GU J Sci, Part C, 6(2): 358-373 (2018)

Gazi Üniversitesi

Fen Bilimleri Dergisi

PART C: TASARIM VE TEKNOLOJİ

dergipark.gov.tr/http-gujsc-gazi-edu-tr

Eşitlikçi Çok Amaçlı Sırt Çantası Problemi

Özlem KARSU [1, *]

[1] *Bilkent Üniversitesi, Mühendislik Fakültesi, Endüstri Mühedisliği Bölümü, 06800, Bilkent/ANKARA*

Makale Bilgisi

Başvuru: 05/12/2017
Düzeltme: 12/12/2017
Kabul: 18/01/2018

Anahtar Kelimeler

Çok amaçlı sırt çantası problemi
Eşitlikçi tercihler
Eşitlikçi Pareto çözümler
Dinamik programlama
Epsilon-kısıt yöntemi

Keywords

Multi-objective knapsack problem
Equitable preferences
Equitable efficiency
Dynamic programming
Epsilon-constraint approach

Öz

Bu çalışmada, eşitlikçi kaygıların olduğu kaynak dağıtımı problemi için kullanılabilecek, çok amaçlı matematiksel modelleme yaklaşımı geliştirilmiştir. Karar vericinin eşitlikçi tercih ilişkisine sahip olduğu varsayılmış ve eşitlikçi Pareto çözümler bulunması amaçlanmıştır. Eşitlikçi Pareto çözüm kümesinin bulunması için, problemdeki eşitlikçi kaygıları gözönüne alarak tasarlanmış, eşitlikçi Pareto çözümler vermeyecek durum vektörlerini alt ve üst sınırlar kullanarak eleyen, bir dinamik programlama algoritması önerilmiştir. Bu algoritmada, yazında önerilen alt sınırlara ek olarak yeni bir alt sınır mekanizması kullanılmış ve etkililiği gösterilmiştir. Dinamik programlama algoritması, epsilon kısıt yöntemi ile iki amaçlı problemler için karşılaştırılmıştır. Ayrıca, üç amaçlı problemler için epsilon kısıt yöntemi sonuçları verilmiştir.

Multiobjective Knapsack Problem with Equity Concerns

Abstract

In this paper, a multi-objective mathematical modeling approach has been developed for resource distribution problem which has equity concerns. We assume that the preference model of the decision maker satisfies properties related to inequity-aversion, hence we focus on finding nondominated solutions in line with the properties of inequity-averse preferences, namely the equitably efficient solutions. We propose a dynamic programming (DP) based algorithm, which exploits different lower and upper bounds to eliminate partial solutions that will not lead to equitably efficient solutions. In addition to the lower bounds previously discussed in the literature, we define a new lower bound and demonstrate its effectiveness. We perform experiments to show and discuss the performances of the DP algorithm and another well-known exact approach, the epsilon constraint method, for bi-objective settings. We also provide results of the epsilon constraint method for three-objective settings.

1. GİRİŞ (INTRODUCTION)

Günlük hayatta karşılaşılan pek çok kaynak dağıtımı probleminde, karar vericilerin kaynakları alıcılar arasında eşitlik gözeterek dağıtması istenmektedir. Özellikle son yıllarda yöneylem arastırması yazınında, çeşitli problemler ele alınırken eşitlik kaygıları da gözönünde bulundurulmaya başlanmıştır. Bunun sebebi, günlük hayatta bu tip kararların sadece verimliliği ya da toplam faydayı ençoklayan çözümü seçerek alınmadığının gözlenmiş olmasıdır. Sistemde bulunan alıcılar (örneğin bir kurumun hizmet verdiği müşteriler, bir firmada bulunan çeşitli departmanlar, bir ülkenin farklı coğrafi bölgelerinde yasayan insanlar) arasında eşitlik gözetmeden bulunan çözümler, çoğu durumda, paydaşlarca reddedilmekte ve uygulanmamaktadır.

Bu problemlere tipik bir örnek, farklı kategorilere ayrılabilen bir grup projeden, belirli bir bütçe dahilinde, hangilerinin destekleneceğine karar verme, yani proje seçimi (fonlama) problemleridir. Bu durumlarda karar vericiler, seçilecek proje portföyünü belirlerken, farklı kategorilere mümkün olduğunca eşit kaynak ayırmaya, ya da portföyün farklı kategorilerdeki faydalarını eşitlemeye çalışacak şekilde karar vermeye çalışabilirler. Bu kategoriler, bir organizasyon içindeki farklı departmanları, farklı sektörleri ya da farklı teknoloji alanlarını temsil edebilir.

*İletişim yazarı, e-mail: ozlemkarsu@bilkent.edu.tr

DOI:10.29109/http-gujsc-gazi-edu-tr.362369

Proje fonlama problemleri yöneylem araştırması literatüründeki bilinen problemlerden biri olan Sırt Çantası Problemi (Knapsack Problem) şeklinde modellenebilir. Eğer fonlama kararı 0-1 şeklindeyse yani projelerin kısmi fonlanması mümkün değilse bu problem kesikli sırt çantası problemi, projelerin kısmi olarak fonlanması mümkünse (örneğin, projenin talep ettiği kaynağın %75'ini karşılamak), problem sürekli sırt çantası problemidir. Bu çalışmada ele alınan problem bir kesikli sırt çantası problemidir.

Bu tip problemleri çözmek için literatürde genellike toplam faydayı ençoklayan (ya da bazı durumlarda maliyeti enazlayan) modeller ele alınmış, yukarıda bahsedilen türde bir denge kriteri çoğu zaman gözardı edilmiştir. Ancak, gerçek hayatta karar vericilerin kaynak dağılımında en sık gözönünde bulundurduğu kriterlerden biri de denge gözetmedir [8]. Bu iki kriter (toplam fayda ve denge) arasında çoğu zaman bir ödünleşme vardır. Yani toplam faydayı ençoklayan portföy dengesiz bir kaynak ya da fayda dağılımına yolaçabilir. Öte yandan dengeyi ençoklayan, bütün kategorilerin eşit miktarda kaynakla ya da eşit miktarda fayda gözlemlenecek şekilde fonlandığı bir portföy, toplam faydayı ençoklayan portföy olmayabilir. Burada karar verici, dengeli bir kaynak dağılımı isteyebileceği gibi, sadece faydaya odaklanıp dengeli bir fayda dağılımı da amaçlayabilir.

Bu çalışmada, bu tip kaynak dağıtımı problemlerinde denge ve toplam fayda kriterlerini aynı zamanda gözönüne alan, çok amaçlı matematiksel modelleme ve optimizasyon tabanlı bir karar destek sistemi geliştirilmiştir. Problem her bir kategoride gözlenmesi beklenen toplam faydayı (ya da istenirse her bir kategoriye ayrılan kaynağı) en çoklayan çok amaçlı bir model kullanılarak modellenmiştir, yani her amaç fonksiyonu bir kategoride gözlenecek faydayı en çoklamaktadır.

2. PROBLEM TANIMI VE İLGİLİ LİTERATÜR (PROBLEM DEFINITION AND LITERATURE REVIEW)

Bu çalışmada kesikli sırt çantası problemlerinde hem toplam fayda kriterini gözetecek hem de çeşitli kategorilerde elde edilen faydayı dengeli şekilde dağıtacak çok amaçlı optimizasyona dayalı bir çözüm yöntemi geliştirilmiştir. Ele alınan problem özel bir kesikli sırt çantası problemidir. Bu nedenle, literatür taraması, üça ana başlık altında tartışılacaktır: İlk olarak klasik (eşitlikçi kaygıların olmadığı, sadece toplam faydayı ençoklayan) sırt çantası problemlerinden bahsedilecek, sonrasında çok amaçlı kesikli sırt çantası konusunda yapılan çalışmalar tartışılacak ve en sonunda bu çalışmada ele alınan eşitlikçi sırt çantası problemi tanıtılacaktır.

2.1. Klasik kesikli sırt çantası problemleri ve uygulama alanları:

Sırt cantası problemi, yöneylem araştırması literatüründe en çok ele alınan problemlerden biridir. Klasik kesikli sırt çantası problemi, değerleri ve ağırlıkları verilmiş olan n tane parçadan hangilerinin sonlu kapasiteli bir sırt çantasına alınacağına karar verir. Amaç, verilen toplam kapasiteyi aşmadan alınan parçaların toplam değerini ençoklamaktır. Matematiksel modeli aşağıdaki gibidir. (Parçalar 1den n'ye kadar numaralandırılmıştır).

$$Max \sum_{i=1}^{n} v_i x_i$$

$$s.t. \sum_{i=1}^{n} c_i x_i \leq B$$

$$x_i \in \{0,1\}$$

Modelde x_i değişkeninin değeri bir çözümde i parçasının alınıp alınmadığını gösterir. v_i ve c_i değerleri modelin parametreleri olup, sırasıyla i numaralı parçanın değerini ve ağırlığını, B parametresi ise toplam kapasiteyi göstermektedir.

Klasik kesikli sırt çantası probleminin bir çok uygulama alanı mevcuttur. Bunlardan bazıları, yatırım bütçelemesi, kargo yükleme ve kesme problemleridir ([8], [14]). Bu çalışmada ele alınan proje, portföy seçme problemi bir sırt çantası problemidir. Hep parça bir projedir ve karar, bütçe dahilinde hangi projelerin fonlanacağıdır.

2.2. Çok amaçlı kesikli sırt cantası problemi ve çözüm yöntemleri

Model 1'de görüldüğü üzere klasik kesikli sırt çantası problemi her projenin tek tip bir değer getirdiğini varsayar. Bu problemin çok amaçlı olarak genellenmesi aşağıdaki modeli verir ([2], [4], [9]).

$$Max\, z_1 = \sum_{i=1}^{n} v_i^1 x_i$$

$$z_2 = \sum_{i=1}^{n} v_i^2 x_i$$

$$\ldots$$

$$z_p = \sum_{i=1}^{n} v_i^p x_i$$

$$s.t. \sum_{i=1}^{n} c_i x_i \le B$$

$$x_i \in \{0,1\}$$

Bu problemde bir projenin değeri ya da faydası tek bir boyut yerine birden fazla boyutta ölçülür, bu nedenle bir skalar yerine bir vektör ile gösterilir. Her projenin fonlanması halinde *j* amaç fonksiyonuna katkısı v_i^j ile gösterilir. Diğer parametre ve değişken tanımları model 1 ile aynıdır.

Yukardaki problemin çözümünden kasıt Pareto çözüm kümesinin bulunmasıdır. Tek amaçlı 0-1 sırt çantası problemi NP-tam karakterlidir, yani karar problemi NP-tam, eniyileme problemi NP-zordur. Benzer şekilde, çok amaçlı versiyonunda Pareto çözümleri bulmak da NP-tam karakterlidir [4].

Bu model, literatürde tek amaçlı modele göre daha az çalışılmıştır ve ele alınan modellerin çoğu iki kriterlidir. Bu modelin çözümü (Pareto çözüm kümesinin bulunması) için dal-sınırı algoritmaları ([17]), etiketleme algoritmaları [3], epsilon-kısıt yöntemi [12], [13] ve dinamik programlama (DP) algoritmaları ([2], [5], [16]) gibi kesin çözüm yöntemleri önerilmiştir.

Klamroth ve Wiecek [9] çok amaçlı tam sayılı sırt çantası modeline uygulanabilecek bazı dinamik programlama yöntemlerini (ilgili aşama ve durum tanımları ve yineleme formülleri ile birlikte) özetledikleri calışmalarında 0-1 sırt çantası problemini tamsayılı sırt çantası probleminin bir uzantısı olarak ele almış ve tam sayı problemi için geliştirilen DP algoritması tanımlarının bu problem için adapte edilebileceğini belirtmişlerdir. Ancak bu çalışmalarında sadece teorik bir tartışma yaparak aşama ve durum tanımlarını vermişler, herhangi bir karşılaştırmalı deneysel sonuç vermemişlerdir.

Bazgan vd. [2] yeni bir DP algoritması geliştirerek yöntemlerini daha önce önerilen kesin çözüm yöntemleri ile deneysel olarak karşılaştırmışlar ve algoritmalarının, o güne dek bilinen en verimli kesin çözüm yöntemi olan etiketleme algoritmasından ([3]) ve epsilon-kısıt yönteminden daha hızlı çalıştığını göstermişlerdir. Bu nedenle bu çalışmada Bazgan vd. tarafından önerilen aşama ve durum tanımları kullanmış ve algoritmaları eşitlikçi çözümler üretecek şekilde değiştirilmiştir. Tabii ki bu durum üst ve alt sınır (upper and lower bounds) tanımlarının ve kullandıkları filtreleme (kısmi çözüm/durum eleme) mekanizmalarının değişmesini gerektirmektedir. Bu konu 3. bölümde ayrıntılı olarak ele alınacaktır.

2.3. Eşitlikçi çok amaçlı sırt çantası problemleri, uygulama alanları ve çözüm yöntemleri

Kaynak dağıtımı problemlerinin çoğunda karar vericiler, verimliliğin yanısıra denge kaygıları da gözetmektedir [7]. Bu çalışmada da, karar vericinin faydayı eşitlikçi şekilde dağıtmak istediği ve her projenin talep ettiği kaynak miktarının ve proje gerçekleşirse oluşacak faydanın bilindiği varsayılmaktadır.

Eşitlikçi kaynak dağıtımı problemi aşağıdaki çok amaçlı kesikli sırt çantası modeli kullanılarak modellenebilir.

Model 1

$$Max\ z_1 = \sum_{i=1}^{n} v_i^1 x_i$$

$$Max\ z_2 = \sum_{i=1}^{n} v_i^2 x_i$$

...

$$Max\ z_p = \sum_{i=1}^{n} v_i^p x_i$$

$$s.t. \sum_{i=1}^{n} c_i x_i \leq B$$

$$x_i \in \{0,1\}$$

Bu problem n adet projeden, varolan bütçeyi (B) geçmeyecek şekilde hangilerine kaynak verileceğine karar verir. c_i değişkeni i nolu projenin kullanacağı kaynak miktarını, v_i^j ise i nolu projenin j kategorisinde sağlayacağı faydayı gösterir. Eğer projeler tek bir kategoriye aitse, tek bir kategoride fayda sağlarlar ve j kategorisine ait bir proje icin fayda vektörü $(0,0,..., v_i^j,...,0)$ şeklindedir.

Bu model, bölüm 2.2. de verilen model ile tamamen aynıdır ancak Model 1'in çözümünden kasıt, Pareto çözüm kümesi yerine eşitlikçi Pareto kümesinin bulunmasıdır. Problemlerin farkı aşağıda detaylandırılmıştır.

Model 1'de her bir amaç fonksiyonu bir kategoriye sağlanan faydayı ençoklar. Klasik çok amaçlı kesikli sırt çantası probleminden bir farkı her amaç fonsiyonunun skalasının aynı olmasıdır. Klasik bir çok amaçlı problemde, kullanılan kriterler (amaçlar) genellikle birbirinden farklı skalalarda ölçülürler. Örneğin iki kriterli (fiyat ve yakıt tüketimi) bir araba alım karar problemi düşünelim. Bir alternatif (300 000 lira, 5.6 l/km) şeklinde gösterilebilir. Eşitlikçi çok amaçlı sırt çantası probleminde ise kriterlerin hepsi tek tiptir ve aynı skalada ölçülebilir.

Problemin yazındaki klasik çok amaçlı problemlerden bir diğer farkı da, tercih ilişkisinin simetri özelliğidir, yani iki kategori arasında ayrım yapmayan bir karar verici için (3, 5) vektörü ile (5, 3) aynı derecede iyidir. Yazında, eşitlikçi kaygıların olduğu durumlarda odaklanılması gereken çözüm kümesinin eşitlikçi Pareto kümesi adı verilen bir çözüm kümesi olması gerektiği belirtilmektedir ([6], [10]) .

Dolayısıyla, eşitlikçi kesikli sırt çantası problemi, oluşturulan matematiksel modele klasik Pareto çözümler yerine eşitlikçi Pareto çözümler bulmayı amaçlar. Eşitlikçi Pareto çözüm kavramı aşağıda bir örnekle anlatılmaktadır.

Diyelim ki halk sağlığı iyileştirme projelerinin hangilerine kaynak ayrılacağına karar veren bir karar vericimiz var. Karar verici, toplumun, çocuk, genç-yetişkin ve yaşlı kesimlerine fayda getirmesi düşünülen 10 farklı proje önerisi değerlendiriliyor olsun. Bu projelerin talep ettikleri kaynak (maliyet) ve fayda miktarlarının Tablo 1'de görüldüğü gibi ve toplam bütçenin 135 birim olduğunu varsayalım.

Tablo 1. *Örnek problem verileri*

Proje	**Maliyet**	**Çocuk Fayda**	**Genç-Yetişkin Fayda**	**Yaşlı Fayda**
1	10	30	-	-
2	30	50	-	-
3	25	-	60	-

4	60	-	95	-
5	40	-	80	
6	20	-		30
7	15	-		45

Bu durumda olurlu bir çözüm 170 birim fayda sağlayan (1,1,1,0,0,1,0) dır. Bu çözümün çocuk, genç-yetiskin ve yaşlı segmentlerine sağladığı toplam fayda sırasıyla 80, 60 ve 30 birimdir yani fayda dağılım vektörü (80,60,30) dur. Başka bir olurlu çözüm (0,1,0,0,1,1,0) dir ve bu çözüme karşılık gelen toplam fayda 160 birim ve fayda dağılım vektörü (50,80,30) tir. Bu iki vektör birbirine klasik anlamda baskın değildir ancak eğer toplum grupları arasında fark gözetilmiyorsa, öneimli olan dağıtımın nasıl yapıldığıysa, birinci dağıtım ikinciye eşitlikçi baskındır. Dağıtım vektörleri sıralanmış hale getirilirse, ((30,50,80) ve (30,60,80)) bu baskınlık daha kolay görülür. Başka iki alternatifi ele alalım: (80,140,30) ve (80,95,75). Bu iki dağıtımın toplamda dağıttığı miktar aynıdır. Ancak ikinci dağıtım, birinci dağıtımda, görece iyi olan bir gruptan bir miktar faydanın görece kötü durumda olan bir gruba taşınmasıyla elde edilmiştir, dolayısıyla birinciye eşitlikçi baskındır.

Şimdi de sırasıyla (1,1,1,0,1,1,0) ve (1,1,1,0,0,1,1) çözümlerini düşünelim. İlgili dağıtım vektörleri (80,140, 30) ve (80, 60, 75) dir. Birinci çözüm toplam fayda konusunda daha iyidir ancak bu çözümde yaşlı kategorisi diğer iki kategoriye göre çok daha az fayda elde etmektedir ve bu durum eşitsizlik doğurmaktadır. İkinci çözüm ise toplam fayda konusunda daha kötüdür, ancak üç kategoriye birbirine yakın derecede fayda sağlayacağı için eşitlik açısından daha iyidir. Bu durumda bu iki çözümden biri diğerinden daha "iyi" değildir, yani biri diğerine baskın değildir ve ikisi de eşitlikçi Pareto çözümdür. Tercih, karar verici tarafından yapılacaktır ve farklı rasyonel karar vericiler farklı tercihler yapabilir. Bu çalışmada amaç eşitlikçi Pareto çözüm kümesinin bulunmasıdır. Aşağıda eşitlikçi Pareto çözümlerin tanımı verilmektedir.

Eşitlikçi Pareto Çözümleri

Bu bölümde eşitlikçi Pareto çözümlerin tanımı ve kullanılacak eşitlikçi tercih modelinin hangi aksiyomlara dayandığı anlatılacaktır.

Karar vericinin tercih modeli zayıf tercih ilişkisi (weak preference relation) $\underline{\pi}$ ile karakterize edilebilir ([11]). Örneğin iki alternatif x ve y'den x, y'ye zayıf olarak tercih ediliyorsa bu, $y \underline{\pi} x$ şeklinde gösterilir. Kesin tercih (π) ve farksızlık ($\approx$) ilişkisi zayıf tercih ilişkisi kullanılarak tanımlanabilir.

Literatürde, rasyonel bir karar vericinin tercih ilişkisinin aşağıdaki üç aksiyomu sağladığı varsayılmaktadır. Aşağıdaki tanımlarda $X \subseteq \Re^m$ alternatiflerin kümesi ve $x \in X$, $x=(x_1, x_2,..., x_p)$ tipik bir alternatifi göstermektedir. Her alternatif p boyutlu bir vektördür ve bizim problemimizde kaynağın p kullanıcıya dağıtımını göstermektedir. Amaç her kullanıcıya dağıtılan kaynağın ya da faydanın en çoklanmasıdır yani problem, çok kriterli bir problem olarak ele alınmaktadır.

1. Yansıma (Reflexivity) $x \underline{\pi} x$ bütün $x \in X$ için

2. Geçişlilik (Transitivity) $y \underline{\pi} x$ ve $z \underline{\pi} y$ ise $z \underline{\pi} x$

3. (Kesin) Monotonluk $x \underline{\pi} x+e_i\mathcal{E}$ bütün $x \in X$ için, e_i: i. elemanı 1, diğer elemanları 0 olan m boyutlu vektör, $\mathcal{E}$ küçük bir sayı.

Bu aksiyom diğer alıcılara sağlanan fayda azalmamak koşuluyla bir alıcıya dağıtılan faydanın arttığı bir dağıtımın daha çok tercih edilen bir dağıtım olacağını söylemektedir.

Bu üç aksiyom çok kriterli karar verme literatüründe klasik (yani simetrik olmayan) problemlerde rasyonel bir karar vericinin tercih ilişkisi için varsayılan aksiyomlardır. Eğer bir tercih ilişkisi bu aksiyomları sağlıyorsa Bir "rasyonel tercih ilişkisi"dir. Eğer bir x vektörü (alternatifi) bütün rasyonel tercih ilişkilerine göre y vektörüne tercih ediliyorsa, x, y'ye rasyonel olarak baskındır ve $y \underline{\pi}_r x$ şeklinde

gösterilir. Yani rasyonel baskınlık ilişkisi (rational dominance) bütün rasyonel tercih ilişkilerinin kesişim ilişkisidir.

Önerilen projede ele alınan problemlerde eşitlikçi kaygılar olduğu için yukarıdaki aksiyomlara aşağıda tanımlanan iki aksiyom daha eklenmektedir ([11]).

4. Simetri (Anonimlik): $x \approx \prod^i(x)$, bütün *i=1,2,...,m!* ve bütün $x \in X$ için. Burada $\prod^i(x)$, x vektörünün permütasyonunu göstermektedir. Bu aksiyom alıcılar arasında fark gözetilmemesini sağlamaktadır. Eğer karar vericinin tercih modeli bir değer fonksiyonu *u(.)* ile ifade edilebiliniyorsa, bu değer fonsiyonu simetrik olmalı ve bir vektörün bütün permütasyonlarına aynı değeri vermelidir.

5. Pigou Dalton Transfer Prensibi: Bütün $x \in X$ için, $x_i \leq x_j$ ise $x \underline{\pi} x+e_i\varepsilon- e_j\varepsilon$, $0 \leq \varepsilon \leq x_j - x_i$ için. Bu prensip gelir dağılımı eşitsizliğine odaklanan ekonomi literatüründe yaygın olarak kullanılan bir prensiptir. Bu prensip, kısaca, biri diğerinden daha çok fayda alan iki alıcıdan görece iyi durumda olandan biraz fayda alıp diğerine aktarılarak oluşturulan yeni dağıtımın daha çok tercih edilmesi gerektiğini belirtmektedir.

Eğer bir tercih ilişkisi yukarıda sayılan beş aksiyomu sağlıyorsa Bir "eşitlikçi rasyonel tercih ilişkisi"dir. Eğer bir x vektörü (alternatifi) bütün eşitlikçi rasyonel tercih ilişkilerine göre y vektörüne tercih ediliyorsa, x, y'ye eşitlikçi rasyonel olarak baskındır ve $y \underline{\pi}_e x$ şeklinde gösterilir. Yani eşitlikçi (rasyonel) baskınlık ilişkisi (equitable dominance) bütün eşitlikçi rasyonel tercih ilişkilerinin kesişim ilişkisidir.

Tanım: Bir olurlu çözüm, eğer kendisine eşitlikçi baskın başka bir olurlu çözüm yoksa eşitlikçi Paretodur.

Bu çalışmada karar vericinin eşitlikçi rasyonel bir tercih ilişkisine sahip olduğu varsayılmıştır. Ancak eşitlikçi rasyonel tercih ilişkileri kümesinde hangi spesifik tercih ilişkisine sahip olduğu bilinmemektedir ve bu yönde başka kısıtlayıcı bir varsayım yapılmayacaktır. Dolayısıyla bütün eşitlikçi Pareto çözümler bulunacak ve karar vericiye sunulacaktır. Eşitlikçi Pareto çözümler kavramı literatürde oldukça yeni bir kavramdır. Eşitlikçi tercihlerin cok kriterli karar verme literatüründe ilk ele alan makale Kostreva ve Ogryczak [10] tarafından yazılmıştır. Bu makalede eşitlikçi rasyonel tercih ilişkisi ve eşitlikçi baskınlık ilişkisinin tanımı yapılmış ve aşağıdaki teorem ispatlanmıştır.

Teorem 1: $y \underline{\pi} e \;\; x \Leftrightarrow \sum_{i=1}^{k} \overset{\rho}{y}_i \leq \sum_{i=1}^{k} \overset{\rho}{x}_i \;\; \forall k = 1,2,...p$ (Burada $\overset{\rho}{x}$ sembolü x vektörünün elemanları küçükten büyüğe sıralanmış sıralı permütasyonunu göstermektedir.)

Daha sonra Kostreva vd. [11] çok amaçlı karar verme literatüründe kullanılan toplama/bütünleştirme fonksiyonu kavramının eşitlik gözeten durumlarda hangi özelliklere sahip olması gerektiğini listelemişler ve eşitlikçi bir toplama fonksiyonun Schur-konkav olması gerektiğini belirtip kullanılabilecek fonksiyonlara örnekler vermişlerdir.

Bataar ve Wiecek [1], genel bir çok amaçlı programlama modelinde eşitlikçi Pareto çözüm kümesinin bulunması için iki tek amaçlı doğrusal olmayan program kullanan iki aşamalı bir yöntem geliştirmişlerdir. Çalışmaları sadece yöntemin açıklamasına dayalıdır ve herhangi bir deneysel sonuç verilmemiştir.

Ogryczak vd. [15] eşitlikçi bant genişliği dağıtımı optimizasyonu problemini ele almış ve referans noktası yöntemi ile eşitlikçi çözümler bulmuşlardır. Bu çalışmada çözümü kolaylaştırması açısından bütün kriterler yerine sınırlı bir kriter kümesi gözönüne alınmıştır.

Toplam faydayı ençoklayan çalışmalar oldukça çok olmakla birlikte dengeyi de gözeterek çözüm öneren yaklaşımlar literatürde yenidir. Daha önce denge kriteri farklı fonksiyonlar kullanılarak matematiksel modellerde kullanılmış ancak bu çalışmada ele alındığı şekilde eşitlikçi çok amaçlı sırt çantası problemleri için özel çözüm yöntemleri üzerinde çalışılmamıştır. Dinamik programlama, daha önce klasik çok amaçlı sırt çantası problemlerinde Pareto çözüm kümesi bulmak için önerilmiş ancak simetrinin olduğu durumlarda eşitlikçi Pareto çözüm kümesi bulunması için kullanılmamıştır. Bu çalışma, dinamik programlama yöntemini böyle durumlar için uyarlayarak kullanması açısından yazına katkı sağlayacaktır. Ayrıca DP yöntemi, başka bir çözüm yöntemi olan epsilon-kısıt yöntemi ile karşılaştırılmakta ve sonuçlar sunulmaktadır.

3. ÇÖZÜM YÖNTEMLERİ (SOLUTION METHODS)

1. Dinamik Programlama

Dinamik programlama, literatürde pek çok optimizasyon probleminin çözümü için kullanılan yöntemlerden biridir. Çoğunlukla, büyük bir problemi küçük problemlere bölerek ve problemin sonundan başına giderek çözüm elde eder. Dinamik programlama yönteminde problem, her aşamada bir karar verilecek şekilde aşamalara bölünür. Her aşamanın durumları vardır. Durum, herhangi bir aşamadaki en iyi kararı verebilmek için gerekli olan bilgiyi verir. Her aşamada verilen karar sonunda, mevcut aşamadaki durumdan bir sonraki aşamadaki bir duruma gidilir. Mevcut durum verildiğinde, kalan aşamaların her biri için en iyi karar daha önceden bulunulan durumlara ya da verilen kararlara bağlı değildir. Problem durumları aşamalara ayrıldığında, $t,t+1,...,T$ aşamalarında kazanılan ödül ya da gözlenen maliyeti $t+1,t+2,...,T$ aşamalarına bağlayan bir yineleme vardır.

Projede ele alınan çok amaçlı kesikli sırt çantası probleminde kullanılacak aşama ve durum tanımları aşağıdaki gibidir. Problem N aşamaya bölünmüştür. Her k aşaması, her biri ilk k madde (proje) ile oluşturulabilecek olurlu çözümleri gösteren durumlardan oluşur. k aşamasındaki (S_k) bir s durum vektörü $p+1$ boyutlu bir vektördür $s=(s^1, s^2,..., s^p,c)$. s^j, j amaç fonksiyonun değerini (j kategorisindeki toplam faydayı) ve c bu kısmi çözümde şimdiye kadar kullanılan toplam kaynağı gösterir. Algoritma k aşamasındaki durumları aşağıdaki yinelemeyi kullanarak oluşturur:

$$S_k := \text{Dom}\ \{\ T_k := S_{k-1} \cup (s^1 + v_k^1,..., s^p + v_k^p, c + c_k):\ c + c_k \le B, s \in S_{k-1}\}$$

$$S_0 = \mathbf{0}.$$

Bu yineleme formülasyonunda T_k kümesi iki kümenin birleşiminden oluşur : Birincisi daha önceki aşamadaki durumlar (S_{k-1}), ikincisi ise bu durumlardaki kısmi çözümlere, eğer olurlu ise, k maddesini ekleyerek elde edilen yeni durumlardır. T_k deki durumlardan sadece baskın olanlar k aşamasını oluşturur (Dom (T_k) durumları bu şekilde filtreleyerek baskın olanları aldığımızı göstermektedir). s durumu aşağıdaki durumlardan biri olursa baskın değildir ve elenir.

Durum eleme (filtreleme) kuralları:

$s \in S_{k-1}$ olsun. Eğer $\sum_{i=k}^{N} c_i \le B - c$ ise, yani kalan bütçe geri kalan bütün projelere yetiyorsa, olabilecek en iyi çözüm bilinmektedir. $s' \in S_k : s' = (s^1 + v_k^1,..., s^p + v_k^p)$ s'ye baskındır ve s elenir.

$k<N$ için, eğer $\exists s' : (s^1, s^2,..., s^p) \underline{\pi}_r (s'^1, s'^2,..., s'^p)$ ve $c' \le c$ ise s elenir. Bu durumda s çözümünden oluşturulabilecek her olurlu tam çözüm, s' dan yola çıkarak da oluşturulabilir. s den den yola çıkarak oluşturulan en iyi çözüm, s' ne uygulandığında daha kötü bir çözüm oluşturması mümkün değildir.

s için bir üst sınır $UB(s)$ ve s' için de bir alt sınır $LB(s')$ varsa ve $UB(s)\ \underline{\pi}_e\ LB(s')$ ise s elenir.

Bu kurallardan ilk ikisi daha önce iki (ve üç) kriterli klasik sırt çantası problemleri için [2] tarafından kullanılmıştır. Üçüncü kural, yazarların kullandığı üçüncü kuralın eşitlikçi sırt çantası problemlerine uyarlanmış halidir. Kural, rasyonel baskınlık yerine eşitlikçi baskınlık kullanılarak güçlendirilmiştir. Belirtmek isteriz ki her ne kadar [2]'de tanımlanan 3. kural bu şekilde güçlendirilmişse de, 2. kuralda rasyonel baskınlık yerine eşitlikçi baskınlık kullanmak mümkün değildir. Bu durum, aşağıda gösterilmiştir.

Not 4 Herhangi bir durum vektörü s, kendisine eşitlikçi baskın olan ve şu ana kadar daha az maliyete sahip olan bir s' olduğu gösterilerek elenemez ($k=N$ değilse).

İspat: $m = 2$ olsun. k ($k < N$) aşamasında iki durum vektörü düşünelim: s = (5; 8; 3)

ve s' = (6; 7; 2). $s\ \underline{\pi}\ e\ s'$ sağlanıyor ve s' şu ana kadar daha az bütçe kullanmış. Varsayalım ki, bu çözümlerden elde edilebilecek tek tam çözüm, ikisine de fayda değerleri ve maliyeti (3,5,1) olan bir proje

eklemekle mümkün olsun. Bu durumda (8.5,9) ve (9.5,8) tam çözümleri elde edilecektir ve bu durumda *s* den elde edilen tam çözüm daha iyidir (diğerine eşitlikçi baskındır). Dolayısıyla *s* nin elenmesi yanlış olacaktır.

Tasarlanan DP algoritması [2] de klasik sırt çantası problemleri için kullanılan algoritmanın ilgili eşitlikçi probleme uyarlanmış halidir. Verilen her S_{k-1} için yukarıda tanımlanan eleme kuralları kullanılarak S_k hesaplanarak en sonunda S_N elde edilir. Algoritmanın detayları Ek 1'de verilmiştir.

Algoritma detayları ve kurallar p=2 durumu için açıklanmıştır. Önerilen yaklaşım kolaylıkla $p>3$ olan genel problemlere uyarlanabilir, ancak bu durumlarda bellek gereksiniminin artması beklenmektedir. 3. eleme kuralında kullanılan üst ve alt sınırlar aşağıda açıklanmaktadır.

Üst sınır

Herhangi bir durum vektörü $s=(s^1, s^2, c)$ için üst sınır vektörü UB_1, UB_2 aşağıdaki adımlarla hesaplanmıştır.

Adım 1. Henüz hakkında karar verilmemiş projeleri (k aşamasında bu projeler indeksleri k dan büyük olan projelerdir) v_i^T / c_i oranına göre büyükten küçüğe diz ($v_i^T = v_i^1 + v_i^2$). Bu projeleri 1 den N-k ya $v_1^T / c_1 \geq v_2^T / c_2 \geq ... \geq v_{N-k}^T / c_{N-k}$ olacak şekilde tekrar numaralandır.

Adım 2. Projeleri,bütçe elverdiği sürece birinciden (v_i^T / c_i oranı en yüksek olandan) başlayarak ekle. Yani $n: \sum_{i=1}^{n-1} c_i \leq B - c$ ve $\sum_{i=1}^{n} c_i > B - c$ yi sağlayan n değerini bul.

Adım 3. Kalan bütçe rc olsun.

$$UB_T = s^1 + s^2 + \sum_{i=1}^{n-1} v_i^T + \max\left\{ rc\frac{v_{n+1}^T}{c_{n+1}}, v_n^T - (c_n - rc)\frac{v_{n-1}^T}{c_{n-1}} \right\}$$

Adım 4. $UB_1 = UB_2 = UB_T/2$.

Teorem 2: *(UB_1,UB_2), s* durum vektöründen elde edilebilecek tüm eşitlikçi Pareto çözümler için bir üst sınırdır.

İspat: *s* den elde edilebilecek herhangi bir olurlu tam çözümdeki toplam çıktı miktarı UB_T' den büyük olamaz ([15], Bölüm 2.3.). Diyelim ki elimizde *s* den elde edilen olurlu bir tam çözüm olsun $\left(LB_1^{'}, LB_2^{'}\right)$ .Bu vektörün *(UB_1,UB_2)* ye göre eşitlikçi baskınlık açısından kötü olmaması için aşağıdakilerden en az biri sağlanmalıdır.

$$Min\left\{LB_1^{'}, LB_2^{'}\right\} > Min\left\{UB_1, UB_2\right\}$$
$$LB_1^{'} + LB_2^{'} > UB_T$$

UB_T nin elde ediliş biçiminden biliyoruz ki $LB_1^{'} + LB_2^{'} \leq UB_T$, dolayısıyla ikinci koşul sağlanmamaktadır. Birinci koşulun sağlanması için $Min\left\{LB_1^{'}, LB_2^{'}\right\} > UB_T / 2$ olması gerekir ancak bu da yine $LB_1^{'} + LB_2^{'} \leq UB_T$ ile çelişmektedir.

Alt sınırlar

Herhangi bir durum vektörü $s=(s^1, s^2, c)$ için aşağıdaki yöntemler kullanılarak alt sınırlar elde edilecektir.

1. Her j kategorisi için, kalan projeleri v_i^j / c_i oranına göre azalmayacak şekilde diz. p=2 olduğu durumda, bu diziliş her projeye iki sıra verir: r_1, r_2. $r_{Maks}=Maks\{r_1,r_2\}+0.5/N(r_1+r_2)$. Projeleri r_{Maks} değerlerine göre azalmayacak şekilde yeniden sırala. Bütçe yettiği sürece sıralı listeden proje ekle.

2. Bu yöntem, 1. yönteme çok benzer. Tek fark kalan projelerin r_{Maks} yerine $r_{Tot}= r_1+r_2$ a göre sıralanmasıdır.

3. Kalan projeleri v_i^T / c_i oranına göre azalmayacak şekilde diz. En yüksek orana sahip olan projeden başlayarak, bütçe yettiği sürece projeleri ekle.

4. Her j kategorisi için, kalan projeleri v_i^j / c_i oranına göre azalmayacak şekilde diz. s durum vektöründeki minimum toplam çıktıya sahip olan kategoriyi bul. Genelliği kaybetmeden bu kategorinin 1. kategori olduğunu varsayalım. Kalan projeleri v_i^1 / c_i oranı en büyük olandan başlayarak teker teker ekle. Her eklemeden sonra kategori 1'in hala en az çıktıya sahip kategori olup olmadığını kontrol et. Öyleyse, eklemeye devam et. Değilse, henüz eklenmemiş projeleri, v_i^2 / c_i oranı en büyük olan başlayarak ekle. Her eklemeden sonra kategori 2'in hala en az çıktıya sahip kategori olup olmadığını kontrol et. Öyleyse, eklemeye devam et. Değilse, henüz eklenmemiş projeleri, v_i^1 / c_i oranı en büyük olan başlayarak ekle. Bu döngü, henüz eklenmemiş projelerdeki minimum maliyet, kalan bütçeyi aşana kadar devam eder.

3. Epsilon-kısıt yöntemi

Epsilon-kısıt yöntemi, literatürde çok (özellikle iki) amaçlı problemlerin çözümü için oldukça yaygın kullanılan bir yöntemdir. Her seferinde, amaçlardan birinin eniyilendiği, diğer amaç değerlerinin kısıtlar vasıtasıyla kısıtlandığı, tek amaçlı bir alt-problemin birden fazla kez çözülmesine dayalıdır. Her iterasyonda, kısıtlarla kontrol edilen amaç değerleri önceki çözümdeki değerlerine göre daha fazla kısıtlanır. Algoritma çözmeye çalıştığı alt problem olursuz olduğunda sonlanır. (Daha detaylı bilgi için bkz:[12]).

Bu yöntem, literatürde ilk kez bu çalışmada eşitlikçi çok amaçlı sırt çantası problemine (eşitlikçi) Pareto çözümler elde etmek için kullanılmıştır.

Teorem 1'in bir sonucu olarak Model 1 için eşitlikçi Pareto çözüm kümesini bulmak için aşağıdaki modelin Pareto çözüm kümesini bulmak yeterlidir.

Model 2

$$Max \overset{p}{Z}_1$$

$$Max \overset{p}{Z}_1 + \overset{p}{Z}_2$$

...

$$Max \overset{p}{Z}_1 + \overset{p}{Z}_2 + ... + \overset{p}{Z}_p$$

$$s.t. \sum_{i=1}^{n} c_i x_i \leq B$$

$$z_j = \sum_i v_i^j x_i \ \forall j$$

$$x_i \in \{0,1\}$$

Yukarıdaki model sıralama operatörünün kullanımı nedeniyle doğrusal olmayan bir modeldir. Ancak $p \leq 3$ için kolaylıkla doğrusal hale getirilebilir. $p=3$ durumu için kullanılan doğrusal model Ek 2 de verilmiştir.

İlgili doğrusal programlama modeli epsilon kısıt yöntemi ile çözülerek, Model 2'nin Pareto çözüm kümesi (Model 1 için eşitlikçi Pareto çözüm kümesi) bulmak mümkündür.

5. DENEYLER (COMPUTATIONAL EXPERIMENTS)

Sayısal deneyler için öncelikle literatürde kullanılan iki amaçlı kesikli sırt çantası problemleri kullanılmıştır ([2], [5], [16]) Bu problemlerde maliyet ve çıktı parametreleri rastlantısal olarak (tekdüze dağılım) aşağıdaki şekilde oluşturulmuştur. Tekdüze dağılımlar için aşağıdaki aralıklar kullanılmıştır.

- A tipi problemler (Rastgele yaratılan): $c_i \in [1,1000]$, $v^1_i \in [1,1000]$, $v^2_i \in [1,1000]$
- B tipi problemler: $c_i \in [1,1000]$, $v^1_i \in [111,1000]$, $v^2_i \in [\, v^1_i\text{-}100,\ v^1_i\text{+}100]$
- C tipi problemler: $c_i \in [1,1000]$, $v^1_i \in [1,1000]$, $v^2_i \in [\text{Maks}(900- v^1_i,1),\min(1100- v^1_i,1000)]$. *Bu yöntem bir projenin iki çıktısı arasında negative korelasyon oluşmasını sağlar.*
- D tipi problemler: $v^1_i \in [1,1000]$, $v^2_i \in [\text{Maks}(900- v^1_i,1),\min(1100- v^1_i,1000)]$, : $c_i \in [\, v^1_i + v^2_i - 200,\ v^1_i + v^2_i + 200]$.

Proje sayısı için de farklı değerler kullanılmıştır. Her parametre kombinasyonu için 10 farklı problem yaratılmıştır. Toplam bütçe $B = \sum_{i=1}^{N} c_i / 2$ olarak alınmıştır.

Algoritmalar Visual C++ da kodlanmış ve Intel Core i5 3.3 GHz işlemci ve 8GB RAM olan bir bilgisayarda çözülmüştür. Bütün matematiksel modeller CPLEX 12.5 kullanılarak çözülmüştür. Çözüm süreleri CPU saniyesi cinsinden raporlanmıştır.

Öncelikle, alt sınırların performanslarını gözlemlemek için, A tipi problemlerde (N=100 alınarak) deneyler yapılmıştır. Sonuçlar Tablo 2' de özetlenmiştir. Tabloda, ilgili alt sınır ve alt sınır kümeleri için çözüm sürelerinin, elenen durum sayısının ve bir aşamada bellekte tutulması gereken maksimum durum sayısının ortalama ve en kötü değerleri verilmiştir. Üçüncü indikatör, algoritmanın bellek gereksinimi ile ilgili bilgi vermektedir.

Sonuçlar, 4. alt sınır mekanizmasının (LB_4), bellek gereksinimi ve dolayısıyla çözüm süresi bakımından diğer alt sınırlardan daha iyi olduğunu göstermektedir. Her ne kadar LB_4 elenen durum sayısı bakımından daha kötü görünsede, bu indikatör tek başına yeterli değildir, çünkü elemelerin ne zaman yapıldığı da önemlidir. LB_4 diğer alt sınırlara göre elemeleri daha önceki aşamalarda yapabilmekte ve böylece DP ağının çok büyümesini önlemektedir. Ayrıca LB_4 ün yanı sıra 2. ve 3. alt sınırları kullanmak, çözüm süreseini arttırmakta ancak bellek gereksinimini azaltmaktadır. Bunların üstüne 1. altsınırın kullanılmasının önemli bir etkisi yoktur. Bu nedenle ana deneylerde iki versiyon denenmiştir: sadece LB_4 ve LB_2, LB_3 ve LB_4. Bu iki versiyonun sonuçları sırasıyla Tablo 3 ve 4 te verilmiştir. Epsilon kısıt algoritması sonuçları da Tablo 5 te görülebilir.

Tablo 2*. Alt sınır mekanizmalarının karşılaştırılması*

	Çözüm süresi		***Elenen durum sayısı***		***Bellekte tutulan durum sayısı***	
Alt sınır mekanizması	***Ort***	***Maks***	***Ort***	***Min***	***Ort***	***Maks***
1	1.63	6.52	20439.5	6581	6782	17554
2	0.89	5.41	12988.9	1524	3347.1	13050
3	0.94	5.53	13203.5	1688	3380.5	13051
4	0.79	4.65	11179	4277	2988.3	11902
1 ve 2	0.98	5.80	12965.6	1524	3344.2	13050
1 ve 3	1.01	5.79	13192.5	1688	3377.6	13051
1 ve 4	0.90	5.37	11120.7	3720	2985.5	11902
2 ve 3	0.99	5.99	12365.8	1351	3282.2	13051
2 ve 4	0.90	5.35	10507.5	1524	2921.6	11902
3 ve 4	0.89	5.25	10506.7	1688	2918.2	11902
1, 2 ve 3	1.08	6.38	12361.6	1351	3279.3	13051
1,2 ve 4	0.97	5.75	10504.9	1524	2918.8	11902

1, 3 ve 4	0.97	5.67	10507.5	1688	2915.3	11902
2 ,3 ve 4	0.99	5.81	10315.2	1351	2906.9	11902
Hepsi	1.06	6.18	10316	1351	2904	11902

Tablo 3. *LB_2 ve LB_4 alt sınır mekanizması kullanan DP algoritması sonuçları*

		Çözüm süresi			***Çözüm sayısı***			***Durum vektörü sayısı***		
Tip	***n***	***Min***	***Ort***	***Maks***	***Min***	***Ort***	***Maks***	***Min***	***Ort***	***Maks***
A	100	0.08	0.79	4.65	4	12.1	49	470	2988.3	11902
	200	0.89	75.10	313.42	3	33.5	92	1075	34823.3	118082
	300	32.58	1098.74	3849.27	3	68.8	163	13489	148935.8	324925
B	600	3.31	121.20	473.98	2	20.1	26	3033	58787.5	171552
	700	2.20	79.14	407.30	2	14.2	37	484	33046.1	125136
	800	10.69	184.21	1156.49	9	19.7	34	8410	59065.3	250292
	900	3.28	604.85	1673.60	1	31.2	56	2291	131816.4	289916
C	100	0.26	9.32	29.99	1	15.1	33	936	10444.8	25064
	200	3.60	103.06	299.01	5	12.3	29	2613	29351.4	82491
	300	113.11	1396.73	5026.98	3	20.7	64	15955	98999.6	247770
D	100	9.77	186.93	397.77	2	8.4	18	19081	59329.3	89564
	150	1081.55	2812.75	10228.13	4	11.6	23	121469	206899.3	366376

Tablo 4. *LB_3 ve LB_4 alt sınır mekanizması kullanan DP algoritması sonuçları*

		Çözüm süresi			***Çözüm sayısı***			***Durum vektörü sayısı***		
Tip	***n***	***Min***	***Ort***	***Maks***	***Min***	***Ort***	***Maks***	***Min***	***Ort***	***Maks***
A	100	0.02	0.99	5.81	4	12.1	49	93	2906.9	11902
	200	0.36	89.11	369.69	3	33.5	92	501	34009.8	118082
	300	4.43	1212.31	4528.17	3	68.8	163	2475	142838.2	324925
B	600	1.26	147.40	577.02	2	20.1	26	991	58583.3	171552
	700	0.46	95.00	481.12	2	14.2	37	215	32898.4	125136
	800	13.90	223.11	1326.29	9	19.6	34	8410	57318.4	250292
	900	1.22	737.28	2002.89	1	31.2	56	713	131618	289916
C	100	0.12	10.24	32.76	1	15.1	33	594	10155.6	25064
	200	4.11	107.11	325.61	5	12.3	29	2613	27978.9	82491
	300	29.26	1310.65	4401.07	3	20.7	64	9915	88078	247770
D	100	1.24	92.44	263.20	2	8.4	18	1602	25767	60684
	150	201.53	1804.55	9490.72	4	11.6	23	41113	117658.1	366502

Tablo 5. *Epsilon kısıt algoritması sonuçları*

		Çözüm süresi			***Çözüm sayısı***		
Tip	***n***	***Min***	***Ort***	***Maks***	***Min***	***Ort***	***Maks***
A	100	0.21	0.65	2.87	4	12.1	49
	200	0.30	2.69	8.03	3	33.5	92
	300	0.41	7.25	18.05	3	68.8	163
	400	0.68	8.70	28.10	4	64.7	206
	500	1.28	10.62	39.91	11	66.3	257
	600	1.36	18.30	50.05	6	89.6	270
	700	0.64	26.11	70.85	3	111	329
B	600	0.51	2.85	3.85	2	20	26
	700	0.23	2.35	5.85	2	13.9	35
	800	1.72	3.81	6.16	9	19.4	33
	900	0.41	7.19	13.10	1	31.2	56
	1000	2.56	9.79	14.29	13	39.5	52
	2000	10.93	43.07	99.41	44	104	211
	3000	54.12	166.82	253.09	69	228.9	317
	4000	116.29	365.07	532.39	125	324.3	469
C	100	0.03	2.33	15.28	1	15.1	33
	200	0.27	3.49	21.92	5	12.4	29
	300	0.71	10.31	74.09	3	20.7	64
	400	1.16	78.01	403.01	7	49.44	205
	500	1.45	68.37	181.80	3	66.8	173
D	100	0.11	1.137	4.51	2	8.4	18
	150	0.47	2.826	7.43	4	11.7	23
	200	0.45	2.334	5.82	4	11.3	30
	250	1.06	82.42	652.11	5	20.67	49

DP algoritması klasik Pareto çözümler yerine eşitlikçi Pareto çözümleri bulmak için kullanıldığında, algoritmanın bellek gereksiniminde ve çözüm süresinde azalma görülmesi beklenmektedir. Kullanılan kaynak kodlar ve donanım farklı olduğu için çözüm sürelerinin klasik sırt çantası problemi çözen DP çalışmalarıyla ([2],[5]) karşılaştırılması mümkün olmasa da, bellek gereksinimindeki düşüşü gözlemlemek mümkündür.

Bir aşamada bellekte tutulması gereken maksimum durum vektörü sayısına bakıldığında, bu sayının eşitlikçi Pareto çözümlerin bulunduğu durumlarda azaldığı gözlenmektedir. Yine de epsilon kısıt yöntemi, dinamik programlama algoritmasından daha iyi çalışmaktadır.

Bunun sebebi epsilon kısıt yönteminde, çözülmesi gereken tek amaçlı model sayısının, klasik Pareto çözümlerin bulunduğu duruma gore, büyük oranda azalması olabilir. (Eşitlikçi Pareto çözüm kümesi, Pareto çözüm kümesinin bir alt kümesidir.).

Epsilon kısıt yönteminin gösterdiği yüksek performanstan yola çıkarak, daha büyük problemlerin çözümü denenmiştir. Bu problemler yukarıda detayları verilen, veri yaratma yöntemleriyle elde edilmiştir.

Tablo 6 da çözüm sürelerinin ve bulunan çözüm sayılarının minimum, maksimum ve ortalama değerleri görülmektedir. Görüldüğü gibi algoritma, p=2 olduğu durumda büyük problemleri çözebilmektedir.

Tablo 6. *Büyük problem sonuçları*

		Çözüm süresi			Çözüm sayısı		
Tip	***n***	***Min***	***Ort***	***Maks***	***Min***	***Ort***	***Maks***
	1000	1.43	19.68	97.64	2	53.4	245
	2000	7.69	86.20	304.93	6	162.2	528
	3000	20.98	201.22	725.27	35	257	840
	4000	30.71	181.56	606.40	17	158.6	401
A	5000	11.23	355.62	867.28	5	255.5	568
	6000	36.55	278.74	706.19	15	160	405
	7000	60.36	426.87	930.14	35	246	490
	8000	85.02	534.60	1944.35	19	258.6	908
	9000	17.88	508.64	3449.84	6	169.5	1073
	10000	57.64	950.24	2862.64	11	372	1124
	5000	7.79	754.42	1230.45	6	537.6	784
	6000	742.95	1006.31	1405.63	459	664.5	836
B	7000	844.71	1581.31	2280.12	399	827.1	1171
	8000	1795.90	2497.72	2946.48	789	1101.5	1319
	9000	2964.73	3616.33	4267.92	1180	1411.5	1643
C	1000	4.22	465.419	3135.87	5	76.4	287
	2000	7113.72	7113.72	7113.72	886	886	886
D	1000	27.83	817.52	3821.37	7	114.33	289

Epsilon kısıt yöntemi, 3 kriterli (p=3) A tipi problemlerin çözümü için de kullanılmış ve sonuçlar Tablo 7 de özetlenmiştir. Görüldüğü gibi kriter sayısı arttığında çözüm süresi önemli ölçüde artmaktadır.

Tablo 7. *p=3 A tipi problem sonuçları*

		Çözüm süresi		Çözüm sayısı		
n	***Min***	***Ort***	***Maks***	***Min***	***Ort***	***Maks***
50	0.57	3.94	14.72	5	16.6	46
100	3.89	192.19	1365.34	12	74.7	255
150	4.69	51.64	252.65	9	41.4	122
200	29.37	8364.58	61016.90	36	266.1	1224
250	25.11	5007.78	22411.52	27	264.8	885

6. SONUÇ (CONCLUSION)

Bu çalışmada, verimliliğin yanısıra eşitlikçiliğin de önemli olduğu gerçek hayat uygulamalarından yola çıkarak, özel bir çok kriterli kesikli sırt çantası problemi ele alınmıştır. Amaç, eşitlikçi Pareto çözümlerin bulunmasıdır.

Bu problem için iki çözüm yaklaşımı kullanılmıştır. İlki, literatürde klasik sırt çantası problemleri için kullanılan dinamik programlama algoritmalarına dayalıdır. Algoritmada, çeşitli alt ve üst sınırlar kullanılarak eşitlikçi Pareto çözüm elde edilmesi mümkün olmayan durumlar elenmiştir. Bunun için literatürde kullanılan sınırların yanı sıra, problemin özel yapısını kullanan yeni sınırlar tanımlanmıştır.

Yapılan deneyler eşitlikçi sırt çantası probleminin çözümü için klasik sırt çantası problemine kıyasla çok daha az bellek gerektiğini göstermiştir.

Kullanılan ikinci yaklaşım, epsilon kısıt yaklaşımıdır. Bu yaklaşımın, iki kriterli problemler için, dinamik programlama algoritmasından çözüm süresi açısından daha iyi olduğu görülmüştür. Bu yaklaşım üç kriterli problemlerin çözümü için de kullanılmış ve sonuçlar raporlanmıştır.

Gelecek çalışmalar, eşitlikçi sırt çantası probleminin üç ve daha fazla kriterli versiyonları için çözüm algoritmaları geliştirmeye odaklanabilir. Bu doğrultuda Pareto çözümleri veren algoritmaların yanı sıra yaklaşık Pareto çözümler veren çok amaçlı metasezgisel algoritmalar kullanılması mümkündür. Karar vericinin tercihlerini dikkate alan interaktif yaklaşımlar geliştirilmesi de başka bir araştırma konusudur.

KAYNAKLAR (REFERENCES)

[1] D. Baatar, M.M. Wiecek, Advancing equitability in multiobjective programming, Computers & Mathematics with Applications, Volume 52, Issues 1–2, 2006, 225-234.

[2] C. Bazgan, H. Hugot, D. Vanderpooten, Solving efficiently the 01 multi-objective knapsack problem, Computers & Operations Research 36 (1) (2009) 260-279, part Special Issue: Operations Research Approaches for Disaster Recovery Planning.

[3] M. E. Captivo, J. Clímaco, J. Figueira, E. Martins, J. L. Santos, Solving bicriteria 0–1 knapsack problems using a labeling algorithm, Computers & Operations Research, 30 (12), 2003, 1865-1886.

[4] M. Ehrgott ve Gandibleux, A survey and annotated bibliography of multiobjective combinatorial optimization, OR Spectrum 22 (4), (2000) 425-460.

[5] J. R. Figueira, L. Paquete, M. Sim~oes, D. Vanderpooten, Algorithmic improvements on dynamic programming for the bi-objective f0,1g knapsack problem, Computational Optimization and Applications 56 (1) (2013) 97-111.

[6] Ö. Karsu, A. Morton, Inequity averse optimisation in operational research, 245 (2), 2015, 343-359.

[7] Ö. Karsu, A. Morton Incorporating balance concerns in resource allocation decisions: A bi-criteria modelling approach, Omega 44 (2014) 70 - 82.

[8] H. Kellerer, U. Pferschy, D. Pisinger, Knapsack Problems, Springer, Berlin, 2004.

[9] K. Klamroth, M. M. Wiecek, Dynamic programming approaches to the multiple criteria knapsack problem, Naval Research Logistics 47 (1) (2000) 57-76.

[10] M. M. Kostreva, W. Ogryczak, Linear optimization with multiple equitable criteria, RAIRO Operations Research 33 (1999) 275-297.

[11] M. M. Kostreva, W. Ogryczak, A. Wierzbicki, Equitable aggregations and multiple criteria analysis, European Journal of Operational Research, 158, (2), 2004, 362-377.

[12] M. Laumanns, L. Thiele, E. Ziztler, An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method, European Journal of Operational Research 169 (2006) 932-942.

[13] M. Laumanns, L. Thiele, E. Zitzler, An adaptive scheme to generate the pareto front based on the epsilon-constraint method, in: J. Branke, K. Deb, K. Miettinen, R. E. bSteuer (Eds.), Practical Approaches to Multi-Objective Optimization, no. 04461 in Dagstuhl Seminar Proceedings, Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, Dagstuhl, Germany, 2005.

[14] S. Martello, P. Toth, Knapsack problems: algorithms and computer implementations, John Wiley & Sons, Inc., 1990.

[15] W. Ogryczak, A. Wierzbicki, M. Milewski, A multi-criteria approach to fair and efficient bandwidth allocation, Omega, 36 (3), 2008, 451-463.

[16] A. Rong, J.R. Figueira, M. PatoA two state reduction based dynamic programming algorithm for the bi-objective 0-1 knapsack problem Computers & Mathematics with Applications, 62 (2011), pp. 2913-2930.

[17] M. Visee, J. Teghem, M. Pirlot, E. Ulungu, Two-phases method and branch and bound procedures to solve the bi-objective knapsack problem, Journal of Global Optimization 12 (2) (1998) 139-155.

EK 1 (APPENDIX 1)

DP Algoritması

Algoritma her bir S_{k-1} için S_k yı üç durum eleme kuralını kullanarak hesaplamaktadır.Bu kümelerdeki durum vektörleri şimdiye kadarki toplam maliyetlerine göre azalmayacak şekilde dizilir. Eğer toplam maliyeti aynı olan iki durum varsa minimum kategori çıktısı daha büyük olana öncelik verilir. Bu sıralama lex ile gösterilir ve 1. ve 2. eleme kurallarının daha kolay kontrol edilmesini sağlar.

$$s \geq_{lex} s' \Leftrightarrow c < c' \text{ ya da } (c = c' \text{ ve } Min\{s^1, s^2\} > Min\{s'^1, s'^2\}) \text{ ya da } (c = c' \text{ ve } Min\{s^1, s^2\} = Min\{s'^1, s'^2\} \text{ ve } Max\{s^1, s^2\} = Max\{s'^1, s'^2\})$$

Algoritmada üç tane küme kullanılır. S_k (S_{k-1}) kümeleri yukarda tanımlanmıştır. M_k kümesi 2. eleme kuralını daha hızlı bir şekilde kontrol etmek için kullanılır ve S_k kümesinin sadece çıktı vektörlerine göre (rasyonel olarak) domine edilmemiş elemanlarını tutar. s_k durumu sadece ve sadece M_k kümesinde kendisine rasyonel baskın olan başka bir durum varsa 2. baskınlık kuralı tarafından elenebilir. Bunun nedeni, durumların lex sıralaması ile baskınlığı koruyan bir sırada yaratılmış olmasıdır, yeni yaratılan bir durum en az daha önceden yaratılan durumlar kadar maliyete sahip olacaktır, dolayısıyla elemenin mümkün olup olmadığını anlamak için çıktı değerlerine bakmak yeterlidir (daha fazla bilgi için bkz: [2]). Üçüncü küme, F_k şimdiye kadar bulunan ve henüz başka bir çözüm tarafından elenmemiş alt sınır vektörlerini (olurlu tam çözümleri) tutar. Bu küme şu ana kadar elde edilmiş çözüm adaylarını tutar ve yeni bir durum, eğer üst sınır vektörü, bu kümedeki herhangi bir vektör tarafından domine edilmişse elenir.

Algoritma

S_k=Ø M_k=Ø.
$|S_{k-1}|=r$ olsun.
S_{k-1} deki 1. kural tarafından elenmemiş ilk durum vektörünü bul. (Bu durumdan sonraki durum vektörleri daha fazla toplam maliyete sahip olacağı için 1. kural tarafından elenmeyecekleri kesindir). Bu indeks j olsun.
S_{k-1} deki her durum için:
Eğer olurlu ise bu durumu k. projeyi (varlığı) ekleyerek genişlet. Bu yeni durum s^{yd} olsun.
$j \leq r$ ve $s^j \geq_{lex} s^{yd}$ olduğu sürece
Baskınlarıtut *fonksiyonunu $s^j \in S_{k-1}$ için çağır. $j = j + 1$*
Baskınlarıtut *fonksiyonunu s^{yd} için çağır.*
Baskınlarıtut *fonksiyonunu $j \leq i \leq r$ aralığındaki her $s^i \in S_{k-1}$ için çağır.*
Eğer $k=N$ ise $S_N = M_N$.
Değilse,
F_k=Ø.
S_k daki her durum vektörü için
Kullanılan her alt sınır mekanizması i için
Durum vektörü için kural i yi kullanarak bir alt sınır vektörü bul.
*Alt sınır vektörü için **Baskınlarısakla** fonksiyonunu çağır.*
S_k daki her durum vektörü için
Üst sınır hesapla

F_k da bu üst sınıra baskın olan herhangi bir alt sınır olup olmadığını kontrol et. Öyleyse, durum vektörünü S_k dan çıkar.
S_k yı dön.

Baskınlarıtut fonksiyonu verilen bir yeni durum vektörünün (s^{yd}) *2* kural tarafından elenip elenemeyeceğini kontrol eder. Eğer yeni durum elenmiyorsa, durumu M_k ve S_k kümelerine bu kümelerdeki sıralama kuralına uygun şekilde ekler.
Baskınlarısakla fonksiyonu da benzer şekilde çalışır ve F_k kümesinde sadece domine edilmemiş alt sınır vektörlerini tutar.

Bu algoritma [2]de önerilen algoritmanın eşitlikçi probleme uyarlanmış ve genişletilmis halidir.
Temel olarak iki değişiklik vardır:

1. Alt sınır vektörleri yaratılırken, bu çalışmada S_k kümesindeki bütün durumlar düşünülmüştür. [2] de ise sadece M_k dekiler kullanılmıştır. Ayrıca, iki yeni alt sınır tanımlanıp kullanılmıştır. (3. ve 4. alt sınır mekanizmaları).
2. 3. eleme kuralı ele alınan problemin özelliklerinden faydalanılarak değiştirilmiştir.

EK 2 (APPENDIX 2)

Doğrusal Model

$$Max\ f_1$$

$$Max\ z_1+z_2+z_3-f_2$$

$$Max\ z_1+z_2+z_3$$

$$s.t.\sum_{i=1}^{n} c_i x_i \le B$$

$$z_j = \sum_{i=1}^{n} v_i^1 x_i \quad j=1,2,3$$

$$f_1 \le z_j \quad j=1,2,3$$

$$f_2 \ge z_j \quad j=1,2,3$$

$$x_i = \{0,1\}$$

Model 2'yi p=3 olduğunda doğrusallaştırmak için iki yardımcı karar değişkeni kullanılmıştır. Bunlardan f_1, kategoriler üzerinden minimum faydaya, f_2 ise maksimum faydaya eşittir.