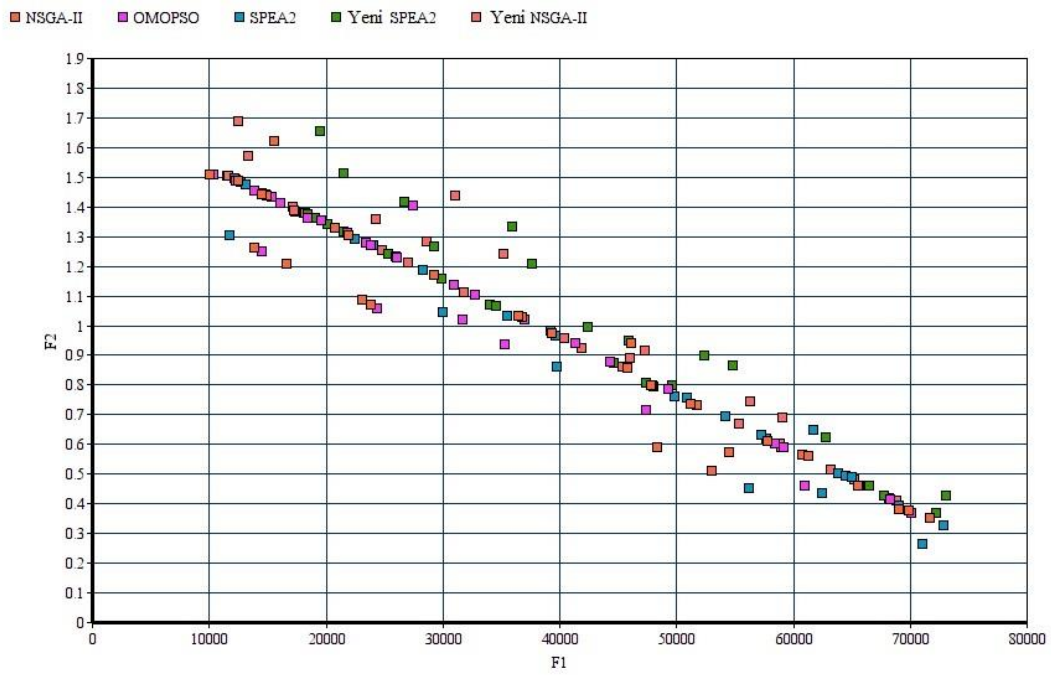
Şekil 3.2 : Tek Ürünlü Envanter Probleminin 5000 Popülasyon Boyutuyla Elde Edilen Pareto Cephesi



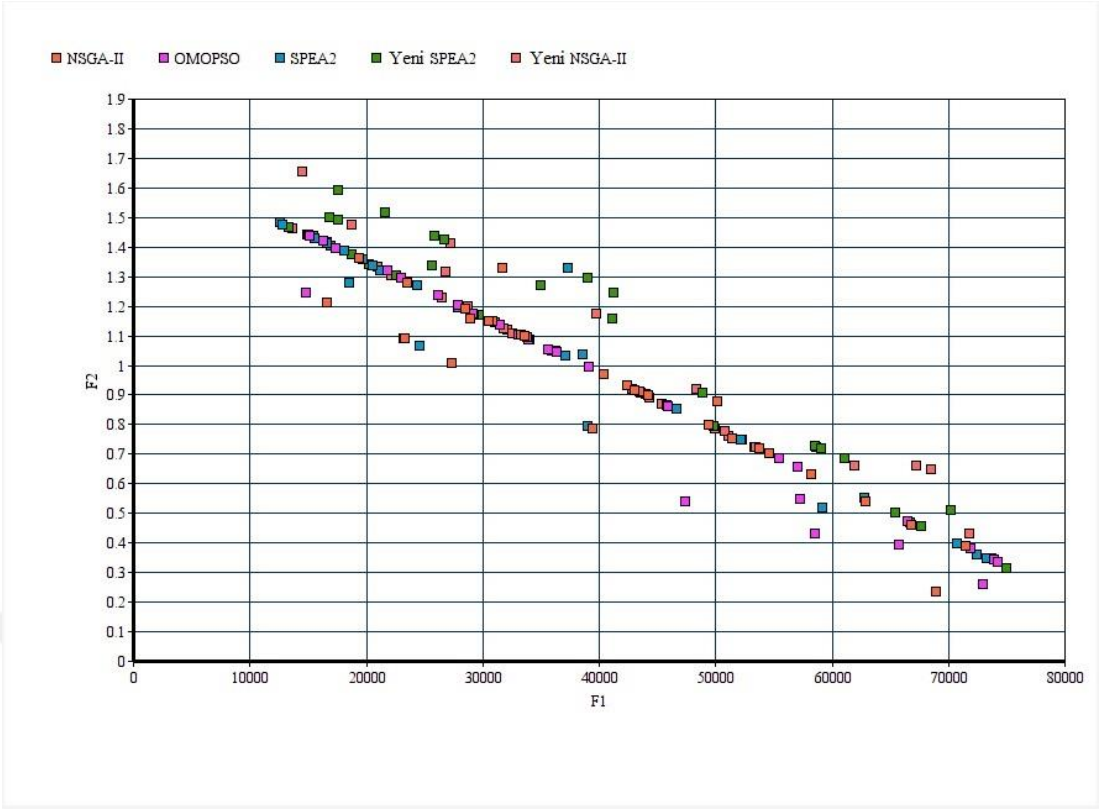
Şekil 3.3 : Tek Ürünlü Envanter Probleminin 10000 Popülasyon Boyutuyla Elde Edilen Pareto Cephesi



Şekil 3.4 : Çok Ürünlü Envanter Probleminin 1000 Popülasyon Boyutuyla Elde Edilen Pareto Cephesi



Şekil 3.5 : Çok Ürünlü Envanter Probleminin 5000 Popülasyon Boyutuyla Elde Edilen Pareto Cephesi



Şekil 3.6 : Çok Ürünlü Envanter Probleminin 10000 Popülasyon Boyutuyla Elde Edilen Pareto Cephesi

Elde edilen metrik değerlerinin toplamıyla tek yönlü ANOVA testi yapılmıştır. Sonuçlar **Çizelge 3.8** ve **Çizelge 3.9**'da görülmektedir.

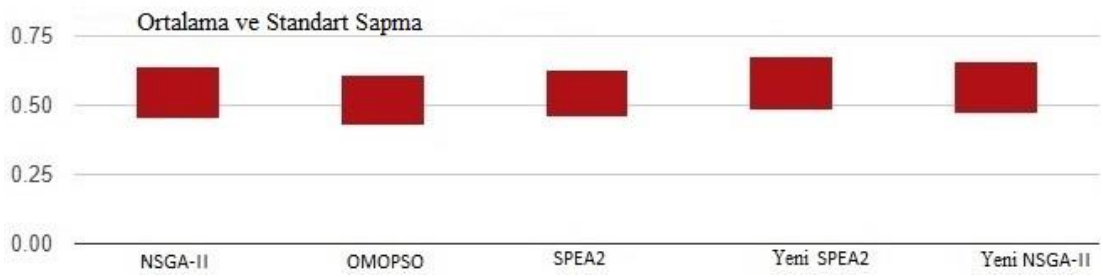
Çizelge 3.8 : Tek Yönlü ANOVA Testi Sonuçları

	N=15	NSGA-II	OMOPSO	SPEA2	Yeni SPEA2	Yeni NSGA-II
Üst Hacim	Ortalama	0.5465	0.5182	0.5424	0.5795	0.563
	SD	0.0925	0.0918	0.0881	0.098	0.0937
	SE	0.0239	0.0237	0.0227	0.0253	0.0242
Kuşaksal Mesafe	Ortalama	0.0006	0.0005	0.0006	0.0004	0.0004
	SD	0.0007	0.0006	0.0007	0.0007	0.0007
	SE	0.0002	0.0002	0.0002	0.0002	0.0002
Boşluk	Ortalama	621.8878	436.8853	633.5401	332.8734	379.5127
	SD	322.3466	293.7553	461.6456	222.3455	242.7304
	SE	83.2295	75.8473	119.1964	57.4094	62.6727

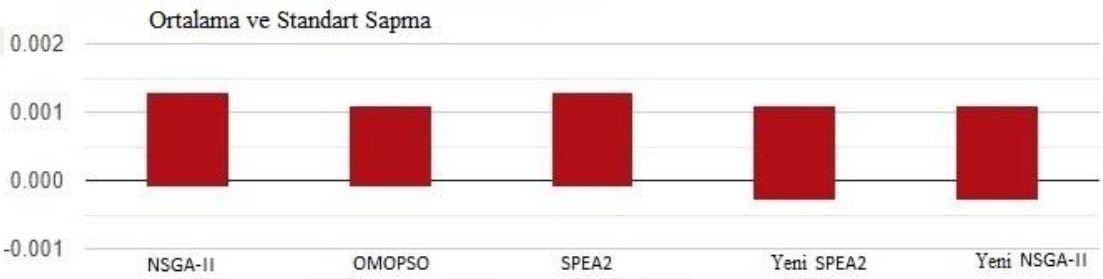
Çizelge 3.9 : Tek Yönlü ANOVA Testi Özeti

		Serbestlik Derecesi	Kareler Toplamı	Ortalama Karesi	F-Stat	P-Value
Üst Hacim	Gruplar Arası	4	0.0318	0.008	0.9219	0.4563
	Grup İçinde	70	0.6038	0.0086		
	Toplam:	74	0.6356			
Kuşaksal Mesafe	Gruplar Arası	4	0	0	0.3233	0.8615
	Grup İçinde	70	0	0		
	Toplam:	74	0			
Boşluk	Gruplar Arası	4	1159576.57	289894.1425	2.8328	0.0308
	Grup İçinde	70	7163404.294	102334.3471		
	Toplam:	74	8322980.864			

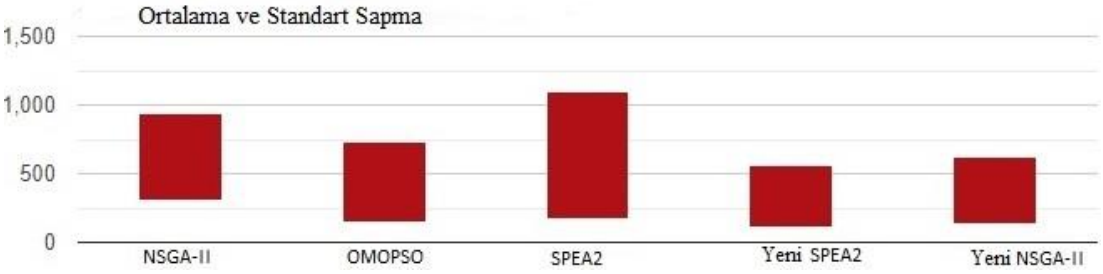
Şekil 3.7, **3.8** ve **3.9**'da her metrik için tek yönlü ANOVA testi sonuçları grafiksel biçimde gösterilmiştir.



Şekil 3.7 : Üst Hacim İçin Tek Yönlü ANOVA Testi Sonuçları



Şekil 3.8 : Kuşaksal Mesafe İçin Tek Yönlü ANOVA Testi Sonuçları



Şekil 3.9 : Boşluk İçin Tek Yönlü ANOVA Testi Sonuçları

4. GERÇEK VERİLER İLE ENVANTER OPTİMİZASYONU UYGULAMASI

4.1 Amaç

Büyük şirketlerin genelde kendi envanter yönetim politikaları vardır. Ayrıca çoğu büyük şirket gerek yapay zeka gerekse metasezgisel algoritmalar yardımıyla envanterlerini optimize etmektedir. Çoğu küçük ve orta boy işletmenin belirli bir envanter yönetim politikası yoktur ve envanter optimizasyon yöntemlerinden faydalanmamaktadır. Halbuki işletme ne kadar küçükte olsa, envanteri var olduğu sürece envanter optimizasyonundan yarar görebilmektedir. Bu sebeplerden dolayı bu bölümdeki deney orta boy, envanter yönetim politikası olmayan ve optimizasyon yöntemlerinden faydalanmayan bir işletme üzerinde yapılacaktır.

Önceki bölümlerdeki test işlemleri gerçek olmayan veri setleri üzerinde gerçekleşmiştir. Bu veri setleri üzerinde geliştirilmiş SPEA2 algoritmasının başarısı kanıtlanmıştır. Bu sebepten dolayı artık gerçek veriler üzerinde denenmeye hazırdır. Bu deneme sonrası geliştirilmiş SPEA2 algoritmasının kullanıcıya finansal kazanç sağlaması amaçlanmıştır.

Bu deney Nilgün Haciu-Ayşe Teyze'nin Çantaları isimli bir şahıs şirketinde yapılacaktır. Bu şirket el yapımı çantalar satmaktadır. Yapılan ürünler sezonluktur ve modası geçince satıştan kaldırılmaktadır. Genel olarak üç farklı hacimde üç farklı çanta modeli üretilmektedir. Ambar alanı kısıtlıdır. Şirket bir sezon için bu farklı çantaları herhangi bir kılavuz yöntem olmadan rassal miktarda üretmektedir. Şirketin bir atölyesi, satış yaptığı bir internet sitesi ve satış yaptığı fiziksel bir mağazası bulunmaktadır. Çantalar atölye de tek parti olarak üretilip fiziksel mağazaya gönderilmektedir. Bu sebepten dolayı envanter modeli tek aşamalı olacaktır. Fiziksel mağazada ve internet sitesinde satılacak her çanta fiziksel mağazada bulunmaktadır. Yani mağaza şirketin tek ambarı olarak iş görmektedir.

Bu deneyde şirketin envanter yöneticisinin bir sonraki sezon için beklentileri alınarak her tür çanta için üretilmesi gereken optimum sayı bulunacaktır. Yani optimal sipariş miktarı bulunacak ve işletme bu sayıyı gerçek hayatta uygulamaya koyacaktır.

Ayrıca şirketin SPEA2 yardımı olmadan her çantanın üretileceği varsayımsal miktarlar önceden kaydedilecektir. Bu sayılar ile geliştirilmiş SPEA2'nin bulduğu değerler ve gerçek satış miktarları karşılaştırılıp geliştirilmiş SPEA2 ile yapılan envanter optimizasyonun direkt finansal kazanca sebep olduğu kanıtlanacaktır. Ayrıca eğer en başta kaydedilen şirketin kılavuz olmadan üreteceği farazi çanta miktarları gerçek talep miktarıyla karşılaştırılacaktır. Bu karşılaştırmanın sonuçları ile geliştirimiş SPEA2 ile talep miktarı arasında yapılan karşılaştırmanın sonuçları, yine, birbirileriyle karşılaştırılacaktır.

4.2 Problem Modeli

Problemin amaç fonksiyonları karı ve servis seviyesini maksimize etmektir. Ambar kapasitesi kısıtlamalardır. Karar değişkeni sipariş miktarıdır. Üç farklı hacimde üç farklı ürün vardır. Talep tek biçimli sürekli bir dağılıma sahiptir. Modelde yer alacak değişkenlerin anlamları şunlardır:

i = 1, 2, 3....n için

ZK : Beklenen kar

ZD : Beklenen ambar doluluk oranı

D : Ambar kapasitesi

G : Beklenen gelir

Cb : Beklenen bulundurma maliyeti

Cs : Beklenen stoksuzluk maliyeti

Cm : Beklenen sipariş maliyeti

T : Ürün sayısı

P : Satış ücreti

Qi : i. ürünün sipariş miktarı

Xi : i. ürünün stokastik talebi

$Fxi(xi)$: i. ürünün talep probabilite fonksiyonu

λi : i. ürünün beklenen talep miktarı

μi : i. ürünün beklenen maksimum talep miktarı

πi : i. ürünün beklenen minimum talep miktarı

b : Bulundurma maliyetinin satış fiaytına olan yüzde oranı

s : Stoksuzluk maliyetinin satış fiyatına olan yüzde oranı

m : Sipariş maliyetinin satış fiyatına olan yüzde oranı.

Kar, bulundurma maliyetinin satış fiyatına yüzde oranı, stoksuzluk maliyetinin satış fiyatına yüzde oranı, bulundurma maliyetinin satış fiyatına yüzde oranı sıraysıyla **(4.1)**, **(4.2)**, **(4.3)** ve **(4.4)** eşitliklerinde verilmiştir. Bu eşitliklerden faydalanılarak eşitlik **(4.5)** ile kar hesaplanabailir. Eşitlik **(4.6)**'da servis seviyesi verilmiştir. Bu iki eşitliğe kısıtlamaların da eklenmesiyle eşitlik **(4.7)**'de envater modelinin tamamı hesaplanabilir.

$$G = \sum_{i=1}^{T} P_i X_i F x_i(x_i) \quad \textbf{(4.1)}$$

$$C_{sb} = \sum_{i=1}^{T} (X_i - Q_i) F x_i(x_i) P_i b \quad \textbf{(4.2)}$$

$$C_s = \sum_{i=1}^{T} (X_i - Q_i) F x_i(x_i) P_i s \quad \textbf{(4.3)}$$

$$C_m = \sum_{i=1}^{T} (Q_i - X_i) F x_i(x_i) P_i m \quad \textbf{(4.4)}$$

$$Z_k = G - C_m - max(C_b, C_s) \quad \textbf{(4.5)}$$

$$Z_d = Q/X \quad \textbf{(4.6)}$$

Maksimize et:

$$Z_k, Z_d \quad \textbf{(4.7)}$$

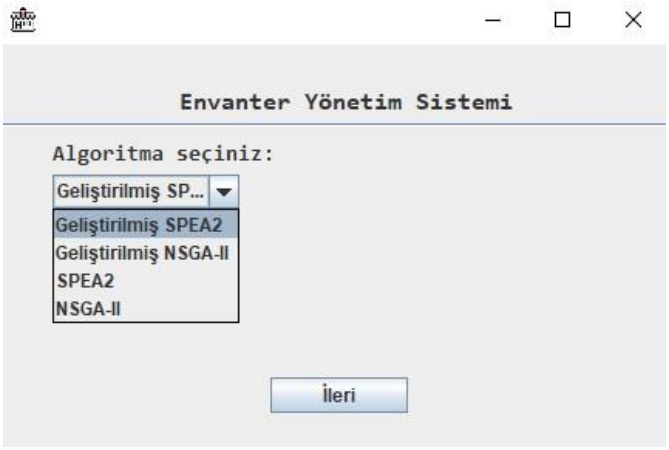
Kısıtlamalar:

$$D \leq 300$$

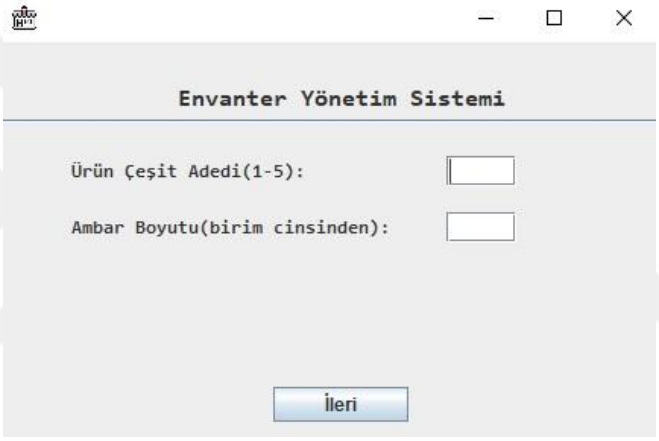
4.3 Deney Ortamı

Deneyi gerçekleştirmek için MOEA kütüphanesi kullaılarak, Java 8 dili ile bir masaüstü uygulaması kodlanmıştır (Hadka, 2019). Envanter yöneticisi her çanta modeli için minimum ve maksimum beklenen talebi, hacim ve fiyat bilgilerini girer. Ayrıca istenirse ambar boyutu da modifiye edilebilir. Programın çalışmasıyla,

geliştirilmiş SPEA2 algoritması Pareto sonuçlarını bulur ve rassal olarak kullanıcıya bir tanesini sunar. Programın arayüzü **Şekil 4.1**, **4.2**, **4.3** ve **4.4**'te görülmektedir.

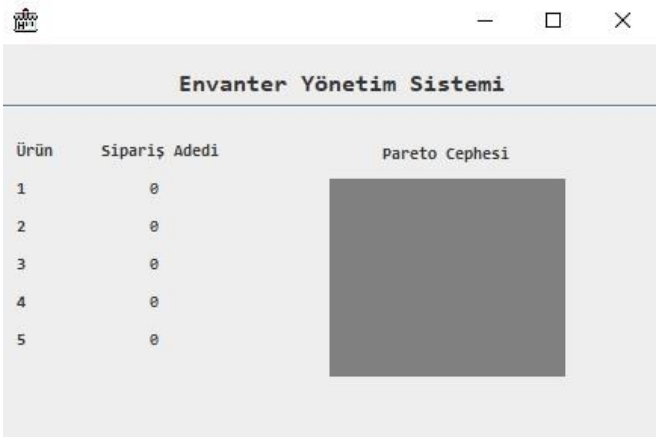


Şekil 4.1 : Geliştirilmiş SPEA2 ile Envanter Optimizasyon Uygulaması Arayüzü



Şekil 4.2 : Geliştirilmiş SPEA2 ile Envanter Optimizasyon Uygulaması Arayüzü

Şekil 4.3 : Geliştirilmiş SPEA2 ile Envanter Optimizasyon Uygulaması Arayüzü



Şekil 4.4 : Geliştirilmiş SPEA2 ile Envanter Optimizasyon Uygulaması Arayüzü

Envanter yöneticisinin beklentileri 5 Ocak 2020'de alınmıştır. Bu beklentiler 6 Ocak-19 Ocak 2020 arası satış sezonu içindir. Beklentiler **Çizelge 4.1**'de sunulmuştur. Ürünler hakkındaki diğer bilgiler **Çizelge 4.2**'dedir.

Çizelge 4.1 : Envanter Yöneticisinin Beklentileri

	Minimum Talep	Maksimum Talep
Sırt Çantası	1	300
Messenger Çanta	1	250
El Çantası	1	150

Çizelge 4.2 : Ürün Bilgileri

Ambar Seviyesi=300	1	2	3
P	160	110	100
s	0.3	0.3	0.3
m	0.3	0.3	0.3
b	0.2	0.2	0.2

4.4 Sonuçlar

Gerçek stokastik talep, geliştirilmiş SPEA2 algoritmasının bulduğu talep ve kullanıcının algoritma yardımı olmadan tahmin ettiği talep Çizelge 4.3'de görülmektedir.

Çizelge 4.3 : Gerçek Sonuçların, Geliştirilmiş SPEA2 Yardımıyla veya Yardımı Olmadan Elde Edilen Talep Tahminleriyle Karşılaştırılması

	Gerçek Talep	Optimize Tahmin	Optimize Olmayan Tahmin
Sırt Çantası	42	38	50
Messenger Çanta	21	23	25
El Çantası	73	84	50

Geliştirilmiş SPEA2 ile elde edilen kar ve geliştirilmiş SPEA2 olmadan elde edilmiş olacak kar **Çizelge 4.4**'de karşılaştırılmıştır.

Çizelge 4.4 : Kar Miktarlarının Karşılaştırması

	Toplam Kâr
Optimize Edilmeyen Envanter	4675 TL
Optimize Edilen Envanter	6993 TL

Geliştirilmiş SPEA2 algoritmasının kullanılmasıyla işletme 6993 TL kar etmiştir. Eğer geliştirilmiş SPEA2 algoritması kullanılmasaydı işletme 4675 TL kar edecekti. Yani bu optimizasyon yöntemin kullanarak işletme optimizasyon yapmadığı durumdan %49.583 daha fazla kar etmiştir. Bu günümüz ticaret şartlarına göre çok büyük bir orandır. Buradan envanter optimizasyonunun ne kadar önemli oduğu görülmektedir. Bu deneyin uygulandığı işletmenin orta boy bir işletme olması da dikkate değer ayrı bir konudur. Buradan anlaşılan, işletme ne boyutta olursa olsun envanteri olduğu sürece optimizasyon yöntemleri ile kar edebileceğidir.

Ayrıca kodlanan envanter uygulamasının çalıştırılması için, bir çok yapay zeka uygulamasında da olduğu için özelleştirilmiş ve güçlü bir donanım sistemine ihtiyaç duyulmamaktadır. Envanter uygulaması Intel i5-5200u işlemcisi, 16 GB RAM'i ve 512GB SSD'si olan bir dizüstü bilgisayarda çalıştırılmıştır. Çalışma süresi 42 saniye sürmüştür. Yani orta segmentte bir bilgisayar ile envanter yöneticisinin işlerini aksatmayacağı bir zaman süresinde envanter uygulaması sonuca ulaşmıştır. Bu sonuca ulaşmak için firma gerekse donanımsal olarak gerekse zaman olarak dikkate alınmayacak kadar az harcama yapmıştır. Bu sebeplerden dolayı uygulamanın yaygın olarak kullanılabilmesinin önü açıktır.

5. SONUÇ VE ÖNERİLER

5.1 Sonuç

Bu çalışma, en başarılı metasezgisel algoritmalar olan evrimsel algoritmalar üzerine olmuştur. Çalışmanın ilk bölümünde daha sonraki bölümlere konu olacak algoritmanın seçimi yapılmıştır. Bu seçim deneysel bir karşılaştırma ile gerçekleştrilmiştir. Karşılaştrıma sonucu envanter optimizasyonu için en iyi seçimin SPEA2 olduğu görülmüştür.

Sonraki bölümde, başarısı kanıtlanmış olan bu algoritma envanter optimizasyonu için daha da geliştirilmiştir. Karşılaştırma amacıyla SPEA2'nin en yakın rakibi olan NSGA-II algoritmasına da benzer geliştirmeler yapılmıştır. SPEA2'ye getririlen yeniliklerin başarılı olduğunu kanıtlamak amacıyla, bu algoritma geliştirilmiş NSGA-II, SPEA2, NSGA-II ve bir parçacık sürü algoritmasıyla karşılaştırılmıştır.

Karşılaştırma üç adet güvenilir test problemi ve biri çok diğeri tek ürünlü olan iki envater modeli çözümündeki performansları üzerinden yapılmıştır. Test problemleri gerçekte karşılaşılacak probleme benzer seçilmiştir. Performans ölçüm metrikleri de en kapsamlı biçimde seçilmiştir. Bütün problemlerin her versiyonunda her performans metriği için geliştirilmiş SPEA2 algoritması daha iyi performans göstermiştir.

Başarısı kanıtlanan geliştirilmiş SPEA2 algoritması bu tez çalışmasının en son bölümünde gerçek bir probleme uygulanmıştır. Uygulanan problem önceki bölümlerde çözülen problemlere benzemektedir. Problemin iki amaç fonksiyonu karı ve servis seviyesini maksimize etmektir. Depo alanında kısıtlama vardır ve sipariş miktarı karar değişkenidir. Yapılan deney sonunda probleme konu olan işletme büyük bir oranda kazanç sağlamıştır.

Bütün bu deneyler evrimsel algoritmaların, özellikle geliştirilmiş SPEA2 algoritmasının çok az bir zaman ve donanım gereksinimiyle işletmelerin önemli finansal kazanç elde etmesini sağlayacaklarını kanıtlamıştır. Bu sebepten dolayı envanter optimizasyonu olmayan her işletmenin evrimsel algoritmalarla yapılan envanter optimizasyon uygulamalarından yararlanmaları gerekmektedir.

5.2 Öneriler

Hem teoride hem de pratik birbirilerinden çok farklı envanter modeli bulunmaktadır. Her işletmenin ihtiyacına göre farklı amaçları, kısıtlamaları ve karar değişkenleri olabilir. Optimizasyon algoritmları her problem modelinde farklı performans göstermektedir. Bu sebeple tek bir algoritma en iyi algoritma ilan edilemez. Yani farklı modeller için farklı algoritmalar test edilebilir. Test edilecek algoritmalar seçilirken önce gerçek probleme benzeyen test problemleri üzerinde denenmelidirler. Yine de evrimsel algoritmalar ve özellikle SPEA ile NSGA ailesine ait algoritmalar çoğu problemde iyi performans göstermektedirler.

Optimizasyon algoritmalarının envanter problemi dışında da birçok uygulama alanı vardır. Öncelikle her tür yön eylem araştırmasında optimizasyon algoritmalarından faydalanılabilir. Tedarik zincirinin diğer halkalarında da optimizasyon algoritmaları kullanılabilir. Sistem ve ağ alanlarında anahtarlama, yönlendirici ayarları gibi alanlarda optimizasyon algoritmalarına başvurulabilir. İnsansız hava ve kara araçları filo yönetimi optimizasyon algoritmalarının kullanılabileceği başka bir alandır. Optimizasyon algoritmalarından yapay zeka alanında da faydalanılmaktadır. Özellik ve şema seçimi buna örnektir.

Bütün bunlar gözönüne alındığında geliştirilmiş SPEA2 başta olmak üzere evrimsel algoritmaların kullanım alanları ve popülaritesinin artacağı açıkça görülmektedir. Varolan evrimsel algoritmalarının gelişmeye açık olması ayrı bir avantajdır. Düşük maliyetle yüksek getiri sağlamaları ve kolay implementasyonları zaten kullanımı yaygın olan evrimsel algoritmaların hızla daha da yayılmasını sağlayacaktır.

KAYNAKLAR

Al-Hajri, M. T., ve Abido, M. A. (2011). Multiobjective optimal power flow using improved strength pareto evolutionary algorithm (spea2). In *2011 11th International Conference on Intelligent Systems Design and Applications* (Sf. 1097-1103).IEEE.

Alikar, N., Mousavi, S. M., Ghazilla, R. A. R., Tavana, M., ve Olugu, E. U. (2017). Application of the NSGA-II algorithm to a multi-period inventory-redundancy allocation problem in a series-parallel system. *Reliability Engineering & System Safety*, *160*, Sf. 1-10.

Azuma, R. M., Coelho, G. P., ve Von Zuben, F. J. (2011). Evolutionary multi-objective optimization for the vendor-managed inventory routing problem. In *2011 IEEE Congress of Evolutionary Computation (CEC)* (Sf. 1457-1464). IEEE.

Belgasmi, N., Said, L. B., ve Ghedira, K. (2011). Greedy local improvement of SPEA2 algorithm to solve the multiobjective capacitated transshipment problem. In *International Conference on Learning and Intelligent Optimization* (Sf. 364-378). Springer, Berlin, Heidelberg.

Bhesdadiya, R. H., Trivedi, I. N., Jangir, P., Jangir, N., ve Kumar, A. (2016). An NSGA-III algorithm for solving multi-objective economic/environmental dispatch problem. Cogent Engineering, 3(1), 1269383.

Bi, X., ve Wang, C. (2017). An improved NSGA-III algorithm based on objective space decomposition for many-objective optimization. Soft Computing, 21(15), Sf.4269-4296.

Binh, T. T., ve Korn, U. (1997). MOBES: A multiobjective evolution strategy for constrained optimization problems. In *The Third International Conference on Genetic Algorithms (Mendel 97)* (Cilt 25, Sf. 27).

Chen, S. P., ve Ho, Y. H. (2013). Optimal inventory policy for the fuzzy newsboy problem with quantity discounts. *Information Sciences*, *228*, Sf. 75-89.

Chołodowicz, E., ve Orłowski, P. (2017). Comparison of SPEA2 and NSGA-II applied to automatic inventory control system using hypervolume indicator. *Studies in Informatics and Control*, *26*(1), Sf. 67-74.

Coello, C. A. C., Lamont, G. B., ve Van Veldhuizen, D. A. (2007). *Evolutionary algorithms for solving multi-objective problems* (Cilt 5, Sf. 79-104). New York: Springer.

De Araújo, D. R., Bastos-Filho, C. J., ve Martins-Filho, J. F. (2015). An evolutionary approach with surrogate models and network science concepts to design optical networks. *Engineering Applications of Artificial Intelligence*, *43*, Sf. 67-80.

Deb, K., Agrawal, S., Pratap, A., ve Meyarivan, T. (2000). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *International conference on parallel problem solving from nature* (Sf. 849-858). Springer, Berlin, Heidelberg.

Deb, K., ve Jain, H. (2013). An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints. *IEEE transactions on evolutionary computation*, *18*(4), Sf. 577-601.

Emmerich, M. T., ve Deutz, A. H. (2018). A tutorial on multiobjective optimization: fundamentals and evolutionary methods. *Natural computing*, *17*(3), 585-609.

Fonseca, C. M., ve Fleming, P. J. (1995). An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary computation*, *3*(1), Sf. 1-16.

Hnaien, F., Dolgui, A., ve Wu, D. D. (2016). Single-period inventory model for one-level assembly system with stochastic lead times and demand. *International Journal of Production Research*, *54*(1), Sf. 186-203.

Hopp, W. J., Spearman, M. L., ve Zhang, R. Q. (1997). Easily implementable inventory control policies. *Operations Research*, *45*(3), Sf. 327-340.

Hosseini, S. V., Moghadasi, H., Noori, A. H., ve Royani, M. B. (2009). Newsboy problem with two objectives, fuzzy costs and total discount strategy. *Journal of Applied Sciences*, *9*(10), Sf. 1880-1888.

Huband, S., Hingston, P., Barone, L., ve While, L. (2006). A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation*, *10*(5), Sf. 477-506.

Kim, M., Hiroyasu, T., Miki, M., ve Watanabe, S. (2004). SPEA2+: Improving the performance of the strength Pareto evolutionary algorithm 2. In *International Conference on Paralle Problem Solving from Nature* (Sf. 742-751). Springer, Berlin, Heidelberg.

Kunkle, D. (2005). A summary and comparison of MOEA algorithms. In *Internal Report*. College of Computer and Information Science, Northeastern University.

Laumanns, M., Thiele, L., Deb, K., ve Zitzler, E. (2002). Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary computation*, *10*(3), Sf. 263-282.

Li, H., ve Zhang, Q. (2008). Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II. *IEEE transactions on evolutionary computation*, *13*(2), Sf. 284-302.

Maiti, M. K., ve Maiti, M. (2006). Fuzzy inventory model with two warehouses under possibility constraints. *Fuzzy Sets and systems*, *157*(1), Sf. 52-73.

Maheta, H. H., ve Dabhi, V. K. (2014). An improved SPEA2 Multi objective algorithm with nondominated elitism and Generational Crossover. In *2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT)* (Sf. 75-82). IEEE.

Mousavi, S. M., Hajipour, V., Niaki, S. T. A., ve Alikar, N. (2013). Optimizing multi-item multi-period inventory control system with discounted cash flow and inflation: two calibrated meta-heuristic algorithms. *Applied Mathematical Modelling*, *37*(4), Sf. 2241-2256.

Mousavi, S. M., Sadeghi, J., Niaki, S. T. A., ve Tavana, M. (2016). A bi-objective inventory optimization model under inflation and discount using tuned Pareto-based algorithms: NSGA-II, NRGA, and MOPSO. *Applied Soft Computing*, *43*, Sf. 57-72.

Panda, D., Kar, S., Maity, K., ve Maiti, M. (2008). A single period inventory model with imperfect production and stochastic demand under chance and imprecise constraints. *European Journal of Operational Research*, *188*(1), Sf. 121-139.

Pasandideh, S. H. R., Niaki, S. T. A., ve Keshavarzi, A. A. (2017). A two-echelon single-period inventory control problem with market strategies and customer satisfaction. *Journal of Uncertain Systems*, *11*(1), Sf. 18-34.,

Pasandideh, S. H. R., Niaki, S. T. A., ve Mousavi, S. M. (2013). Two metaheuristics to solve a multi-item multiperiod inventory control problem under storage constraint and discounts. *The International Journal of Advanced Manufacturing Technology*, *69*(5-8), Sf. 1671-1684.

Pasandideh, S. H., Niaki, S. T. A., ve Rashidi, R. (2011a). A two-echelon single-period inventory control problem under budget constraint. *The International Journal of Advanced Manufacturing Technology*, *56*(9-12), Sf. 1205-1214.

Pasandideh, S. H. R., Niaki, S. T. A., ve Tokhmehchi, N. (2011b). A parameter-tuned genetic algorithm to optimize two-echelon continuous review inventory systems. *Expert Systems with Applications*, *38*(9), Sf. 11708-11714.

Sanchez, D., Amodeo, L., ve Prins, C. (2010). Meta-heuristic approaches for multi-objective simulation-based optimization in supply chain inventory management. In *Artificial intelligence techniques for networked manufacturing enterprises management* (Sf. 249-269). Springer, London.

Schaffer, J. D. (1986). SOME EXPERIMENTS IN MACHINE LEARNING USING VECTOR EVALUATED GENETIC ALGORITHMS

(ARTIFICIAL INTELLIGENCE, OPTIMIZATION, ADAPTATION, PATTERN RECOGNITION).

Schott, J. R. (1995). *Fault tolerant design using single and multicriteria genetic algorithm optimization* (No. AFIT/CI/CIA-95-039). Air force inst of tech Wright Patterson afb OH.

Sharma, S. (2004). Optimal production policy with shelf life including shortages. *Journal of the Operational Research Society*, *55*(8), Sf. 902-909.

Seada, H., ve Deb, K. (2014). U-NSGA-III: A unified evolutionary algorithm for single, multiple, and many-objective optimization. COIN report, 2014022.

Sheng, W., Liu, Y., Meng, X., ve Zhang, T. (2012). An Improved Strength Pareto Evolutionary Algorithm 2 with application to the optimization of distributed generations. *Computers & Mathematics with Applications*, *64*(5), Sf. 944-955.

Silver, E. A. (2008). Inventory management: An overview, Canadian publications, practical applications and suggestions for future research. *INFOR: Information Systems and Operational Research*, *46*(1), Sf. 15-27.

Srinivas, N., ve Deb, K. (1994). Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation*, *2*(3), Sf. 221-248.

Srivastav, A., ve Agrawal, S. (2017). Multi-objective optimization of a mixture inventory system using a MOPSO–TOPSIS hybrid approach. *Transactions of the Institute of Measurement and Control*, *39*(4), Sf. 555-566

Taleizadeh, A. A., Akhavan Niaki, S. T., ve Hoseini, V. (2009). Optimizing the multi-product, multi-constraint, bi-objective newsboy problem with discount by a hybrid method of goal programming and genetic algorithm. *Engineering Optimization*, *41*(5), Sf. 437-457.

Van Veldhuizen, D. A. (1999). *Multiobjective evolutionary algorithms: classifications,analyses, and new innovations* (No. AFIT/DS/ENG/99-01). AIR FORCE INST OF TECH WRIGHT-PATTERSONAFB OH SCHOOL OF ENGINEERING

Whitley, D. (1994). A genetic algorithm tutorial. *Statistics and computing*, *4*(2), Sf. 65-85.

Wu, T. T., Geng, H. T., Yang, J. Y., ve Song, Q. X. (2009). An improved individual evaluation and elitism selection for distribution performance of SPEA2. In *2009 First International Conference on Information Science and Engineering* (Sf. 125-128). IEEE.

Yuan, Y., Xu, H., ve Wang, B. (2014). An improved NSGA-III procedure for evolutionary many-objective optimization. In Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation (Sf. 661-668).

Zhao, F., Lei, W., Ma, W., Liu, Y., ve Zhang, C. (2016). An improved SPEA2 algorithm with adaptive selection of evolutionary operators scheme

for multiobjective optimization problems. *Mathematical Problems in Engineering*, *2016*.

Zheng, Z. H., Ai, Q., Xu, W. H., HAN, L., JIANG, C. W., FENG, S. G., ve GU, C. H. (2009).Multi-objective load dispatch in wind power integrated system based on pseudo-parallel SPEA2 algorithm. *Journal of Shanghai Jiaotong University*, *43*(8), Sf. 1222-1227

Zhihuan, L., Yinhong, L., ve Xianzhong, D. (2010). Improved strength pareto evolutionary algorithm with local search strategies for optimal reactive powerflow. *Information Technology Journal*, *9*(4), Sf. 749-757.

Zitzler, E., Laumanns, M., ve Thiele, L. (2001). SPEA2: Improving the strength Pareto evolutionary algorithm. *TIK-report*, Sf. *103*

Zitzler, E., Laumanns, M., Thiele, L., Fonseca, C. M., ve da Fonseca, V. G. (2002). Why quality assessment of multiobjective optimizers is difficult. In *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation* (Sf. 666-674). Morgan Kaufmann Publishers Inc..

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., ve Da Fonseca, V. G. (2003). Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on evolutionary computation*, *7*(2), Sf. 117-132.

Zitzler, E., ve Thiele, L. (1998). Multiobjective optimization using evolutionary algorithms—a comparative case study. In *International conference on parallel problem solving from nature* (Sf. 292-301). Springer, Berlin, Heidelberg.

Zitzler, E., ve Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto evolutionary algorithm. *IEEE Trans. Evol. Comput. Cilt 3 i4*, Sf. 257-271.

Hadka, D. (2019, December 30). MOEA Framework. adres: http://moeaframework.org/

UNSD. (2019, Kasım 12). UNSD. (n.d.). adres: https://unstats.un.org/home/

Plotly. (2020, January 14). plotly/dash. adres: https://github.com/plotly/dash

EKLER

EK A: MOEA Kütüphanesinde Yer Alan SPEA2 Kaynak Kodu

EK B: MOEA Kütüphanesinde Yer Alan NSGA-II Kaynak Kodu

EK A:

```
public class SPEA2 extends AbstractEvolutionaryAlgorithm {

    /**
     * The selection operator.
     */
    private final Selection selection;

    /**
     * The variation operator.
     */
    private final Variation variation;

    /**
     * The number of offspring.
     */
    private final int numberOfOffspring;

    /**
     * Strength-based fitness evaluator.
     */
    protected final StrengthFitnessEvaluator fitnessEvaluator;

    /**
     * Compares solutions based on strength.
     */
    protected final FitnessComparator fitnessComparator;

    /**
     * Constructs a new instance of SPEA2.
     *
     * @param problem the problem
     * @param initialization the initialization procedure
     * @param variation the variation operator
```

```
     * @param numberOfOffspring the number of offspring generated each
iteration
     * @param k niching parameter specifying that crowding is computed using
     *       the {@code k}-th nearest neighbor, recommend {@code k=1}
     */
    public SPEA2(Problem problem, Initialization initialization,
                    Variation variation, int numberOfOffspring, int k) {
        super(problem, new Population(), null, initialization);
        this.variation = variation;
        this.numberOfOffspring = numberOfOffspring;

        fitnessEvaluator = new StrengthFitnessEvaluator(k);
        fitnessComparator = new
FitnessComparator(fitnessEvaluator.areLargerValuesPreferred());
        selection = new TournamentSelection(fitnessComparator);
    }

    @Override
    protected void initialize() {
        super.initialize();

        fitnessEvaluator.evaluate(population);
    }

    @Override
    protected void iterate() {
        // mating and selection to generate offspring
        Population offspring = new Population();
        int populationSize = population.size();

        while (offspring.size() < numberOfOffspring) {
            Solution[] parents = selection.select(variation.getArity(),
                    population);
            Solution[] children = variation.evolve(parents);
```

```
                offspring.addAll(children);
        }

        // evaluate the offspring
        evaluateAll(offspring);

        // evaluate the fitness of the population and offspring
        offspring.addAll(population);
        fitnessEvaluator.evaluate(offspring);

        // perform environmental selection to downselect the next population
        population.clear();
        population.addAll(truncate(offspring, populationSize));
}

/**
 * Returns the population of solutions that survive to the next generation.
 *
 * @param offspring all offspring solutions
 * @param size the number of solutions to retain
 * @return the population of solutions that survive to the next generation
 */
protected Population truncate(Population offspring, int size) {
        Population survivors = new Population();

        // add all non-dominated solutions with a fitness < 1
        Iterator<Solution> iterator = offspring.iterator();

        while (iterator.hasNext()) {
                Solution solution = iterator.next();
                double fitness = (Double)solution.getAttribute(
                                FitnessEvaluator.FITNESS_ATTRIBUTE);
```

```
                if (fitness < 1.0) {
                        survivors.add(solution);
                        iterator.remove();
                }
        }

        if (survivors.size() < size) {
                // fill remaining spaces with dominated solutions
                offspring.sort(fitnessComparator);

                while (survivors.size() < size) {
                        survivors.add(offspring.get(0));
                        offspring.remove(0);
                }
        } else if (survivors.size() > size) {
                // some of the survivors must be truncated
                MutableDistanceMap map = new
MutableDistanceMap(computeDistanceMatrix(survivors));

                while (survivors.size() > size) {
                        int index = map.findMostCrowdedPoint();

                        map.removePoint(index);
                        survivors.remove(index);
                }
        }

        return survivors;
}

/**
 * Computes the distance matrix containing the pair-wise distances between
 * solutions in objective space.  The diagonal will contain all 0's.
 *
```

```
     * @param population the population of solutions
     * @return the distance matrix
     */
    protected double[][] computeDistanceMatrix(Population population) {
        double[][] distances = new
double[population.size()][population.size()];

        for (int i = 0; i < population.size(); i++) {
            distances[i][i] = 0.0;

            for (int j = i+1; j < population.size(); j++) {
                distances[i][j] = distances[j][i] =

    IndicatorUtils.euclideanDistance(problem,
                                                    population.get(i),
population.get(j));
            }
        }

        return distances;
    }

    /**
     * Mapping of pair-wise distances between points.  This mapping is mutable,
     * allowing points to be removed.
     */
    public static class MutableDistanceMap {

        /**
         * The internal mapping of distances.
         */
        private List<List<Pair<Integer, Double>>> distanceMatrix;

        /**
```

```
     * Constructs a new mapping of pair-wise distances between points.
     *
     * @param rawDistanceMatrix the distance matrix
     */
    public MutableDistanceMap(double[][] rawDistanceMatrix) {
        super();
        initialize(rawDistanceMatrix);
    }

    /**
     * Initializes the internal data structures.
     *
     * @param rawDistanceMatrix the distance matrix
     */
    protected void initialize(double[][] rawDistanceMatrix) {
        distanceMatrix = new LinkedList<List<Pair<Integer,
Double>>>();

        for (int i = 0; i < rawDistanceMatrix.length; i++) {
            List<Pair<Integer, Double>> distances = new
LinkedList<Pair<Integer, Double>>();

            for (int j = 0; j < rawDistanceMatrix[i].length; j++) {
                if (i != j) {
                    distances.add(new Pair<Integer,
Double>(j, rawDistanceMatrix[i][j]));
                }
            }

            Collections.sort(distances, new
Comparator<Pair<Integer, Double>>() {

                @Override
                public int compare(Pair<Integer, Double> o1,
```

```
                                        Pair<Integer, Double> o2) {
                                    return  Double.compare(o1.getSecond(),
o2.getSecond());
                                }

                        });

                        distanceMatrix.add(distances);
                }
        }

        /**
         * Returns the most crowded point according to SPEA2's truncation
         * strategy.  The most crowded point is the point with the smallest
         * distance to its nearest neighbor.  Ties are broken by looking at
         * the next nearest neighbor repeatedly until a difference is found.
         *
         * @return the index of the most crowded point
         */
        public int findMostCrowdedPoint() {
                double minimumDistance = Double.POSITIVE_INFINITY;
                int minimumIndex = -1;

                for (int i = 0; i < distanceMatrix.size(); i++) {
                        List<Pair<Integer,      Double>>      distances      =
distanceMatrix.get(i);
                        Pair<Integer, Double> point = distances.get(0);

                        if (point.getSecond() < minimumDistance) {
                                minimumDistance = point.getSecond();
                                minimumIndex = i;
                        } else if (point.getSecond() == minimumDistance) {
                                for (int k = 0; k < distances.size(); k++) {
```

```
                                        double kdist1 =
distances.get(k).getSecond();
                                        double kdist2 =
distanceMatrix.get(minimumIndex).get(k).getSecond();

                                        if (kdist1 < kdist2) {
                                                minimumIndex = i;
                                                break;
                                        } else if (kdist2 < kdist1) {
                                                break;
                                        }
                                }
                        }
                }

                return minimumIndex;
        }

        /**
         * Removes the point with the given index.
         *
         * @param index the index to remove
         */
        public void removePoint(int index) {
                distanceMatrix.remove(index);

                for (List<Pair<Integer, Double>> distances : distanceMatrix) {
                        ListIterator<Pair<Integer, Double>> iterator =
distances.listIterator();

                        while (iterator.hasNext()) {
                                Pair<Integer, Double> point = iterator.next();

                                if (point.getFirst() == index) {
```

```
                                        iterator.remove();
                                    } else if (point.getFirst() > index) {
                                        // decrement the index so it stays aligned
with the
                                        // index in distanceMatrix
                                        iterator.set(new Pair<Integer, Double>(
                                                point.getFirst()-1,
point.getSecond()));
                                    }
                                }
                            }
                        }

    }

    /**
     * Fitness evaluator for the strength measure with crowding-based niching.
     */
    public class StrengthFitnessEvaluator implements FitnessEvaluator {

        /**
         * Crowding is based on the distance to the {@code k}-th nearest
         * neighbor.
         */
        private final int k;

        /**
         * Pareto dominance comparator.
         */
        private final DominanceComparator comparator;

        /**
         * Constructs a new fitness evaluator for computing the strength
         * measure with crowding-based niching.
```

```
     *
     * @param k crowding is based on the distance to the {@code k}-th
     *        nearest neighbor
     */
    public StrengthFitnessEvaluator(int k) {
        super();
        this.k = k;

        comparator = new ParetoDominanceComparator();
    }

    @Override
    public void evaluate(Population population) {
        int[] strength = new int[population.size()];
        double[] fitness = new double[population.size()];

        // count the number of individuals each solution dominates
        for (int i = 0; i < population.size()-1; i++) {
            for (int j = i+1; j < population.size(); j++) {
                int comparison = comparator.compare(population.get(i),
                        population.get(j));

                if (comparison < 0) {
                    strength[i]++;
                } else if (comparison > 0) {
                    strength[j]++;
                }
            }
        }

        // the raw fitness is the sum of the dominance counts (strength)
        // of all dominated solutions
        for (int i = 0; i < population.size()-1; i++) {
```

```
                        for (int j = i+1; j < population.size(); j++) {
                                int comparison =
comparator.compare(population.get(i),
                                                population.get(j));

                                if (comparison < 0) {
                                        fitness[j] += strength[i];
                                } else if (comparison > 0) {
                                        fitness[i] += strength[j];
                                }
                        }
                }

                // add density to the fitness
                double[][] distances = computeDistanceMatrix(population);
                KthSelector selector = new KthSelector();

                for (int i = 0; i < population.size(); i++) {
                        double kdist = selector.select(distances[i], null, k);
                        fitness[i] += 1.0 / (kdist + 2.0);
                }

                // assign fitness attribute to solutions
                for (int i = 0; i < population.size(); i++) {
                        population.get(i).setAttribute(FITNESS_ATTRIBUTE,
fitness[i]);
                }
        }

        @Override
        public boolean areLargerValuesPreferred() {
                return false;
        }
```

```
        }

    }
```

EK B:

```
public class NSGAII extends AbstractEvolutionaryAlgorithm implements
        EpsilonBoxEvolutionaryAlgorithm {

    /**
     * The selection operator.  If {@code null}, this algorithm uses binary
     * tournament selection without replacement, replicating the behavior of the
     * original NSGA-II implementation.
     */
    private final Selection selection;

    /**
     * The variation operator.
     */
    private final Variation variation;

    /**
     * Constructs the NSGA-II algorithm with the specified components.
     *
     * @param problem the problem being solved
     * @param population the population used to store solutions
     * @param archive the archive used to store the result; can be {@code null}
     * @param selection the selection operator
     * @param variation the variation operator
     * @param initialization the initialization method
     */
    public NSGAII(Problem problem, NondominatedSortingPopulation
population,
            EpsilonBoxDominanceArchive archive, Selection selection,
            Variation variation, Initialization initialization) {
        super(problem, population, archive, initialization);
        this.selection = selection;
        this.variation = variation;
```

```
	}

	@Override
	public void iterate() {
		NondominatedSortingPopulation population = getPopulation();
		EpsilonBoxDominanceArchive archive = getArchive();
		Population offspring = new Population();
		int populationSize = population.size();

		if (selection == null) {
			// recreate the original NSGA-II implementation using binary
			// tournament selection without replacement; this version works by
			// maintaining a pool of candidate parents.
			LinkedList<Solution> pool = new LinkedList<Solution>();

			DominanceComparator comparator = new ChainedComparator(
					new ParetoDominanceComparator(),
					new CrowdingComparator());

			while (offspring.size() < populationSize) {
				// ensure the pool has enough solutions
				while (pool.size() < 2*variation.getArity()) {
					List<Solution> poolAdditions = new ArrayList<Solution>();

					for (Solution solution : population) {
						poolAdditions.add(solution);
					}

					PRNG.shuffle(poolAdditions);
					pool.addAll(poolAdditions);
				}
```

```
                        // select the parents using a binary tournament
                        Solution[] parents = new Solution[variation.getArity()];

                        for (int i = 0; i < parents.length; i++) {
                            parents[i] =
TournamentSelection.binaryTournament(
                                    pool.removeFirst(),
                                    pool.removeFirst(),
                                    comparator);
                        }

                        // evolve the children
                        offspring.addAll(variation.evolve(parents));
                    }
                } else {
                    // run NSGA-II using selection with replacement; this version
allows
                    // using custom selection operators
                    while (offspring.size() < populationSize) {
                        Solution[] parents =
selection.select(variation.getArity(),
                                population);

                        offspring.addAll(variation.evolve(parents));
                    }
                }

                evaluateAll(offspring);

                if (archive != null) {
                    archive.addAll(offspring);
                }
```

```
            population.addAll(offspring);
            population.truncate(populationSize);
        }

        @Override
        public EpsilonBoxDominanceArchive getArchive() {
            return (EpsilonBoxDominanceArchive)super.getArchive();
        }

        @Override
        public NondominatedSortingPopulation getPopulation() {
            return (NondominatedSortingPopulation)super.getPopulation();
        }

}
```

ÖZGEÇMİŞ

Ad-Soyad **:** Ali Bayrakdar

Doğum Tarihi ve Yeri: 16.02.1985 Şişli

E-posta **:** bayrakdarali85@gmail.com

ÖĞRENİM DURUMU:

- **Lisans :** 2008, Yıldız Teknik Üniversitesi, Elektrik-Elektronik Fakültesi, Elektrik Mühendisliği
- **Yüksek Lisans:** 2020, İstanbul Aydın Üniversitesi, Bilgisayar Mühendisliği, Bilgisayar Mühendisliği

MESLEKİ DENEYİM VE ÖDÜLLER:

- 2018-Devam, Solenospys Yazılım Ltd. Şti., Yazılım Geliştirici
- 2013-2015, Tora Petrol A.Ş, Bakım Mühendisi
- 2011-2013, Boğaziçi Elektrik Dağıtım A.Ş, Bakım Koordinatörü
- 2008-2009, Un Ro-Ro, Elektrik Zabiti

TEZDEN TÜRETİLEN YAYINLAR, SUNUMLAR VE PATENTLER:

- Huseyinov, Ilham, and Ali Bayrakdar. "Performance Evaluation of NSGA-III and SPEA2 in Solving a Multi-Objective Single-Period Multi-Item Inventory Problem." 2019 4th International Conference on Computer Science and Engineering (UBMK). IEEE, 2019.