TOPLAM MALİYET - DARBOĞAZ ZAMAN TABANLI ULAŞTIRMA PROBLEMLERİ İÇİN BİR ALGORİTMA

Aydın SİPAHİOĞLU

Anadolu Üniversitesi
Fen Bilimleri Enstitüsü
Lisansüstü Yönetmeliği Uyarınca
Endüstri Mühendisliği Anabilim Dalı
Yöneylem Araştırması Bilim Dalında
YÜKSEK LİSANS TEZİ
Olarak Hazırlanmıştır.

Danışman : Prof. Dr. İmdat KARA

ŞUBAT — 1990

Aydın Sipahioğlu'nun YÜKSEK LİSANS tezi olarak hazırladığı "Toplam Maliyet-Darboğaz Zaman Tabanlı Ulaştırma Problemleri İçin Bir Algoritma" başlıklı bu çalışma, jürimizce lisansüstü yönetmeliğinin ilgili maddeleri uyarınca değerlendirilerek kabul edilmiştir.

13./02./1990

üye : Prof.Dr.İmdat KARA

üye : Doçdr.Nimetullah BURNAK

üye : Y.Doç.Dr.Doğan EROL

Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 14.ŞUBAT.1990. gün ve 233/8..... sayılı kararıyla onaylanmıştır.

Prof. Dr. Rüstem KAYA
Enstitü Müdürü

İÇİNDEKİLER

İÇİNDEKİLER (devam)

ŞEKİLLER DİZİNİ

ÖZET

Darboğaz ulaştırma problemi, zamanın çok önemli olduğu günümüzde, gerek stratejik ve gerekse günlük hayatta uygulanabilecek önemli bir karar problemidir. İki amaçlı ulaştırma problemi ise hem zamanın hem de maliyetin birarada düşünülerek eniyilenmeye çalışıldığı karar problemidir. Her iki karar problemi de pek çok uygulama alanına sahiptir. Bu nedenle ikisinin de önemini koruduğu ve değişik çözüm yaklaşımlarının geliştirilmesiyle daha da önem kazanacağı görülmektedir.

Bu çalışmada, darboğaz ulaştırma problemleri hem tek hem de çok amaçlı olmak üzere ayrı ayrı ele alınmıştır. Her iki problem türünün ayrıntılı olarak tanımları yapılmış, gelişim süreçleri incelenerek günümüze değin geliştirilen bazı algoritmalardan örnekler verilmiştir.

Çalışmada, toplam maliyet ve darboğaz zaman tabanlı iki amaçlı bir ulaştırma problemi için, sıralı çözüm ikilileri bularak üstün çözüm ikililerini ayıran, daha sonra karar vericiye önerilecek seçeneği belirleyen bir yöntem geliştirilmiştir. Önerilen yöntemin işlemleri, bir algoritma haline getirilerek uygulaması için Turbo Pascal programlama dili ile bir de paket program hazırlanmıştır.

SUMMARY

Bottleneck transportation problem is both strategic and applicable in the daily life, where time is very important. Furthermore, two-objective transportation problem is a decision problem in which both time and cost are thought together to obtain an optimal solution. Both decision problems have various application area. Therefore, it has been seen that both keep their importance and gain importance by improving various solution approaches.

In this study, bottleneck transportation problems have been investigated separately either one-objective or two objectives. Detailed definition of each kind has been stated, some improved algorithm examples that has been investigated in historical perspective has been given.

A method which can find sequential pairs, and choose superior ones, and then define the choice that will be proposed to the decision-maker has been developed for total cost and bottleneck time based two-objective transportation problem. The operations of proposed method has been converted to an algorithm and software which is written with Turbo Pascal has been prepared for application.

1. GİRİŞ

Doğrusal Programlama (Linear Programming-LP), 2. dünya savaşı yıllarında yoğun şekilde karşılaşılan, kısıtlı kaynakların dağıtımı problemlerini sistematik bir yaklaşımla çözüme kavuşturmak isteyen araştırmacıların çabaları sonucunda geliştirilmiştir. Konuyla ilgili ilk çalışmalar 1940'lı yıllarda Yöneylem Araştırması çalışmaları adı altında İngiliz araştırmacılar tarafından gerçekleştirilmiştir. Bu dönem, aynı zamanda pek çok Yöneylem Araştırması tekniğinin de temel olarak oluşturulduğu dönemdir. İzleyen yıllarda Amerikalı araştırmacı George B.Dantzig ve ekibi, doğrusal programlamanın teorisi ile çözüm tekniklerini geliştirmiştir.

Doğrusal Programlamanın günümüze kadar en fazla uygulanan ve en çok ilgiyi gören alt başlığı Klasik Ulaştırma Problemi (Standard Transportation Problem-STP) olmuştur. Klasik ulaştırma probleminde (KUP) amaç, belli sayıdaki üretim merkezleri ile dağıtım bölgeleri arasında, toplam taşıma maliyetini enküçükleyecek dağıtım programını bulmaktır. KUP'un çözüme kavuşturulmasından sonra üretim merkezleri ile dağıtım bölgeleri arasında yapılacak taşımada, ölçüt olarak sadece taşıma maliyetinin değil, zamanın da alınabileceği düşünülmüş ve bu fikirden hareketle 1959 yılında Zaman Yönlü Ulaştırma Problemi (Time Transportation Problem-TTP) tanımlanmıştır. Günümüzde aynı problem, Darboğaz Ulaştırma Problemi (Bottleneck Transportation Problem-BTP) olarak anılmaktadır.

Darboğaz Ulaştırma Probleminde (DUP) amaç, üretim merkezlerindeki ürünün, dağıtım merkezlerine mümkün olan enkısa sürede ulaştırılmasını sağlayacak dağıtım programını bulmaktır. Başka bir ifadeyle darboğaz ulaştırma probleminde, sürenin taşıma maliyetlerinden daha önemli olduğu kabul edilmektedir. Kısaca, taşımadaki maliyet bileşeni gözardı edilmektedir. Taşıma süresinin enküçüklenmeye çalışılması ve maliyet bileşeninin gözardı

edilmesi ilk bakışta gerçekçi görünmeyebilir. Oysa insanlığın karşılaşabileceği sel, kasırga, deprem vb. herhangi bir doğal afetten sonra, felaket bölgelerine gönderilecek yardım ekiplerinin ve yardım malzemelerinin ulaştırılması söz konusu olduğunda, taşıma maliyetinin değil taşıma süresinin önem kazanacağı açıktır. Benzer şekilde bir savaş sırasında, cephelerin taleplerinin karşılanmasında en önemli faktör zaman olacaktır. Aynı pazarı paylaşan işletmeler için de, yeni bir ürünü piyasaya sunmadaki öncelik kaygısı, yine zamanın önemini ön plana çıkarmaktadır. Nitekim Cola şirketleri arasındaki rekabet ve Avrupalı otomobil üreticilerinin, daha ekonomik otomobil üreterek piyasayı Japonlara kaptırmama çabaları, buna iyi birer örnek oluşturmaktadır. Kolay bozulabilen besin maddelerinin taşınması probleminde de zamanın ne kadar önemli olduğu açıktır. Bu örneklerden de anlaşılabileceği gibi DUP, gerek günlük hayatta ve gerekse stratejik kararlarda karşılaşılabilecek önemli bir karar problemi olarak kendini göstermektedir.

1970'li yıllarda yaşanan çeşitli ekonomik durgunluklar, bunalımlar ve özellikle piyasalardaki yoğun rekabet; işletmeleri, çoğu karar problemlerini hem zaman hem de maliyet yönüyle ele almaya zorlamıştır. Bu gereksinim, ulaştırma problemlerine de yansıyarak yeni yaklaşımların ve özellikle de aynı ulaştırma problemi için çok amaçlı çözüm yaklaşımlarının geliştirilmesine yol açmıştır. Günümüzde aynı ulaştırma problemi için farklı iki amaca göre çözümler üretilerek, karar vericiye maliyet-zaman ikililerinden oluşan bir çözüm kümesi sunulmakta ve bu küme içinden seçim, karar vericiye bırakılmaktadır. Geetha-Vartak ikilisi tarafından 1989 yılında yayınlanan makalelerde, üç boyutlu bir atama problemi için maliyet ve zaman değerlerinden oluşan sıralı çözüm ikilileri bulunarak, içlerinden sadece bir tanesinin karar vericiye önerilmesine ilişkin bir yöntem geliştirilmiştir (Geetha and Vartak, 1989). Böylece üç

boyutlu bir atama problemi için, karar vericiye, elde edilen çözüm kümesinin içinden önerilebilecek bir çözümün belirlenmesi olanağı sağlanmıştır.

Yukarda özetlenen gelişim doğrultusunda bu çalışmayla;

- Darboğaz ulaştırma problemleri ve çözüm yaklaşımlarını incelemek,
- Ulaştırma problemini maliyet-zaman yönüyle ele alan yaklaşımları inceleyerek, üstün sıralı çözümlere ulaşmak için kolay uygulanabilir bir algoritma geliştirmek,
- Ulaştırma problemlerinin üstün sıralı ikililerini, Geetha-Vartak'ın atama problemleri için önerdiği değerlendirme yaklaşımıyla ele alarak, karar vericiye tek çözümün önerilebilirliğini araştırmak,

amaçlanmıştır.

Belirtilen amaçlara bağlı olarak çalışma beş bölümden oluşturulmuştur. İkinci bölümde DUP ele alınarak, bugüne kadar DUP ile ilgili olarak gerçekleştirilen araştırmaların ayrıntılı tarihçesi sunulmaktadır. Ayrıca, DUP için geliştirilmiş bazı önemli çözüm yaklaşımları da ayrıntılarıyla tanıtılmaktadır.

Üçüncü bölümde iki amaçlı ulaştırma problemleri incelenmektedir. Bu bölümde özellikle, iki amaçlı ulaştırma problemlerinin çeşitleri açıklanmakta, bu konu üzerinde çalışan araştırmacıların çalışmalarına ait kısa bir tarihçe sunularak, geliştirilen algoritmaların bir kısmı özetlenmektedir.

Dördüncü bölümde ise iki amaçlı bir ulaştırma problemi için klasik ulaştırma probleminin çözüm yöntemlerinden biri olan MODI metodunu ve darboğaz ulaştırma probleminin çözüm yöntemlerinden biri olan Primal algoritmayı birarada kullanarak, sıralı çözümler üreten ve daha sonra bu çözümler arasından üstün olanlarını seçerek karar vericiye sunulacak çözüm kümesini belirleyen bir yöntem ve buna ait algoritma geliştirilmektedir. Ayrıca, 1989 yılında

Geetha-Vartak tarafından üç boyutlu atama problemleri için geliştirilen, üstün çözümler arasından bir tanesini seçme yöntemi, iki amaçlı ulaştırma problemlerine uyarlanarak, aynı algoritmaya eklenmektedir. Böylece ilk kez iki amaçlı bir ulaştırma problemi için üstün çözümler arasından sadece bir tanesini karar vericiye önerme olanağı sağlanmaktadır. Yine bu bölümde, geliştirilen algoritmanın Turbo Pascal programlama dili ile yazılan ve bir paket program olarak düzenlenen, program listesi sunulmakta, algoritma, rassal olarak türetilen bir örnek üzerinde denenerek elde edilen sonuçlar Ek-III'de verilmektedir.

2. DARBOĞAZ ULAŞTIRMA PROBLEMİ

Darboğaz Ulaştırma Problemi (DUP), Klasik Ulaştırma Probleminden (KUP) esinlenilerek geliştirilmiş, ancak amaç fonksiyonunun doğrusal olmaması, modelin bilinen tekniklerle çözüme kavuşturulmasına engel olmuştur. Problemin tanımlanmasından sonraki ilk çalışmalarda, genellikle simpleks algoritmasından yararlanılmaya çalışılmış, sonradan problemin KUP'a benzetilerek çözümünün daha etkin olacağı anlaşılmıştır. İlerleyen yıllarda problemin, serim modeli şeklinde ifade edilerek de çözümünün mümkün olduğu görülmüştür.

Bu bölümde DUP tanımlanarak genel yapısı açıklanıp KUP ile olan ilişkisi ortaya konmaktadır. Ayrıca DUP'un günümüze değin izlenen gelişim süreci ayrıntılı olarak açıklanmakta ve geliştirilen önemli bazı çözüm yaklaşımları tanıtılmaktadır.

2.1. Klasik Ulaştırma Problemi (KUP)

Genel olarak m üretim merkezinde elde bulundurulan ya da üretilen malların (ürün,malzeme vb.) n dağıtım bölgesine, toplam taşıma maliyetini enküçükleyecek şekilde ulaştırılmasını sağlayacak taşıma programının bulunması problemine, Klasik Ulaştırma Problemi (KUP) denir. KUP ile ilgili öncü niteliğindeki ilk çalışmalar 1940'lı yıllarda Hitchock ve Koopmans tarafından yapılmış (Lee and Moore,1973), ancak problemin genel yapı açısından ortaya konması ve çözümü G.B. Dantzig tarafından 1949 yılında gerçekleştirilmiştir (Phillips et.al.,1976).

Problemin genel yapısı şöyle verilebilir:

i indisi, üretim merkezlerini, (satırlar),

j indisi, dağıtım bölgelerini, (sütunlar),

göstermek üzere I ve J aşağıdaki gibi tanımlanır.

$$I=\{i|i=1,2,\ldots,m\} \quad \text{ve} \quad J=\{j|j=1,2,\ldots,n\}$$

a_i : i. üretim merkezinin kapasitesi,

b_j : j. dağıtım bölgesinin talebi,

$x_{i,j}$: i. üretim merkezinden j. dağıtım bölgesine fiziksel olarak taşınacak miktar,

$c_{i,j}$: i. üretim merkezinden j. dağıtım bölgesine yapılacak bir birim taşımanın maliyeti,

iken KUP'a ait dengelenmiş model;

$$\sum_{j=1}^{n} x_{ij} = a_i \qquad i = 1,2,\ldots,m$$

$$\sum_{i=1}^{m} x_{ij} = b_j \qquad j = 1,2,\ldots,n$$

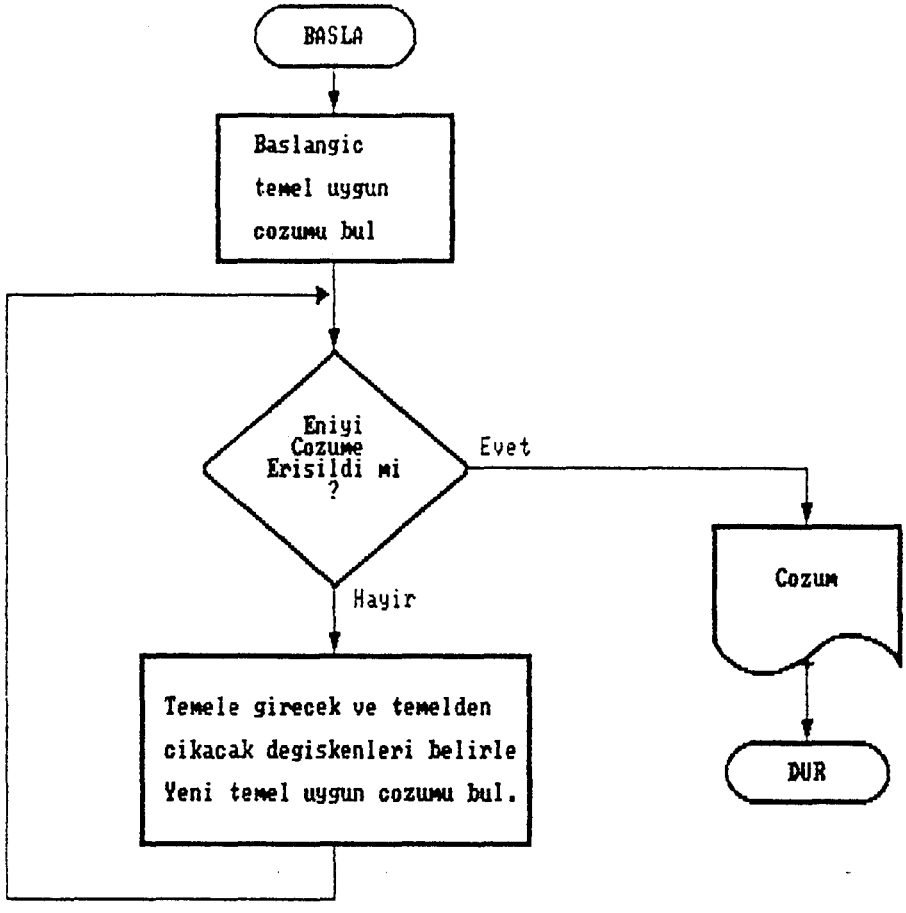
$$x_{ij} \geq 0 \ , \ a_i > 0 \ , \ b_j > 0$$

kısıtları altında,

$$enkz = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} . x_{ij}$$

şeklinde yazılır.

Klasik Ulaştırma Problemini, simpleks algoritmasını kullanarak çözmek mümkündür. Ancak problem boyutlarının büyütülmesi ile karar değişkeni ve kısıt sayısında görülen çok hızlı artış ve her temel uygun çözümde m + n - 1 tane karar değişkeninin temelde bulunması zorunluluğu nedeniyle ortaya çıkan bozulma, KUP için özel çözüm tekniklerinin geliştirilmesine neden olmuştur. Atlama taşı (Stepping stone) ve MODI (MOdified DIstributions) bu çözüm tekniklerinin en yaygın olarak kullanılanlarıdır. Gerçekte bütün bu tekniklerin dayandığı, şekil 2.1'de akış şeması verilen, ardıştırmaya dayalı tek bir yaklaşım vardır.



Sekil 2.1: KUP çözüm algoritmalarının genel akış şeması.

Günümüzde KUP'un çözüm yaklaşımı ile ilgili çalışmalar bitmiş değildir. Özellikle ardıştırma sayısının olabildiğince azaltılmasını sağlamak amacıyla, değişik başlangıç temel uygun çözüm bulma algoritmaları üzerinde çalışılmaktadır (Satır ve Kırca, 1989). Ayrıca KUP'a ait eniyi çözümün bulunabilmesi için de yeni yaklaşımlar üzerinde çalışılmaktadır (Arsham and Kahn, 1989 ve Datta, 1984).

2.2. Darboğaz Ulaştırma Problemi (DUP)

m üretim merkezinde elde bulundurulan malların, mümkün olan en kısa sürede, n dağıtım bölgesine ulaştırılmasını sağlayacak dağıtım programının bulunması problemine, Darboğaz ulaştırma problemi (DUP) denir.

KUP'un çözüme kavuşturulmasından sonra, üretim merkezleri ile dağıtım bölgeleri arasında yapılacak taşımalarda sadece maliyetin değil, zamanın da söz konusu olabileceği düşünülerek ilk kez 1959 yılında zaman yönlü ulaştırma problemi (Time Transportation Problem-TTP) tanımlanmıştır (Szwarc, 1971). Zaman yönlü ulaştırma probleminde toplam taşıma zamanı gibi bir kavramın oluşturulamayacağı açıktır. Çünkü fiziksel olarak taşınan miktar ile (x_{ij}), taşıma süresi (t_{ij}) çarpımının sonucunda elde edilen değerin, birim açısından bir anlamı yoktur. Ancak öyle bir zaman değeri vardır ki, üretim merkezlerinden dağıtım bölgelerine yapılacak olan bütün mümkün taşımaların süresi bu değere eşit veya ondan daha küçüktür. Bir başka deyişle, herhangi bir ulaştırma programında, bütün taşımalara ait zaman değerlerinden oluşan bir küme vardır ve bu kümedeki en büyük değer, ulaştırma programının uygulanarak tamamlanabilmesi için geçmesi gereken zamanı gösterir. Bu değere de darboğaz zaman (Bottleneck Time-BT) denmektedir. Gerçekte önceleri zaman yönlü ulaştırma problemi olarak isimlendirilen bu problemin, sonradan darboğaz ulaştırma problemi olarak anılmasındaki neden de, darboğaz zaman kavramının tanımlanması ve bu kavramın olayı daha iyi açıklamasıdır. Buradan türetilen bir başka kavram ise darboğaz zaman olarak belirlenen süre boyunca taşınması gereken miktarı ifade eden, Darboğaz Taşıma (Bottleneck Shipment-SB) kavramıdır.

DUP'ta amaç, enküçük darboğaz zamanı sağlayacak ulaştırma programını bulmaktır. Srinivasan-Thompson tarafından bu amaca ek olarak darboğaz zamanda yapılacak taşımanın da enküçüklenebileceği gibi yeni bir amaç tanımlanmış ve buna ilişkin bir algoritma geliştirilmiştir (Srinivasan and Thompson, 1976).

DUP'a ait dengelenmiş model aşağıdaki gibi verilebilir:

i indisi, üretim merkezlerini (satırlar),

j indisi, dağıtım bölgelerini (sütunlar),

göstermek üzere I ve J aşağıdaki gibi tanımlanır.

$I=\{i \mid i=1,2,\ldots,m\}$ ve $J=\{j \mid j=1,2,\ldots,n\}$

a_i : i. üretim merkezinin kapasitesi,

b_j : j. dağıtım bölgesinin talebi,

x_{ij} : i. üretim merkezinden j. dağıtım bölgesine fiziksel olarak taşınacak miktar,

t_{ij} : i. üretim merkezinden j. dağıtım bölgesine yapıcak taşımanın süresi,

iken DUP'a ait dengelenmiş model;

$$\sum_{j=1}^{n} x_{ij} = a_i \qquad i=1,2,\ldots,m$$

$$\sum_{i=1}^{m} x_{ij} = b_j \qquad j=1,2,\ldots,n$$

$$x_{ij} \geq 0 \ , \ t_{ij} \geq 0 \ , \ a_i > 0 \ , \ b_j > 0$$

kısıtları altında,

$$enk\, z = \underset{((i,j) \mid x_{ij}>0)}{enb}\ t_{ij}$$

şeklinde yazılır.

Kolaylıkla görülebileceği gibi DUP ile KUP arasında özellikle kısıtlar yönünden hiç bir ayrım yoktur. Modeller arasındaki fark sadece amaç fonksiyonları arasındadır. Ancak DUP için oluşturulan amaç fonksiyonunun doğrusal olmadığına dikkat edilmelidir. Bu nedenle herhangi bir DUP problemi bilinen doğrusal karar modeli çözüm teknikleri ile çözüme kavuşturulamaz. Bu zorluk, DUP için yeni çözüm tekniklerinin geliştirilmesine neden olmuştur.

2.3. DUP'un Tarihsel Gelişimi

Darboğaz ulaştırma problemini ilk olarak tanımlayan ve 1959 yılındaki çalışmalarıyla çözüme kavuşturan araştırmacı A.S.Barsow'dur (Szwarc, 1971). Barsow, problemi simpleks algoritmasının özelliklerinden yararlanarak çözmüştür. Bu

çalışmadan sonra 1962 yılında E.P. Niestierow tarafından aynı problem, ikillikden yararlanılarak çözülmüştür (Szwarc, 1971). Bu iki çalışmanın ortak özelliği, DUP üzerinde gerçekleştirilmiş ilk çalışma olmaları ve çözümün özünde doğrusal programlama özelliklerinin bulunmasıdır.

DUP'a ilişkin çalışmalardan bir diğeri de, I.W. Romanowski tarafından geçekleştirilen ve DUP'un KUP'a indirgenerek çözülmesini sağlayan ardışık çözüm yöntemidir (Szwarc, 1971). Benzer bir başka yöntem de W. Grabowski tarafından 1964 yılında gerçekleştirilmiştir (Szwarc, 1971). Grabowski yönteminde, DUP'u bir dizi dönüşüme tabi tuttuktan sonra problemi KUP biçiminde çözmektedir. Bu iki çalışmanın ortak özelliği ise ilk kez DUP'un çeşitli dönüştürme işlemleri ile değiştirilmesi ve ardıştırmaya dayalı yöntemlerle problemin çözüme kavuşturulmasıdır. Bir başka özellik de, simpleks algoritmasından uzaklaşılmış olunmasıdır.

Grabowski'den sonra DUP'un gerçek anlamda yapısını ve terimlerini tanımlayarak, ardıştırmaya dayalı çözüm yaklaşımı öneren iki araştırmacı gelmektedir: W. Szwarc ve P.L. Hammer. Gerçekte bu iki araştırmacı birlikte çalışmamışlardır ama, sonuçta geliştirdikleri algoritmalar; bugün, birbirlerine çok benzeyen, aynı nitelikte iki algoritma olarak kabul edilmektedir (Srinivasan and Thompson, 1976). Yayınlanan makalelerin tarihleri açısından Szwarc'ın çalışmalarına daha erken başladığı söylenebilir. Bu konudaki gelişmeler Szwarc'ın 1966 yılında teorik yapısı çizge kuramına dayanan (serim modelleri) ve sıralı temel uygun çözümler bularak her adımda eldeki çözümü biraz daha iyileştirmeye çalışan bir algoritma geliştirmesiyle başlamıştır. Szwarc, 1966-71 yılları arasında üstüste yayınladığı makalelerle tekniğini zenginleştirmiştir (Srinivasan and Thompson, 1976). DUP ile ilgili çalışmalar 1966 yılında S.I. Zukhowitsky ve L.I. Avdeyeva' nın, bilinen bir çözüm yönteminin iki farklı türünü yayınlamaları ile devam etmiştir. Ancak sonradan bu

teknik ile bulunan bazı çözümlerin, temel uygun çözüm olmadıkları belirlenmiştir (Srinivasan and Thompson, 1976). 1969 yılında Polonya'lı araştırmacı A. Janicki tarafından yüksek lisans tezi olarak Szwarc'ın geliştirdiği algoritmanın bilgisayar programı gerçekleştirilmiştir. Janicki çalışmasında bu çözüm yönteminin her problem için sonlu olduğunu da göstermiştir. Yine 1969 yılında bu kez de Hammer, Szwarc'ın geliştirdiği algoritmaya çok benzeyen bir başka yöntem geliştirmiştir. Bu çalışmaların ortak özelliği, ilk kez DUP için sıralı temel uygun çözümlerin bulunabileceği fikrinin ortaya atılması ve aynı problem için komşu çözümlerden oluşan bir çözüm kümesinin elde edilmesidir. Çözümün temelinde, bir başlangıç temel uygun çözüm bulunarak darboğaz zamanda taşınacak miktarın mümkün olan diğer hücrelere dağıtılması fikri bulunmaktadır. 1976 yılında Srinivasan-Thompson tarafından bu çalışmaların sonucu niteliğinde bir algoritma geliştirilmiştir. Szwarc'ın algoritması, Srinivasan-Thompson algoritmasının bir özel hali olarak kabul edilmektedir (Srinivasan and Thompson, 1976).

Szwarc ve Hammer'dan sonraki en önemli çalışma, 1971 yılında R.S. Garfinkel ve M.R. Rao tarafından gerçekleştirilmiştir. Bu ikiliye ait Primal algoritma ve Threshold algoritması olmak üzere iki önemli algoritma vardır. Primal algoritma yapı olarak oldukça basittir ve zaman matrisini, bulunan bir başlangıç temel uygun çözümdeki darboğaz zaman değerine göre değiştirerek, problemin KUP biçiminde çözülmesini öngörür. Algoritmanın kolay anlaşılır olması ve problem için sıralı komşu çözümleri vermesi, yöntemin iki önemli avantajıdır. Threshold algoritması ise Primal algoritmadan yapı olarak tamamen farklıdır. Algoritma, Ford-Fulkerson tekniğinin serim modellerindeki enbüyük akışı bulma problemi yöntemine dayanmaktadır. Önemli bir özellik, bu algoritmanın a_{ij} ve b_{ij} değerleri rasyonel veya doğal sayı iken denenmiş olması ve sonuç alınmasıdır. Ancak algoritmanın en önemli dezavantajı da problemde sadece başlangıç temel uygun çözüm

ile sonucun görülebilmesi, ara çözümlerin görülememesidir (Srinivasan and Thompson, 1976). Aslında Primal algoritma yapı olarak, daha önce Grabowski ve çağdaşı diğer araştırmacılar tarafından geliştirilen DUP'un KUP'a dönüştürülerek çözülmesi mantığını taşımaktadır. Bu açıdan bakıldığında yeni bir çözüm tekniği olduğu söylenemez. Ancak Primal algoritmanın üstünlüğü DUP'un KUP'a dönüştürülmesindeki işlemlerin çok kolay ve anlaşılır olmasıdır. Oysa Grabowskinin önerdiği dönüştürme işlemleri Primal algoritmadaki kadar açık ve kolay anlaşılır değildir. Threshold algoritmasının bilime olan katkısı ise, DUP'un serim modeli halinde ifade edilerek en büyük akış problemi gibi çözülebileceği mantığını getirmesidir. Ayrıca Threshold algoritması darboğaz zamanı enküçüklerken, darboğaz taşımayı da enküçüklemeye çalışmaktadır. Bu nedenle Garfinkel ve Rao tarafından geliştirilen bu iki algoritmanın DUP'un gelişim sürecinde önemli bir yer tuttuğu söylenebilir.

1976 senesinden itibaren DUP ile ilgili çalışmalarda dallanma görülmüş ve çalışmalar ikiye ayrılmıştır. Birinci grupta yine DUP'un çözümüne ilişkin çalışmalar yapılırken, ikinci grupta, iki amaçlı ulaştırma problemleri üzerinde çalışılmıştır. İki amaçlı ulaştırma problemlerine ilk kez dikkati çeken araştırmacılar 1976 yılındaki çalışmaları ile V.Srinivasan ve G.L.Thompson olmuştur. Srinivasan-Thompson aynı ulaştırma problemi için hem maliyet hem de zaman matrislerinin varolduğunu belirterek toplam maliyeti enküçükleme, darboğaz zamanı enküçükleme ve darboğaz taşımayı enküçükleme olmak üzere üç ayrı amaç tanımlamışlardır. Aynı çalışmada bu amaç fonksiyonlarını ikili olarak ele alıp, farklı çözüm algoritmaları önermişlerdir. Problemin çözümünde amaç, sıralı komşu çözümler bularak maliyet ve zaman değerlerinden oluşan çözüm ikililerini elde etmektir. Bu çalışmalarla ilgili ayrıntılı inceleme çalışmanın üçüncü bölümde yeralmaktadır.

DUP ile ilgili son yıllara ait çalışmalardan biri de R.A. Russel, D.D. Klingman ve P.P. Navid tarafından 1983 yılında gerçekleştirilmiştir. Çalışmada DUP için önerilen yeni yaklaşımda, problem bir serim modeli halinde ifade edilerek Charnes ve Cooper tarafından geliştirilen poly-w yöntemine dayanan ve XTI olarak kısaltılan Extended Threaded Index tekniğini kullanan bir yöntem önerilmektedir (Russel et.al., 1983). Aynı çalışmada darboğaz taşımanın enküçüklenmesine ilişkin bir algoritma ve ek kısıtları bulunan DUP için de bir çözüm yaklaşımı önerilmektedir.

DUP ile ilgili son olarak sözü edilecek bir diğer araştırmacı Heinz Isermann'dır. Isermann, 1984 yılındaki çalışması ile, DUP'un doğrusal olmayan amaç fonksiyonunu doğrusallaştırmayı ve ikillikten yararlanarak problemi çözüme kavuşturmayı önermektedir (Isermann, 1984). Ayrıca önerilen algortima, darboğaz taşımayı da enküçüklemektedir. Aynı çalışmada, maliyet-zaman tabanlı iki amaçlı ulaştırma problemi için de, yine doğrusal olmayan amaç fonksiyonunu doğrusallaştırılarak, bir çözüm yaklaşımı önerilmektedir.

DUP ile ilgili çalışmalar henüz tamamlanmamıştır. Ayrıca Szwarc tarafından temelleri atılan ve Garfinkel-Rao tarafından geliştirilen DUP'un bir serim modeli şeklinde ifade edilerek çözüme kavuşturulması döneminin de kapandığı söylenemez. Özellikle Hintli araştırmacıların konuya ilgi gösterdikleri son yıllarda yayınlanan makalelerden anlaşılmaktadır (Khanna, 1983 ve Malhotra, 1983).

2.4. DUP İçin Geliştirilen Çözüm Teknikleri

DUP için geliştirilen çözüm teknikleri aşağıdaki dört grupta toplanabilir:

i- Doğrusal programlama özelliklerinin ve simpleks algoritmasının kullanıldığı teknikler.

ii- DUP'un KUP'a dönüştürülerek çözüme kavuşturulduğu teknikler.

iii- DUP için sıralı çözümlerin belirlendiği ve algoritmik yaklaşımların kullanıldığı teknikler.

iv- DUP'un bir serim modeli halinde ifade edilerek çözüme kavuşturulduğu teknikler.

Kuşkusuz bu genel gruplar arasında da alt ayrımlar yapılabilir. Ancak bu dört grup, DUP'un gelişimini genel hatlarıyla dönem temelinde açıklamaktadır. Bu gruplandırmaya göre birinci ve ikinci grup, DUP'a ait ilk çalışmaların yapıldığı dönemler, üçüncü ve dördüncü grup ise yakın zaman içinde geliştirilen çözüm tekniklerinin bulunduğu dönemler olarak ele alınabilir.

Aşağıda, DUP için geliştirilen farklı çözüm tekniklerinden, günümüze daha yakın tarihli olan çalışmalar ağırlıklı olmak üzere, örnekler verilmektedir. Bu yaklaşımla ilk olarak Grabowski'nin tekniği sonra da Garfinkel-Rao ve Srinivasan-Thompson ikililerinin çalışmalarına yer verilmiştir. Srinivasan-Thompson ikilisi tarafından gerçekleştirilen çalışmanın Szwarc ve Hammer'a ait algoritmaları da kapsaması ve onların genelleştirilmiş hali olması nedeniyle Szwarc ve Hammer'a ait çalışmalardan söz edilmemiştir. Son olarak farklı bir yaklaşımı olduğu için Isermann'ın çalışmasına yer verilmiştir. Tekniklerin açıklanmasında her algoritma için bir makro akış şeması çizilmiştir. Bazı temel kavramlar dışında, yazarların kullandıkları gösterimlerin de korunmasına çalışılmıştır. Buna göre aşağıda tanıtılan bütün yöntemler için;

DZ: Herhangi bir çözümdeki darboğaz zaman,

T: Taşıma zamanlarını ifade eden ve $t_{i,j}$ değerlerinden oluşan zaman matrisi,

B: Herhangi bir çözümde, temelde yer alan karar değişkenlerine ait (i,j) hücrelerinin bulunduğu küme $B=\{(i,j)|x_{ij}>0\}$,

X: Herhangi bir çözümde karar değişkenlerinin bulunduğu küme $X=\{x_{ij}|(i,j)\in B\}$,

γ: Zaman matrisindeki (i,j) hücrelerinin tamamını içeren küme,

olarak ele alınacaktır.

2.4.1. Grabowski'nin çözüm tekniği

Grabowski'nin geliştirdiği algoritma DUP'un, KUP'a dönüştürülerek çözüme kavuşturulduğu tekniklere ilk örneklerden biridir. Bu nedenle algoritmadaki dönüşüm işlemleri uzundur. Temel olarak yöntem, bir başlangıç çözüm bulunduktan sonra elde edilen çözüme ait $t_{i,j}$ değerlerini farklı iki değer arasında ifade etmek ve bunlardan yararlanarak zaman matrisini, maliyet matrisine dönüştürmek şeklinde özetlenebilir.

Grabowski algoritmasının ilk adımında herhangi bir yöntem kullanılarak bir başlangıç çözüm bulma işlemi vardır. Bu adımın uygulanmasından sonra B kümesi elde edilmiş olsun. B kümesinin elemanlarına ait olan $t_{i,j}$ değerlerinin tamamı kolaylıkla iki farklı zaman değeri arasında ifade edilebilir. Grabowski bu değerleri t' ve t'' olarak belirlemekte ve bu değerlerin belirlenmesinde herhangi bir önşart koymamaktadır. Ancak kendisinin bir önerisi

$$t' = enb\left(\underset{i}{enb}\, \underset{j}{enk}\, t_{ij} \;,\; \underset{j}{enb}\, \underset{i}{enk}\, t_{ij} \right) \quad ve \quad t'' = DZ$$

olarak seçilmesidir. Burada $t' \leq \underset{(i,j)\in B}{t_{ij}} \leq t''$ ilişkisi de vardır.

Özet olarak yapılan işlem, başlangıç çözüme ait zaman değerlerini matristeki iki farklı değer arasında ifade etmektir.

$(i,j)\in B$ için bütün farklı $t_{i,j}$ değerleri enküçük değerden başlayarak $t_0 = t'$, $t_q = t''$ olmak üzere artan sırada dizilsinler. Böylece k=0,1,2...,q için artan sırada dizilmiş zaman değerlerinden oluşan $(t_0, t_1, t_2, \ldots, t_q)$kümesi elde edilmiş olur.

Belirlenen bu kümeden yararlanarak (i,j) ikililerinden oluşmak üzere, γ kümesinin Ω_k alt kümeleri tanımlanır. Buna göre; k = 1,2,3,...,q+1 ve $(i,j)\in\gamma$ iken,

Ω_0, $t_{ij} \leq t_0$ için (i,j) ikililerinin kümesidir.

Ω_k, $t_{ij}=t_k$ için (i,j) ikililerinin kümesidir.

Ω_{q+1}, $t_{ij}>t_k$ için (i,j) ikililerinin kümesidir.

Ω_k kümesindeki elemanların sayısı ise h_k ($h_k \geq 1$) ile gösterilir.

Alt kümelerin tanımlanmasından sonra DUP'u KUP biçiminde çözmek için gerekli olan maliyet matrisi bu alt kümelerden yararlanılarak tariflenir. Buna göre; k = 1,2,3,...,q+1 olmak üzere;

$\omega_0 = 0$

$\omega_1 = 1$

$\omega_{k+1} = (h_k + 1).\omega_k$

olarak tanımlanır.

Elde edilen w_k değerlerinden yararlanılarak da $c_{ij} = \omega_k$, $(i,j) \in \Omega_k$ olmak üzere mxn'lik yeni bir $C=\{c_{i,j}\}$ matrisi oluşturulur ve bu matris kullanılarak KUP çözülür. Elde edilen çözüm, enküçük darboğaz zamanlı ulaştırma programının sağlanacağı çözümdür. Grabowski'nin algoritmasına ait akış şeması şekil 2.2'de verilmiştir.

Grabowski'nin Algoritması:

1.Adım: Problem için bir başlangıç temel uygun çözüm bul.

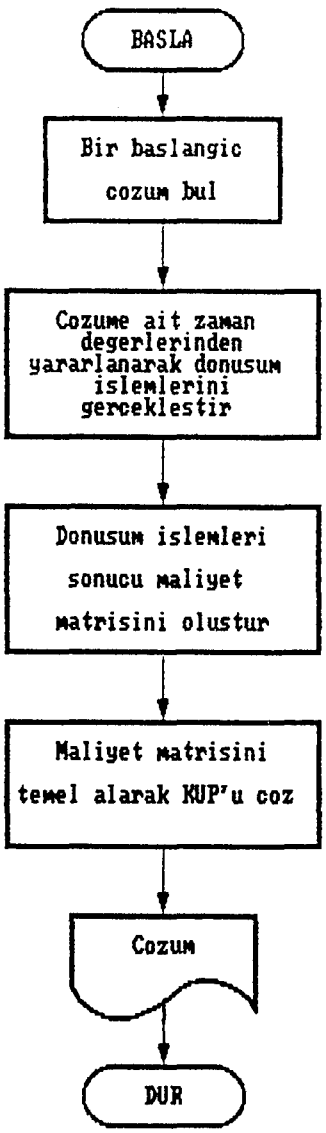
2.Adım: Çözümdeki $t_{i,j}$ değerlerine ait t' *ve* t'' değerlerini tanımla.

3.Adım: t_0, t_q, Ω_k, *ve* ω_k değerlerini hesapla.

4.Adım: $C = \{c_{i,j}\}$ matrisini kullanarak KUP'u çöz.

Kolaylıkla anlaşılabileceği gibi Grabowski'nin algoritması, bulunan bir temel uygun çözümdeki darboğaz zaman değerine eşit ve ondan daha büyük zaman değerine sahip hücrelere büyük maliyet değerleri atayarak, bu hücrelere bir daha taşıma yapılmamasını sağlamaya çalışmaktadır. Ara işlem olarak gerçekleştirilen dönüşüm işlemleri, tamamen taşıma yapılmaması gereken hücreler için büyük maliyet değerlerini elde etmeye yöneliktir. Bu dönüşüm işlemlerinin çokluğu nedeniyle algoritmanın

etkinliği azalmakta ve anlaşılması zorlaşmaktadır. Ancak kaydedilmesi gereken önemli bir nokta, yöntemin bu türe ait ilk algoritmalardan biri olmasıdır.

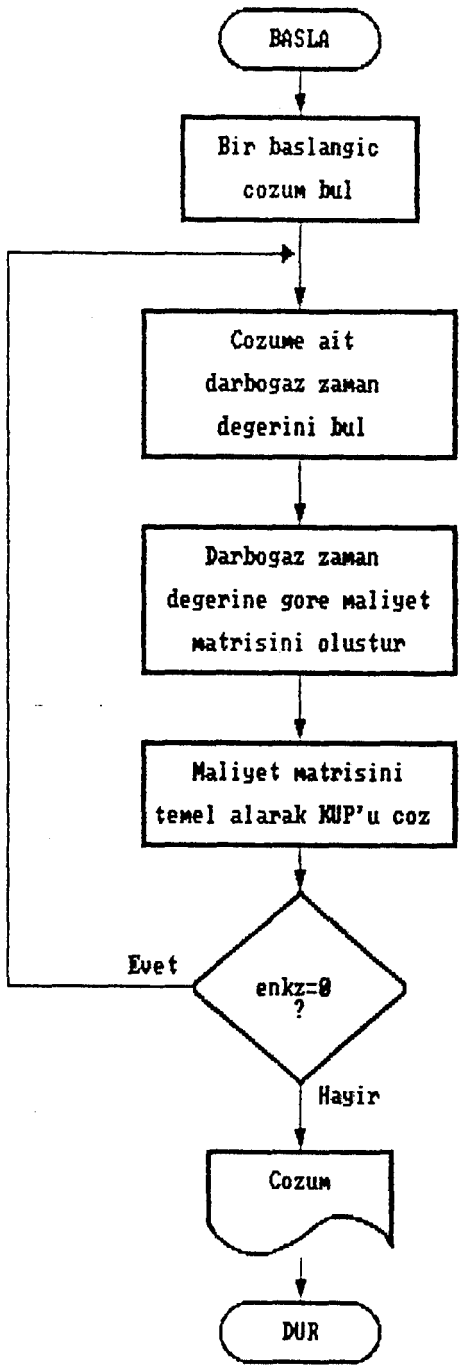


Sekil 2.2: Grabowski'nin algoritmasına ait akış şeması

2.4.2. Primal algoritma

Primal algoritma, Garfinkel-Rao tarafından geliştirilmiş ve DUP'un KUP'a dönüştürülerek çözülmesi mantığını taşıyan ikinci algoritmadır. Grabowski'nin algoritmasından farkı, dönüştürme işlemlerindeki basitlik nedeniyle çok kolay ve anlaşılır olmasıdır. Problemin çözümü, zaman matrisinin elde edilen bir çözümdeki darboğaz

zaman değerine göre indirgenerek, problemin KUP olarak çözülmesi ile sağlanmaktadır. Primal algoritmanın akış şeması şekil 2.3'de verilmiştir.



Şekil 2.3: Primal algoritmanın akış şeması

Primal Algoritma:

1.Adım: Bir başlangıç çözüm bul.

2.Adım: $DZ = enbt_{ij}\{(i,j) | x_{ij} > 0\}$ değerini hesapla.

$$c_{ij} = \begin{cases} 1 & eger\ t_{ij} = DZ \\ 0 & eger\ t_{ij} < DZ \\ M & eger\ t_{ij} > DZ \end{cases} \quad (M > 0\ ve\ cok\ buyuk)$$

3.Adım: C = $\{c_{i,j}\}$ matrisini kullanarak KUP'u çöz. Eğer amaç fonksiyonunun değeri sıfır ise ikinci adıma git, değilse dur. Eldeki çözüm eniyi çözümdür.

Primal algoritma, kolay anlaşılır olmasının yanısıra etkindir. Garfinkel-Rao sundukları makalede, rassal olarak üretilmiş problemler üzerinde yaptıkları denemeler sonucunda Primal algoritmanın, yine kendilerinin geliştirdikleri Threshold algoritmasına göre üç defa daha hızlı olduğunu belirtmişlerdir (Garfinkel and Rao, 1971). Bu nedenlerle Primal algoritma, üzerinde önemle durulması gereken algoritma özelliğini kazanmaktadır.

2.4.3. Threshold algoritması

Threshold algoritması, Garfinkel-Rao tarafından geliştirilmiştir. Fakat DUP için yapısal bir değişiklikle farklı bir çözüm yönteminin önerildiği bir algoritmadır. Threshold algoritmasında ilk olarak problemin bir serim modeli olarak ifade edilmesi ve sonradan Ford-Fulkerson tekniği ile çözüme gidilmesi önerilmektedir. Algoritma, Ford-Fulkerson tekniğini kullanması nedeniyle doğrudan doğruya probleme ait eniyi çözümü vermektedir. Bu nedenle Threshold algoritması ile çözülen bir problem için sadece başlangıç değerler ve eniyi çözüm değerleri görülebilmektedir. Ara çözümlerin elde edilemiyor olması bu algoritma için bir dezavantajdır. Threshold algoritması, Garfinkel-Rao tarafından kapasite ve talep miktarlarının tamsayı değer almadığı durumlarda da incelenmiş ve çözüm bulunmuştur. Bu da yine problemin serim modeli biçiminde ifade edilerek çözülmesinin bir sonucudur.

Threshold algoritmasında esas amaç, darboğaz zaman değerine ait bir alt sınır türeterek çözüm aramaktır. Primal algoritmada model, üstten sınırlandırılarak çözülmektedir. Yani bulunan bir başlangıç çözüme ait darboğaz zaman değeri modelin eniyi çözümünde erişilecek DZ değeri için bir üst sınır olarak kabul edilmekte ve bu değer her ardıştırmada küçültülmeye çalışılarak eniyi DZ değerine erişilmeye çalışılmaktadır. Threshold algoritmasında ise eniyi DZ değerini bulmak için ilk olarak bir alt sınır belirlenmekte, bu değer Ford-Fulkerson algoritmasında başlangıç değer olarak alınıp çözüme gidilmektedir.

DUP'un, kısıtları gözönünde bulundurulduğunda, bütün kapasite kısıtları için, i satırında bulunan zaman değerleri içinde (i=1,2,...,m), enküçük zaman değeri olan r_i, sözkonusu kapasite kısıtının sağlanması için en azından bu değer kadar zamanın geçmesi gerektiğini gösterir. O halde $\forall i$ için $r_i = \underset{j}{enk}\, t_{ij}$'dir ve r_i'ler eniyi darboğaz zaman değeri için birer alt sınırdır.

Benzer mantık bütün talep kısıtları için de yürütülebilir. O halde her j. talep kısıtının sağlanması için de (j=1,2,...,n), geçmesi gereken v_j gibi bir zaman değeri vardır. O halde $\forall j$ için $v_j = \underset{i}{enk}\, t_{ij}$'dir ve v_j'ler de eniyi darboğaz zaman değeri için birer alt sınırdır.

Probleme ait bütün kısıtların gerçeklenmesi için gerekli zaman değeri ise, elde edilen r_i ve v_j değerleri arasındaki enbüyük değer olacaktır. Bu değer, Ford-Fulkerson algoritmasını uygulayabilmek için alınacak başlangıç değerdir.

Sonuç olarak, $\underline{z}^0 = enb\{r_1, r_2, \ldots, r_m, v_1, v_2, \ldots, v_n\}$. Buradaki $\underline{z}^0$ değeri DZ için eniyi alt sınırdır.

Problemin serim modeli halinde ifadesinde, s tane kaynak, t tane de talep noktası vardır. (s,i) ayrıtının

kapasitesi a_i, (j,t) ayrıtının talebi ise b_j'dir. (i,j) ayrıtı için kapasite sınırı yoktur. Threshold algoritmasının akış şeması şekil 2.4'de verilmiştir.

Threshold Algoritması:

1.Adım: $F=-\infty$ ve $x=\underline{x}^0$.

2.Adım: $t_{ij} \leq z$ koşulunu sağlayan bütün (i,j) ayrıtlarına taşıma yapılabilecektir. Bu koşulu sağlayan ayrıtların oluşturduğu serimde Ford-Fulkerson algoritmasını uygulayarak enbüyük akış olan $\bar{F}$ değerini bul. $W=\sum a_i$ olmak üzere eğer $\bar{F}=W$ ise eniyi amaç fonksiyonu değerine erişilmiştir, 4. adıma git. Eğer $\bar{F}<W$ ise 3. adıma git.

3.Adım: $F=\bar{F}$ ve

z = enk{$t_{i,j}$, i işaretlenmiş, j işaretlenmemiş}

olarak tanımla, bu durumda serimde daha fazla akışın sağlanacağı açıktır, 2. adıma dön.

4.Adım: 2. adımda elde edilen akışa ait ayrıtlar, $\bar{X}=\{\bar{x}_{ij}\}$ olsun.

$$\bar{u}=\sum_{(i,j)|t_{ij}=DZ} \bar{x}_{ij}$$

iken,

eğer $\bar{u}=W-\bar{F}$ ise $\bar{X}$ eniyi çözümdür.

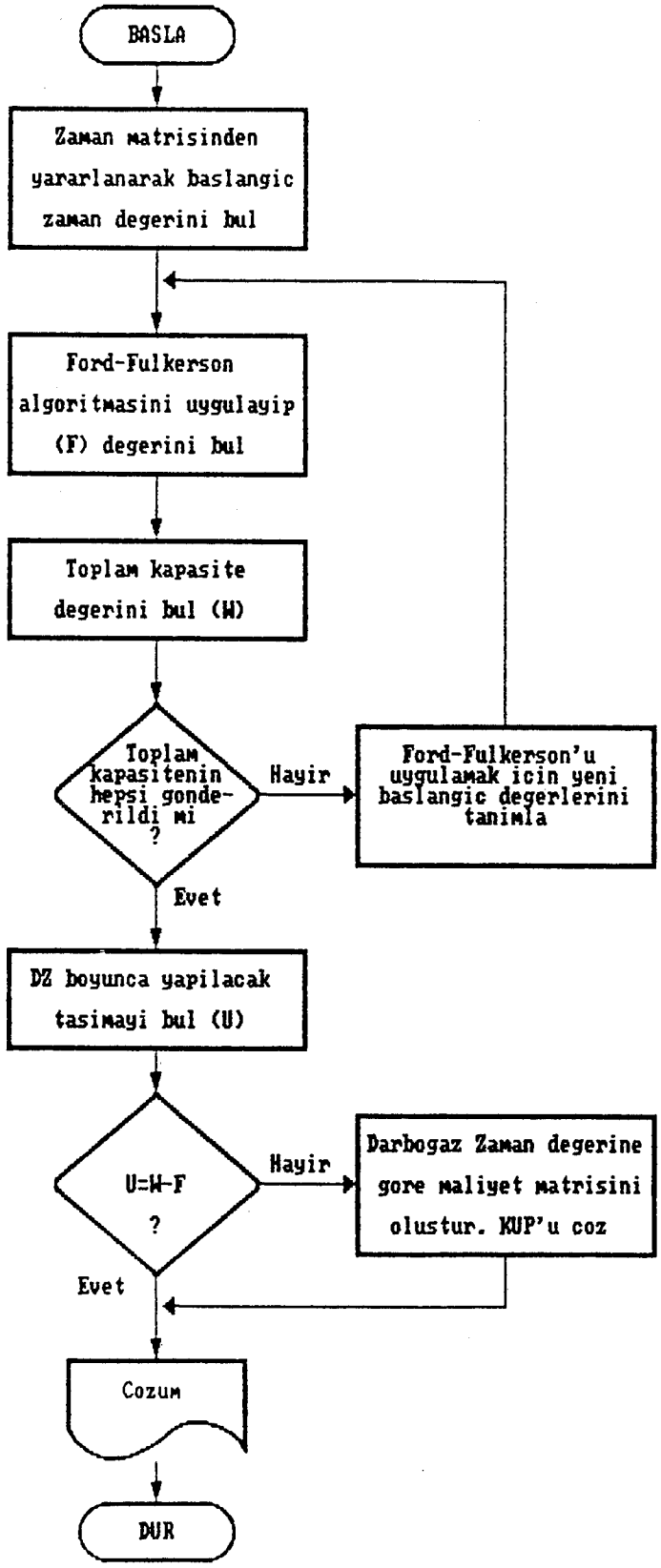
eğer $\bar{u}\neq W-\bar{F}$ ise 5. adıma git.

5.Adım: Kabul edilebilir bütün ayrıtlar için

$$c_{ij}=\begin{cases}1 & \text{eğer } t_{ij}=DZ\\ 0 & \text{eğer } t_{ij}<DZ\end{cases}$$

dönüşümünü yap. C = {$c_{i,j}$} matrisini kullanarak KUP'u çöz.

6.Adım: Elde edilen çözüm darboğaz taşımanın enküçüklendiği çözümdür.



Sekil 2.4: Threshold algoritmasının akış şeması

Bu algoritmanın en önemli özelliği hem darboğaz zamanı hem de darboğaz taşımayı enküçüklemesidir. Dikkat edilirse 2. adımın sonunda darboğaz zamanın enküçüklenmesi amacına göre, problem eniyi çözümüne erişmiş durumdadır. 4. ve 5. adımlar, bulunan eniyi darboğaz zamandaki, taşıma miktarını enküçükleyebilmek için uygulanmaktadır.

2.4.4. Srinivasan-Thompson algoritması

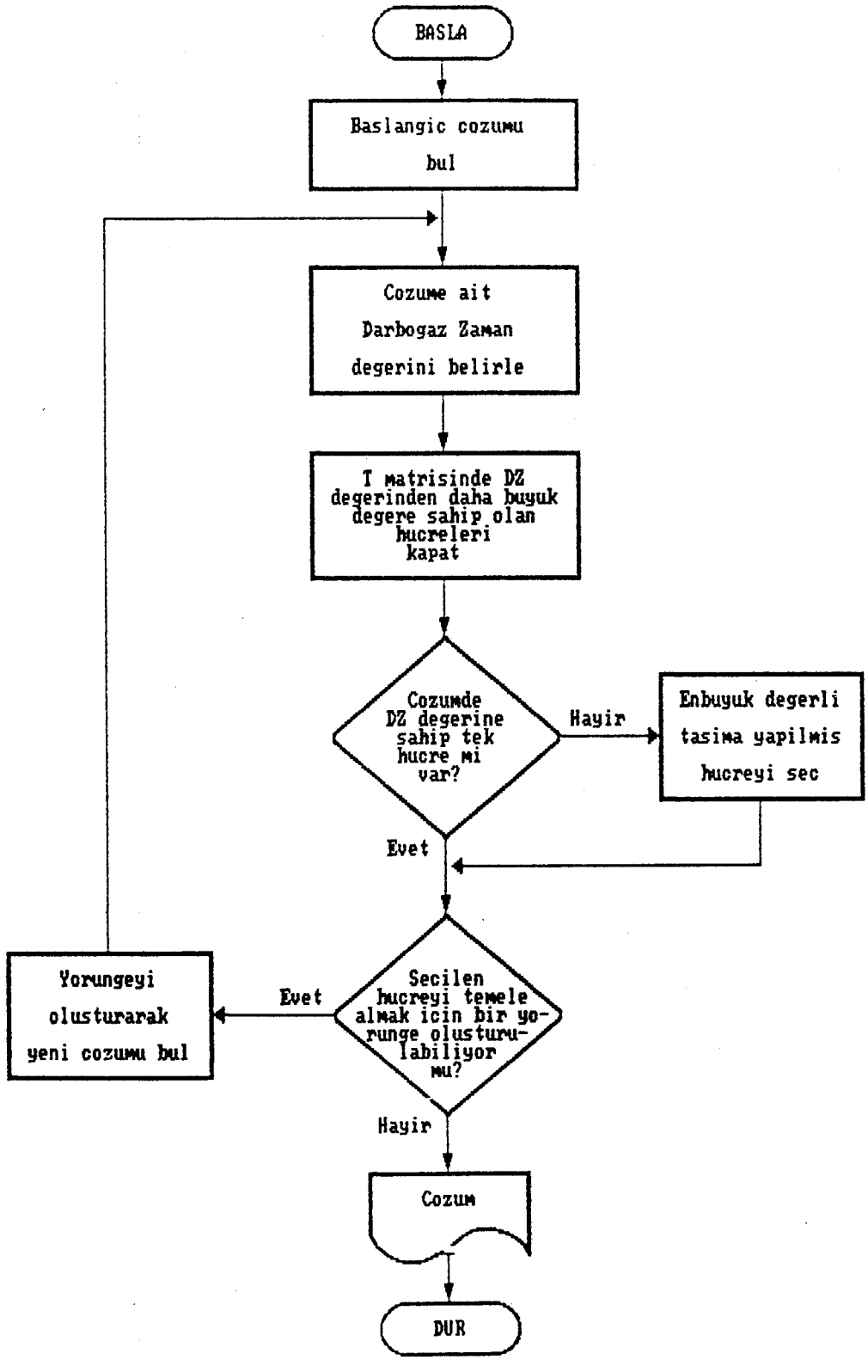
Srinivasan-Thompson tarafından geliştirilen bu algoritma, aslında araştırmacıların iki amaçlı ulaştırma problemlerini çözmek için geliştirdikleri algoritmanın özel halidir. Çözüm mantığı, DUP ile ilgilenen ilk araştırmacıların geliştirdikleri mantıkla aynıdır. Buna göre problem için bir başlangıç çözüm bulunarak bu çözümdeki DZ değeri belirlenmekte ve izleyen adımda DZ değerine sahip hücredeki taşıma, diğer hücrelere dağıtılmaya çalışılmaktadır. Bunun için de bilinen KUP çözüm tekniklerinde yoğun olarak kullanılan, yörünge çizerek taşımayı diğer hücrelere dağıtma yöntemi önerilmektedir. Bu mantık daha önce de sözü edildiği gibi Szwarc ve Hammer'ın algoritmalarındaki mantık ile eşdeğerdir. Benzer çalışmalar başka araştırmacılar tarafından da gerçekleştirilmiş ve yayınladıkları kitaplarda açıklanmıştır (Taha, 1971 ve Phillips et.al., 1976).

Algoritmanın adımlarından önce, yörüngenin oluşturulup dağıtımın nasıl yapılacağını belirleyen iki küme tanımını açıklamak gerekmektedir:

Γ_1: tek sayılı değerlerden oluşan hücrelerin kümesi

Γ_2: çift sayılı değerlerden oluşan hücrelerin kümesi

Srinivasan-Thompson algoritmasına ait akış şeması şekil 2.5'de verilmiştir.



Sekil 2.5: Srinivasan-Thompson algoritmasının akış şeması

Srinivasan-Thompson Algoritması:

1.Adım: k=1 olarak tanımla ve bir başlangıç temel uygun çözüm bul.

2.Adım: $DZ^k = \underset{\{(i,j)|x_{ij}>0\}}{enb}\ t_{ij}$ değerini hesapla.

3.Adım: $x_{pq}>0$ ve $t_{pq}=DZ^k$ olan bir $(p,q)\epsilon B$ hücresini seç. Böyle bir hücre yoksa k=k+1 olsun ve 2. adıma git. Koşulları sağlayan birden çok x_{pq} değeri varsa enbüyük x_{pq} değerine sahip olan hücreyi seç, d=1 olarak al. $\{x_{ij}\}$'nin geçerli çözümlerini $\{x_{ij,1}\}$, B'deki temel değişkenleri ise B_1 göstersin.

4.Adım: $\Psi=\gamma-B$ kümesini tanımla. Eğer Ψ kümesinde $t_{ij}<DZ^k$ eşitsizliğini sağlayan bir (e,f) hücresi yoksa 9. adıma git. Diğer durumlar için $t_{ef}<DZ^k$ ve $(e,f)\epsilon\Psi$ olan bir (e,f) hücresini seç.

5.Adım: (e,f)'nin B_d'ye eklenmesiyle oluşan Γ çevrimini oluştur. Eğer $x_{r=s,d}=0$, $t_{rs}\geq DZ^k$ şartlarını sağlayan ve $(r,s)\epsilon\Gamma_2$ olan bir (r,s) hücresi varsa 8. adıma git yoksa 6. adıma git.

6.Adım: $\chi=\{(g,h)\}$ kümesi, $(i,j)\epsilon\Gamma_1$ için $(x_{ij,d})$'nin enküçük değerine sahip hücrelerin kümesi olsun. Eğer $(p,q)\epsilon\chi$ ise (r,s)=(p,q) olarak seç. Diğer durumlarda χ'den rassal olarak bazı (r,s)'leri seç. $\Delta=x_{rs,d}$ olsun ve

$$x_{ij,d+1}=\begin{cases} x_{ij,d}-\Delta & (i,j)\epsilon\Gamma_1 \quad ise \\ x_{ij,d}+\Delta & (i,j)\epsilon\Gamma_2 \quad ise \\ x_{ij,d} & diger\ durumlar \end{cases}$$

kümesini oluştur.

7.Adım: $B_{d+1}=B_d+\{(e,f)\}-\{(r,s)\}$ olarak al. Eğer (p,q)=(r,s) ise $B=B_{d+1}$ olsun ve 3. adıma git. Diğer durumlar için d=d+1 olsun ve 4. adıma git.

8.Adım: $X_{d+1}=X_d$, $B_{d+1}=B_d+\{(e,f)\}-\{(r,s)\}$, d=d+1, 4. adıma git.

9.Adım: Geçerli çözüm $X=X_d$ ve eniyi darboğaz zaman BT^k'dır.

2.4.5. Heinz Isermann algoritması

Darboğaz ulaştırma probleminde amaç fonksiyonu $enkDZ = enbt_{ij}$ $\{(i,j)|x_{ij}>0\}$'dir. Buradaki doğrusal olmayan amaç fonksiyonu doğrusallaştırılarak kısıtların da aynı kalmasıyla standart ulaştırma problemi (Lexicographic Transportation Problem-LTP) tanımlanır. Buna göre yeni amaç fonksiyonu $Lexminz = \sum_i \sum_j d_{ij}.x_{ij}$'dir. Isermann'ın algoritmasına ait akış şeması şekil 2.6'da verilmiştir.

Isermann'ın Algoritması:

1.Adım: LTP için bir başlangıç temel uygun çözüm bul.

2.Adım: $enkDZ = enbt_{ij}$ $\{(i,j)|x_{ij}>0\}$ eşitliği uyarınca DZ değerini bul ve T matrisinden DZ'ye en yakın ve küçük olan $\bar{t}$ değerini alt sınır olarak belirle. $(\bar{t}<DZ)$

$\gamma_1 = \{(i,j)|t_{ij} \geq DZ\}$

$\gamma_2 = \{(i,j)|t_{ij} = DZ\}$

$\gamma_3 = \{(i,j)|t_{ij} = \bar{t}\}$

$\gamma_4 = \{(i,j)|t_{ij} < \bar{t}\}$

olarak alt kümelere ayır. γ_1 alt kümesi için birim vektör e_1, γ_2 alt kümesi için birim vektör e_2,..., γ_4 alt kümesi için birim vektör e_4'dür. Bu ilişkilerden yararlanarak $D=\{d_{ij}\}$ matrisini tanımla.

$d_{ij} = e_k$, $(i,j) \in \gamma_k$ $k=1,2,3,4$

3.Adım: Temelde yer alan değişkenler için $d_{ij} = u_i + v_j$ eşitliği gereğince $\bar{u}_i, \bar{v}_j$ değerlerini bul.

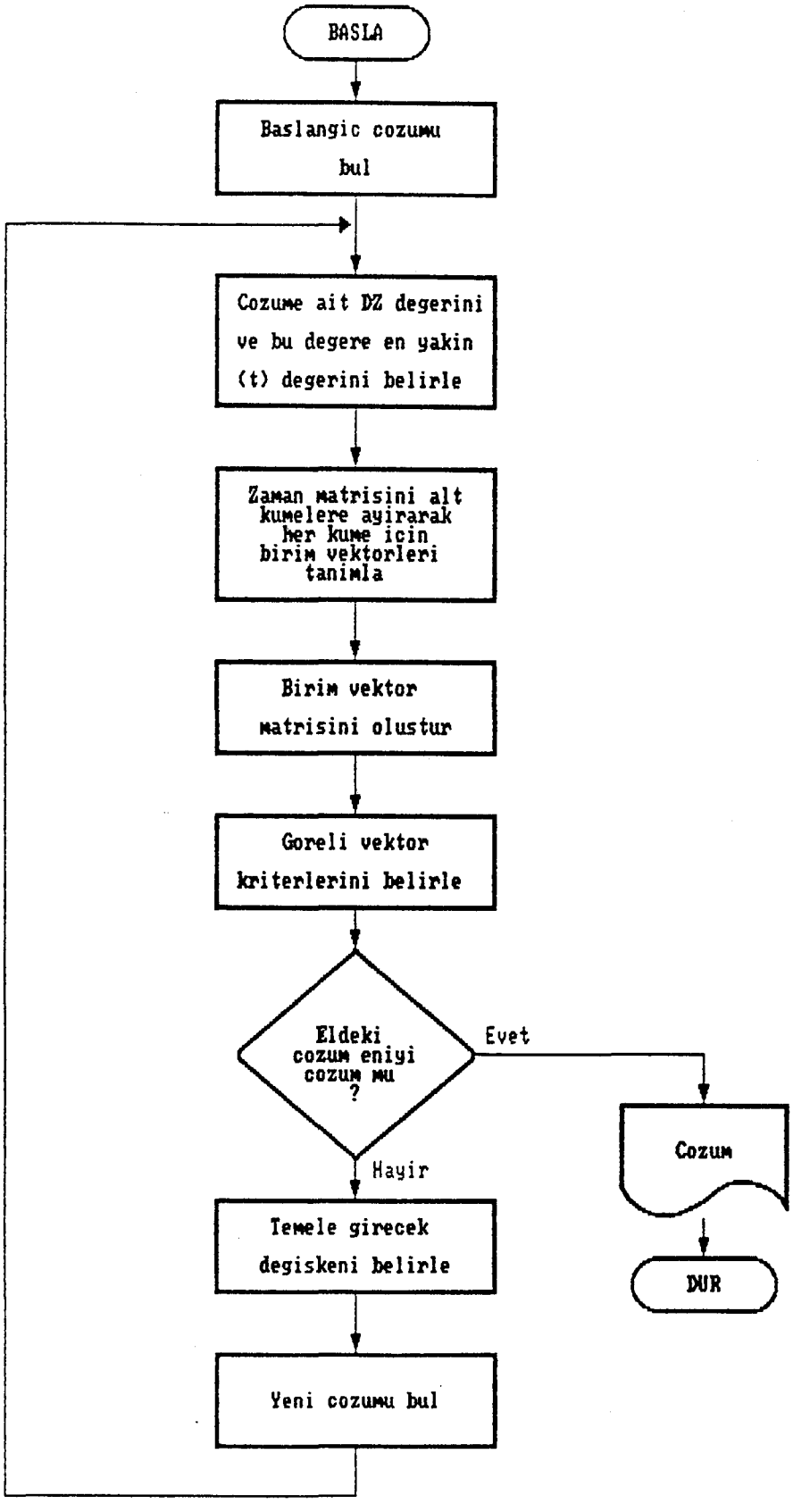
4.Adım: Temelde yer almayan değişkenler için $z_{ij} = d_{ij} - \bar{u}_i - \bar{v}_j$ göreli vektör kriterlerini bul.

5.Adım: Eğer temelde yer almayan bütün değişkenler için z_{ij}, sıfır vektörüne eşit veya ondan büyükse eldeki çözüm eniyi çözümdür, 7. adıma git. Eğer z_{ij}, sıfırdan küçükse 6. adıma git.

6.Adım: Temelde yer almayan değişkenler için $z_{ij} \leq 0$ koşulunu sağlayan enküçük, z_{k+1} değerini seç, x_{k+1} değişkenin temele alarak yeni çözümü bul ve 3. adıma git.

7.Adım: $\overline{X}=\{\bar{x}_{ij}\}$, LTP için eniyi çözümdür.

Isermann, algoritmasında ikillikten yararlanmıştır. Çözümün mantığında, temelde yeralması gereken değişken için $c_{ij} \geq u_i+v_j$ eşitsizliğinin sağlanması gerekliliği yatmaktadır. Ulaştırma modelinin ikillikle olan bu ilişkisinden yararlanılarak çözüm bulunmaktadır.



Sekil 2.6: Isermann'ın algoritmasına ait akış şeması

3. İKİ AMAÇLI ULAŞTIRMA PROBLEMLERİ

Darboğaz ulaştırma problemi (DUP), bir taşıma programının uygulanması sırasında, en uzun taşıma süresinin olabildiğince kısaltılmasını ve bu nedenle darboğazın oluştuğu noktanın eniyilenmesi amacını taşımaktadır. Oysa kolay bozulabilen besin maddelerinin taşınması probleminde taşıma zamanının önemi kadar, taşımanın işletmeye getirdiği maliyetin de önemi vardır. Böyle bir durumda problemi sadece darboğaz zamanın enküçüklenmesi biçiminde ele almak doğru olmaz. Benzer şekilde aynı pazarı paylaşan işletmelerin, piyasadaki yerlerini koruyabilmeleri için hem zaman hem de maliyet yönlü eniyilenmeye gereksinim duyacakları söylenebilir. Elbette ki zamanın mutlak derecede önemli olduğu durumlar vardır. Afet sonrası yardım sevkiyatları ve savaş zamanındaki sevkiyatlar, buna iyi birer örnektir. Fakat unutulmaması gereken önemli bir nokta, afet ve savaş nedeniyle yapılacak sevkiyatlar ile sıkça karşılaşılmayacağıdır. Başka bir deyişle, günümüzde iki amaçlı ulaştırma problemleri ile DUP'a göre daha sık karşılaşılacağı söylenebilir. Özellikle artan maliyetler ve yoğun rekabet koşulları, işletmeleri çok yönlü düşünmeye ve çok yönlü eniyilemeye zorlamaktadır. İşte bu nedenlerle, hem zaman hem de maliyet eniyilenmesinin inceleneceği modellere gereksinim duyulmuş ve ilk kez Srinivasan-Thompson tarafından iki amaçlı ulaştırma problemleri tanımlanarak çeşitli çözüm yaklaşımları önerilmiştir.

Bu bölümde, iki amaçlı ulaştırma problemlerinin genel yapısı, çeşitleri ve çözüm yaklaşımlarından bazıları incelenmektedir.

3.1. İki Amaçlı Ulaştırma Problemlerinin Genel Yapısı

En genel anlamda üretim merkezlerinden dağıtım bölgelerine gerçekleştirilecek olan taşıma programlarının belirlenmesi probleminde aşağıdaki amaçlardan sözedilebilir.

i- Toplam Maliyetin (TM) enküçüklenmesi : Bu, bilinen klasik ulaştırma problemidir.

ii- Darboğaz Zamanın (DZ) enküçüklenmesi : Bu problem, darboğaz ulaştırma problemi olarak tanımlanmış ve birinci bölümde ayrıntılarıyla incelenmiştir.

iii- Darboğaz Taşımanın enküçüklenmesi : Bu problem ise tek başına incelenmemiş, genellikle darboğaz ulaştırma probleminde, eniyi darboğaz zamana göre çözüm bulunduktan sonra ikincil amaç olarak, darboğaz taşımanın da enküçüklenmesine çalışılmıştır. Aslında darboğaz taşımanın enküçüklenmesi oldukça önemlidir. Çünkü bunun sağlanması ile, DUP olarak çözülen modelin sonuçlarına göre, darboğaz zaman içinde, planlanandan daha fazla ürün, yerine ulaştırılmış olacaktır. Bu özel durum ise, taleplerinin çeşitli nedenlerle bir an önce karşılanması gereken dağıtım bölgeleri için önem taşımaktadır.

Bu tanımların yol göstermesiyle, birden çok amacı bulunan bir ulaştırma problemi için dengelenmiş model aşağıdaki gibi verilebilir:

i indisi, üretim merkezlerini (satırlar),

j indisi, dağıtım bölgelerini (sütunlar),

göstermek üzere I ve J aşağıdaki gibi tanımlanır.

$I=\{i|i=1,2,\ldots,m\}$ ve $J=\{j|j=1,2,\ldots,n\}$

a_i : i. üretim merkezinin kapasitesi,

b_j : j. dağıtım bölgesinin talebi,

x_{ij} : i. üretim merkezinden j. dağıtım bölgesine fiziksel olarak taşınacak miktar,

t_{ij}: i. üretim merkezinden j. dağıtım bölgesine yapıcak taşımanın süresi,

c_{ij}: i. üretim merkezinden j. dağıtım bölgesine yapıcak taşımanın maliyeti,

iken dengelenmiş model;

$$\sum_{j=1}^{n} x_{ij} = a_i \qquad i = 1,2,\ldots,m$$

$$\sum_{i=1}^{m} x_{ij} = b_j \qquad j = 1,2,\ldots,n$$

$$x_{ij} \geq 0 \ , \ t_{ij} \geq 0 \ , \ a_i > 0 \ , \ b_j > 0$$

kısıtları altında,

$$enk\,TM = \sum_{i} \sum_{j} c_{ij} . x_{ij}$$

$$enk\,DZ = \underset{\{(i,j) | x_{ij} > 0\}}{enb} t_{ij}$$

$$enk\,DT = \sum_{(i,j) | t_{ij} = DZ} x_{ij}$$

şeklinde yazılır.

İki amaçlı ulaştırma problemi kavramından ise, aynı kısıtlar altında farklı iki amacın aynı anda eniyilenmeye çalışılması anlaşılmaktadır. Bu tarif gereği Srinivasan-Thompson tarafından iki tür, iki amaçlı ulaştırma problemi tanımlanmaktadır.

i- TM ve DZ tabanlı ulaştırma problemi

ii- TM ve DT tabanlı ulaştırma problemi

Dikkat edilebileceği gibi Srinivasan-Thompson tarafından DZ ve DT tabanlı iki amaçlı bir ulaştırma problemi tanımlanmamıştır. Bunun nedeni daha önce de açıklandığı gibi, DUP içinde darboğaz taşımanın da enküçüklenmeye çalışılmış olması, bir anlamda bu tür iki amaçlı ulaştırma probleminin önceden tanımlanarak çözümüne ait çalışmaların bulunmasıdır.

Toplam maliyeti enküçükleme kriterinin bulunduğu iki amaçlı ulaştırma problemlerinde dikkati çeken önemli bir nokta, aynı problem için tanımlanacak maliyet ve zaman matrislerinin birbirleriyle ilişkili olup olmayacağıdır. İlk bakışta üretim merkezi ile dağıtım bölgesi arasındaki mesafe büyükse, taşıma zamanının da büyük olacağı ve buna bağlı olarak taşıma maliyetinin büyük bir değer alacağı düşünülebilir. Aynı mantıkla, kısa mesafeler söz konusu

olduğunda, taşıma zamanının ve taşıma maliyetinin ilişkili olacağı, böylece problemi iki amaçlı ulaştırma problemi olarak çözmek yerine sadece KUP gibi çözmenin yeterli olacağı düşünülebilir. Fakat bu düşünce yanıltıcıdır. Nedeni, günümüzde çok çeşitli taşıma araçlarının bulunması, her taşıma aracı için taşıma zamanının ve taşıma maliyetinin farklı olmasıdır. Ayrıca her taşıma aracı için yükleme ve boşaltmada kullanılacak işçi sayısı da aynı olmayacağı için işçilik maliyetleri de farklı olacaktır. Bu da, toplam maliyet üzerinde etkili olur. Dolayısıyla, aynı problem için tanımlanacak zaman ve maliyet matrisleri arasında büyük bir ilişkinin olması beklenmemelidir. Bu nedenle de problemi sadece KUP veya sadece DUP gibi çözmek, eniyi toplam taşıma maliyetinin bulunduğu veya eniyi darboğaz zaman değerinin belirlendiği çözümlerin elde edilmesini sağlar, ama her iki amaç fonksiyonun aynı anda ve aynı önemde gözönünde bulundurulduğu anlamına gelmez. Oysa iki amaçlı bir ulaştırma probleminde amaç, iki farklı amaca da aynı derecede önem verilerek, mümkünse her iki amacı birden eniyileyen çözümü belirlemektir. İşte bu nedenlerden dolayı iki amaçlı ulaştırma problemleri tanımlanmış ve farklı çözüm teknikleri geliştirilmeye çalışılmıştır.

İki amaçlı bir ulaştırma problemi için tanımlanan zaman ve maliyet matrislerinin birbirleriyle ilişkili olmaması TM-DZ tabanlı bir ulaştırma probleminde, amaçların birbirleriyle çelişmesine neden olacaktır. Çünkü her iki amaç fonksiyonu da enküçük değerleri bulmayı hedeflemektedir. Bu durumda, modelin birinci amaç fonksiyonuna göre enküçüklenirken, diğerine göre de enküçüklenmesini beklemek yanlış olur. Fakat bu noktada iki amaçlı ulaştırma problemleri için hayati önem taşıyan, sıralı çözüm kavramı ortaya çıkmaktadır. Sıralı çözümler, aynı problem için elde edilen ardışık ve komşu çözümlerdir. İki amaçlı bir ulaştırma probleminde, her iki amacı da eniyileyen bir tek çözüm bulunamayacağı için de, probleme ait sıralı çözümler belirlenerek, çözümlere karşı gelen ve

(TM,DZ) ikililerinden oluşan sıralı çözüm ikilileri belirlenmekte, ve bunlar işletmenin stratejisine göre bir karar verebilmesi için karar vericiye sunulmaktadır.

Srinivasan-Thompson ikilisi çalışmalarında (Srinivasan and Thompson, 1976) özellikle sıralı çözümlerin nasıl bulunacağını ve elde edilen sıralı çözüm ikilileri arasından üstün çözümlerin nasıl belirleneceğini saptamaya çalışmışlar ve TM-DZ tabanlı bir ulaştırma problemi için 3 ayrı algoritma önermişlerdir. Aynı çalışmada TM-DT tabanlı bir ulaştırma problemi için de 2 algoritma önerilmektedir.

3.2. İki Amaçlı Ulaştırma Problemlerinin Tarihsel Gelişimi

İki amaçlı ulaştırma probleminin tanımlanarak çözülmesine ilişkin ilk çalışmalar Srinivasan-Thompson tarafından 1976 yılında gerçekleştirilmiştir. Srinivasan-Thompson sözü edilen bu ilk çalışmalarında, bir ulaştırma problemi için üç farklı amaç fonksiyonu tanımlamakta ve iki amaçlı ulaştırma problemlerini ise TM-DZ ve TM-DT tabanlı olmak üzere ikiye ayırmaktadır. Ayrıca tanımlanan bu yeni problem türleri için de çeşitli algoritmalar önermektedir. Aslında DUP'un incelenmesi sırasında Hammer, Szwarc ve Garfinkel-Rao ikilisi tarafından darboğaz zamanın eniyilenmesi ile birlikte, darboğaz taşımanın da eniyilenmesine ilişkin çalışmalar yapılmıştır. Ancak bu çalışmaların ortak özelliği öncelikle darboğaz zamanı eniyilemeye çalışmak ve bu sağlandıktan sonra mümkün olursa darboğaz taşımayı da enküçüklemektir. Yani darboğaz taşıma ikincil amaç olarak ele alınmakta, darboğaz zamanın eniyilenmesi amacı ile eşdeğerde görülmemektedir. Bu nedenle, gerçekleştirilen bu ilk çalışmaları, iki amaçlı ulaştırma problemi olarak adlandırmak yanlıştır. Bu konudaki ilk ve en kapsamlı çalışma Srinivasan-Thompson ikilisine aittir.

İki amaçlı ulaştırma problemlerine ilgi gösteren bir diğer araştırmacı da H. Isermann'dır. Isermann 1984 yılında yayınladığı makalesinde ilk olarak darboğaz ulaştırma problemini incelemekte ardından TM-DZ tabanlı ulaştırma problemi için bir algoritma önermektedir. Isermann çalışmasında doğrusal olmayan darboğaz zamanı enküçükleme amacını, ilk olarak doğrusallaştırmayı önermekte ve sonra problemi her iki amacı da doğrusal olan iki amaçlı bir ulaştırma problemi olarak ele almaktadır.

3.3. İki Amaçlı Ulaştırma Problemleri İçin Geliştirilen Çözüm Teknikleri

Bu bölümde, konuyla ilgili en yoğun çalışmaları gerçekleştiren Srinivasan-Thompson ikilisinin ve Isermann'ın çalışmaları incelenmektedir. Yöntemlerin anlaşılırlığını arttırmak amacıyla, algoritmanın adımlarından önce makro akış şemaları verilmiştir. Darboğaz zaman gibi ortak bazı kavramlar dışında, yine yazarların kullandıkları gösterimlerin korunmasına özen gösterilmiştir. Burada;

DZ: Herhangi bir çözümdeki darboğaz zaman,
T: Taşıma zamanlarını gösteren matris,
C: Taşıma maliyetlerini gösteren matris,
B: Herhangi bir çözümde, temelde yer alan karar değişkenlerine ait (i,j) hücrelerinin bulunduğu küme $B=\{(i,j)|x_{ij}>0\}$,
X: Herhangi bir çözümde karar değişkenlerinin bulunduğu küme $X=\{x_{ij}|(i,j)\in B\}$,
γ: Zaman matrisindeki (i,j) hücrelerinin tamamını içeren küme,

olarak ele alınacaktır.

3.3.1. TM-DZ için 1. Srinivasan-Thompson algoritması

Bu algoritma, sıralı temel uygun çözümler bulup buna bağlı olarak üstün sıralı çözüm ikililerinin belirlenmesini sağlamaya çalışır. Yazarlara göre izlenecek yol, önce KUP'u

çözmek, bu çözümden elde edilen darboğaz zaman değerine göre eldeki maliyet matrisini indirgemek ve yeniden KUP'u çözerek komşu bir başka çözüme erişmektir. Bu algoritmadaki önemli bir nokta KUP'un çözümü için Ford-Fulkerson metodunun önerilmesidir. Algoritmanın akış şeması şekil 3.1'de verilmiştir.

Algoritma:

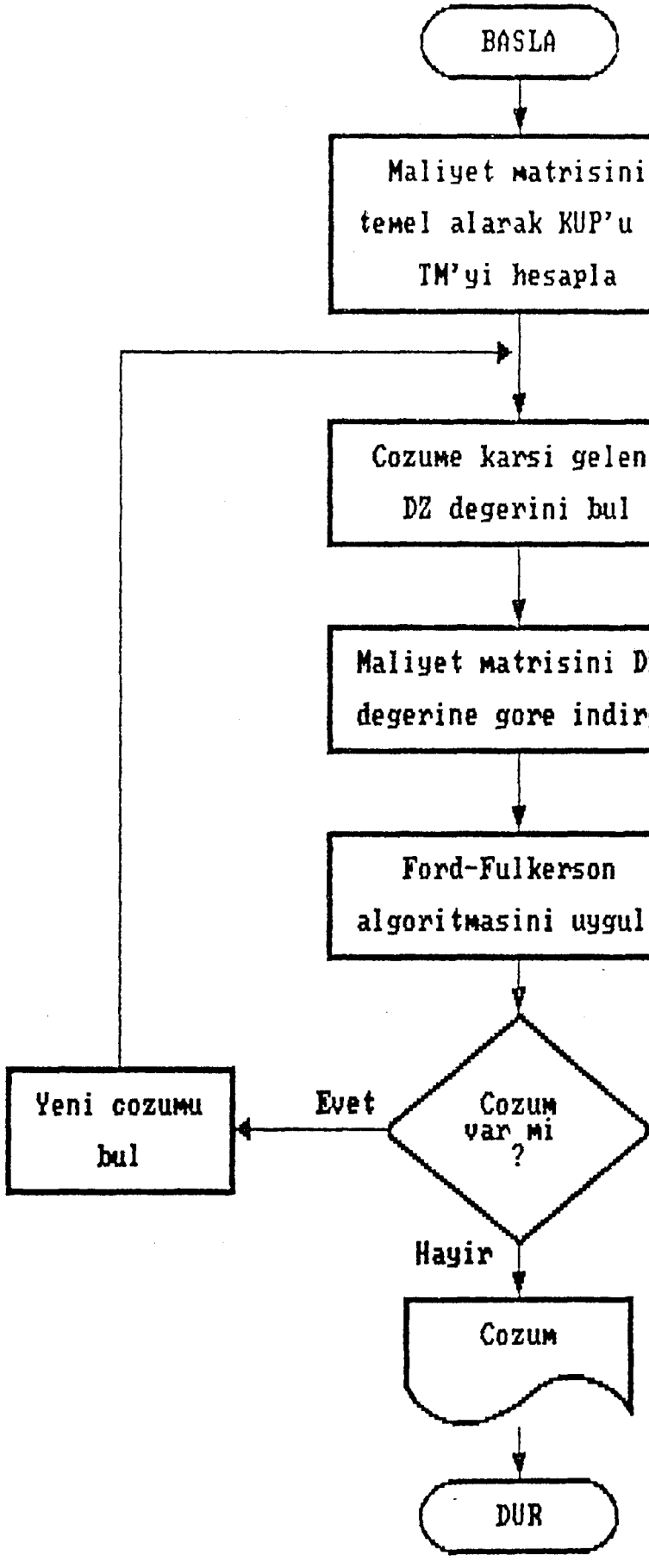
1.Adım: k=1. KUP'u çöz. Bulunan toplam maliyet değerine TM^1 adını vererek elde edilen çözüme zaman matrisinde karşı gelen darboğaz zaman değerini belirle ve DZ^1 ismini ver. $DZ=enbt_{ij}\{(i,j)|x_{ij}>0\}$

2.Adım: a-) $\Pi=\{(i,j)|t_{ij}\leq DZ^k\}$ kümesini tanımla.

b-) Π kümesinden yararlanarak KUP'u çöz. Eğer çözüm bulunamıyorsa 4. adıma git. Çözüm varsa TM'yi bul ve 3. adıma git.

3.Adım: k=k+1 , TM^k eniyi maliyettir. DZ^{k+1}'yı hesapla 2. adıma dön.

4.Adım: Uygun çözüm ikilileri kümesi $\{(TM^1,DZ^1),(TM^2,DZ^2),\ldots,(TM^k,DZ^k)\}$ dır. j>i iken $TM^i=TM^j$ koşulunu sağlayan çözüm ikilileri varsa (TM^i,DZ^i) ikilisini kümeden çıkar. Geriye kalan elemanlardan oluşan küme üstün sıralı çözüm ikililerinin bulunduğu kümedir.



Sekil 3.1: TM-DZ için 1. Srinivasan-Thompson Algoritmasının Akış Seması.

Srinivasan-Thompson tarafından, bu algoritmaya çok benzeyen ve sadece algoritmanın 2. adımında değişiklik öneren bir başka çözüm yaklaşımı daha geliştirilmiştir. Yenilik KUP'un çözümünde Ford-Fulkerson tekniğinin kullanılması yerine, bilinen MODI metodunun kullanılmasıdır. Ayrıca çözüm sonucunda bulunan darboğaz zamandaki taşıma miktarı da, yörünge çizme ilkeleri kullanılarak diğer hücrelere dağıtılmaya çalışılmaktadır. Kısaca darboğaz zaman değerine sahip taşıma miktarı sıfırlanmaya çalışılmaktadır. Aslında öneri, 2. bölümde sözü edilen Srinivasan-Thompson algoritması ile aynı öneridir. Yörüngeleri çizerek taşımaların dağıtılması, çok etkin bir yöntem değildir. Yöntemin bu algoritmada kullanılması da etkinliğini arttırmamaktadır Fakat, yukarda açıklanan algoritmaya göre darboğaz taşıma miktarının da mümkün olduğunca enküçüklenebilmesi gibi bir üstünlük sağlamaktadır. Dikkat edilmesi gereken bir diğer nokta da algoritmanın üç amacı da eniyilemeye çalışıyor olması, fakat darboğaz taşıma amacının yine diğerleriyle eşit önemde olmamasıdır. Dolayısıyla, üç amaçlı ulaştırma probleminin incelendiğini söylemek doğru olmaz.

Srinivasan-Thompson tarafından önerilen bu yeni algoritmanın sadece 2. adımını açıklamak yeterli olacaktır. (Algoritmanın 1.,3. ve 4. adımların yukarda açıklanan algoritma ile aynıdır).

Algoritma:

2.Adım: a-) $\Pi=\{(i,j)|t_{ij}\leq DZ^{*}$ ve $x_{ij}=0\}$

b-) $t_{pq}=DZ^{*}$ ve $x_{pq}>0$ olan bir (p,q) hücresini seç. Eğer bu koşulları sağlayan hiçbir (p,q) hücresi bulunamıyorsa 3. adıma git. Diğer durumlar için 2. adımın c-)'sine git.

c-) Π kümesindeki elemanların arasından TM değerinin eniyiliği bozulmayıncaya kadar x_{pq}'yu sıfıra doğru sür. Eğer bu sağlanamıyorsa 4. adıma git. Diğer durumlar için ise $\Pi=\Pi\cup\{p,q\}$ ve 2. adımın b-)'sine git.

Srinivasan-Thompson tarafından önerilen bu iki algoritmada dikkat edilmesi gereken nokta KUP'un maliyet matrisine göre çözülüyor olmasıdır. Srinivasan-Thompson tarafından, bu algoritmaların hesaplama yönlü etkinlik açısından aralarında bir fark bulunmadığı belirtilmektedir. Fakat açıklanan bu algoritmalardan ikinci adımı değiştirilmiş olanı, $C=\{c_{i,j}\}$ matrisi sıfır olarak alındığında Szwarc ve Hammer tarafından önerilen DUP'u çözme algoritmalarının aynısını vermektedir. (Srinivasan and Thompson, 1976)

3.3.2. TM-DZ için 2. Srinivasan-Thompson algoritması

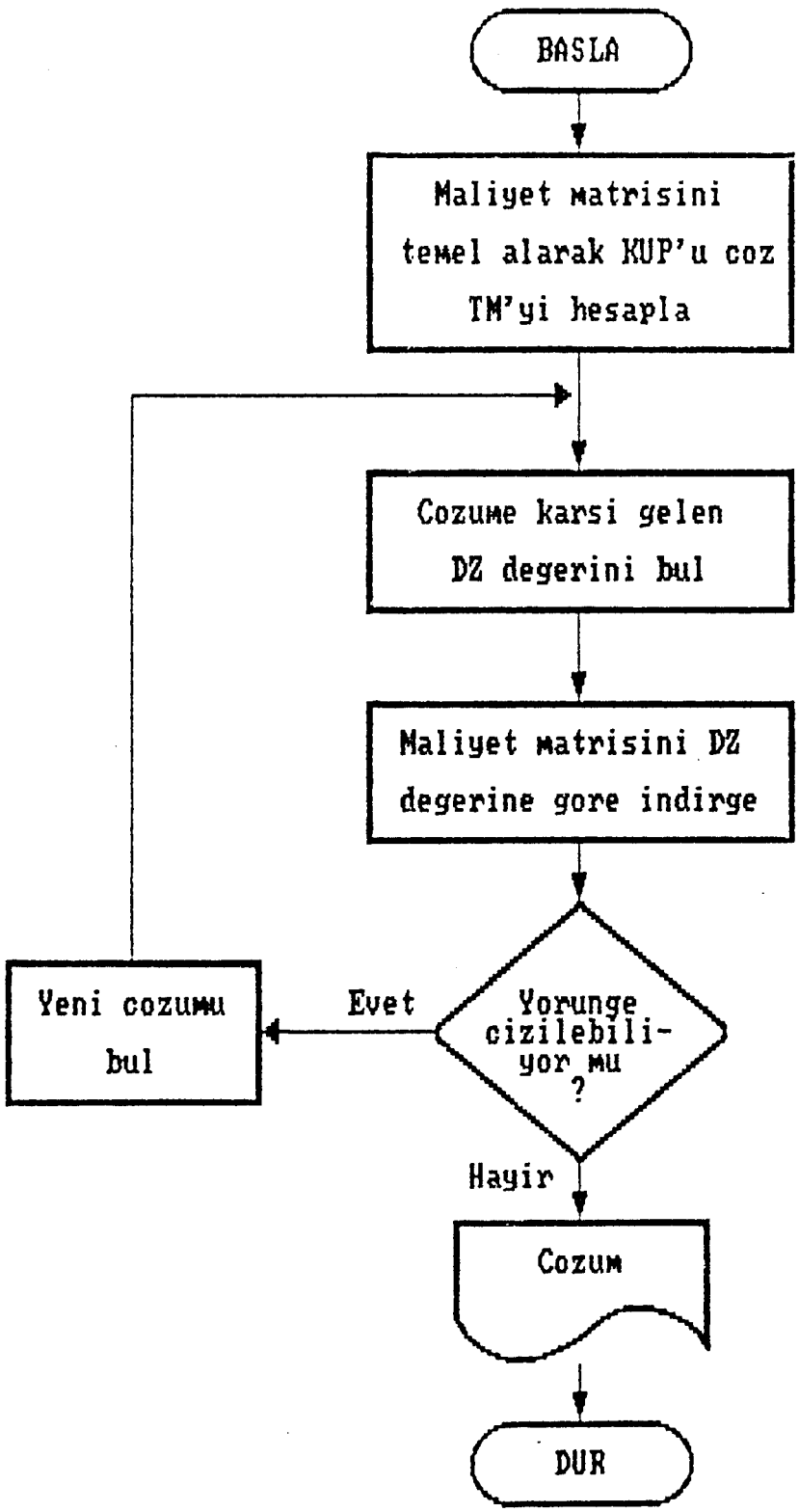
Srinivasan-Thompson'un önerdiği 2. algoritma, temelde 3.3.1'de açıklanan algoritma ile özdeştir. Aradaki fark, 2. adımın c-) şıkkındaki x_{pq} değerinin sıfırlanmasına ilişkin daha açık bir yöntemin tarif edilmiş olmasıdır. Yine bu algoritmanın önemli bir özelliği, ikillikten yararlanılmasıdır.

Algoritmanın adımlarından önce, yörüngenin oluşturulup dağıtımın nasıl yapılacağını belirleyen iki küme tanımını açıklamak gerekmektedir:

Γ_1: tek sayılı değerlerden oluşan hücrelerin kümesi

Γ_2: çift sayılı değerlerden oluşan hücrelerin kümesi

Yönteme ilişkin akış şeması şekil 3.2'de verilmiştir.



Sekil 3.2: TM-DZ için 2. Srinivasan-Thompson Algoritmasının Akış Seması.

Algoritma:

1.Adım: k=1, KUP'u çöz. Çözümde temelde yer alan değişkenleri B kümesi olarak tanımla eniyi çözüm değerini de TM^1 olarak al.

2.Adım: Başlangıç olarak $TM^{k+1}=TM^k$ eşitliğini sağla ve DZ^k değerini bul. C^k maliyet matrisinden yararlanarak aşağıdaki kurallar gereğince C^{k+1} maliyet matrisini oluştur.

$$c_{ij}^{k+1}=\begin{cases} M & eğer\ t_{ij}\geq DZ^k\ ve\ (i,j)\in B\ ise \\ c_{ij}^k & diğer\ durumlar \end{cases}$$

3.Adım: $x_{pq}>0$ ve $t_{pq}=DZ^k$ olan bir (p,q) hücresini seç. Eğer böyle bir hücre yoksa k=k+1 ve 2. adıma dön. Eğer seçmek için birden fazla sayıda hücre varsa enbüyük x_{pq} değerli olanı seç. $\delta=0$ ve d=1 olarak tanımla. $\{x_{i,j}\}$, $\{u_i\}$ ve $\{v_j\}$'nin geçerli değerleri olarak $\{x_{i,j,1}\}$, $\{u_{i,1}\}$, $\{v_{j,1}\}$ işaretle, yeni temel değişkenleri B_d olarak al.

4.Adım: (p,q)'ya ve B_d'ye karşı gelen I_R, J_R, I_C, J_C kümelerini tanımla.
$\Psi=\{(i,j)\in\{I_R\times J_C\}-\{(p,q)\}\quad c_{ij}^{k+1}\neq M\}$ olarak tanımla. Eğer Ψ boş küme ise 10. adıma git. Değilse, $\mu_d=\underset{(i,j)\in\Psi}{enk}\{c_{ij}^{k+1}-u_{i,1}-v_{j,1}\}$ değerini hesapla ve enküçüğün alındığı (e,f) hücresini kaydet.

5.Adım:
$$u_{i,d+1}=\begin{cases} u_{i,d}+\mu_d & i\in I_R \\ u_{i,d} & i\in I_C \end{cases}$$

$$v_{j,d+1}=\begin{cases} v_{j,d}-\mu_d & j\in J_R \\ v_{j,d} & j\in J_C \end{cases}$$

c_{pq}^{k+1}'yi $c_{pq}^{k+1}+\mu_d$ olarak yeniden tanımla ve $\delta=\delta+\mu_d$ kümesini oluştur.

6.Adım: (e,f)'nin B_1'ye eklenmesiyle oluşan Γ çevrimini tanımla. Eğer $x_{r,s}=0$, $t_{rs}\geq DZ^k$ şartlarını sağlayan ve $(r,s)\in\Gamma_2$ olan bir (r,s) hücresi varsa 9. adıma git yoksa 7. adıma git.

7.Adım: $x_{gh,d}$: $(i,j)\in\Gamma_1$ için $(x_{ij,d})$'nin enküçük elemanı olmak üzere $\chi=\{(g,h)\}$ kümesini tanımla (r,s)=(p,q) ise $(p,q)\in\chi'$ yi seç. Diğer durumlarda rassal olarak χ'den bir (r,s)'yi seç. $\Delta=x_{rs,d}$ olsun ve

$$x_{ij,d+1}=\begin{cases} x_{ij,d}-\Delta & (i,j)\in\Gamma_1 \quad ise \\ x_{ij,d}+\Delta & (i,j)\in\Gamma_2 \quad ise \\ x_{ij,d} & d.d. \end{cases}$$

olarak tanımla.

8.Adım: Eğer $t_{rs}\geq DZ^k$ ise $c_{rs}^{k+1}=M$ olarak yeniden tanımla. $TM^{k+1}=TM^{k+1}+\delta\Delta$ olsun. $B_{d+1}=B_d+\{(e,f)\}-\{(r,s)\}$ olarak al. Eğer (p,q)=(r,s) ise $B=B_{d+1}$ olsun ve 3. adıma git. Diğer durumlar için d=d+1 olsun ve 4. adıma git.

9.Adım: $X_{d+1}=X_d$, $c_{rs}^{k+1}=M$ olarak tanımla $B_{d+1}=B_d+\{(e,f)\}-\{(r,s)\}$, d=d+1, 4. adıma git.

10.Adım: Uygun çözüm ikilileri kümesi $\{(TM^1,DZ^1),(TM^2,DZ^2),\ldots,(TM^k,DZ^k)\}$ dır. j>i iken $TM^i=TM^j$ koşulunu sağlayan çözüm ikilileri varsa (TM^i,DZ^i) ikilisini kümeden çıkar. Geriye kalan elemanlardan oluşan küme üstün çözüm ikililerinin bulunduğu kümedir.

$\{(TM^1,DZ^1),(TM^2,DZ^2),\ldots,(TM^m,DZ^m)\}$.

3.3.3. TM-DZ ve TM-DT için Srinivasan-Thompson algoritması

Bu algoritma herbir DZ değeri için (TM,DZ) ve (TM,DT) ikililerini bulmayı amaçlamaktadır. (TM,DZ) ikilisini bulmak için geliştirilen 1. Srinivasan-Thompson algoritmasının bir çeşitidir.

1.Adım: k=1. KUP'u çöz. Bulunan toplam maliyet değerine TM^1 adını vererek elde edilen çözüme zaman matrisinde karşı gelen darboğaz zaman değerini belirle ve DZ^1 ismini ver.

2.Adım: a-) $\Pi=\{(i,j)|t_{ij}\leq DZ_k\}$ ve $\Omega=\{(i,j)|t_{ij}=DZ^k\}$ kümelerini tanımla.

b-) $DT-\sum_{(i,j)\in\Omega} x_{ij}$

miktarını, Π çözümleri, arasında, TM^1 değeri korunmak koşuluyla sıfırlamaya çalış.

c-) DT miktarı sıfırlanamıyorsa 4. adıma git. Diğer durumlar için 3. adıma git.

3.Adım: k=k+1 , TM^k eniyi maliyettir. DZ^{k+1}yı hesapla 2. adıma dön.

4.Adım: Uygun çözüm ikilileri kümesi $\{(TM^1,DZ^1),(TM^2,DZ^2),\ldots,(TM^k,DZ^k)\}$ dır. j>i iken $TM^i=TM^j$ koşulunu sağlayan çözüm ikilileri varsa (TM^i,DZ^i) ikilisini kümeden çıkar. Geriye kalan elemanlardan oluşan küme üstün çözüm ikililerinin bulunduğu kümedir.

5.Adım: DT miktarı enküçüklenirken, her DZ^j (j=1,2,...,l) değerine karşı gelen (TM,DT) ikilisini hesapla. Elde edilen (TM,DT) ikilileri arasında üstün olanları ayır. (Eğer $TM'=TM$ ve $DT'<DT$ ise (TM',DT') üstündür.)

3.3.4. TM-DZ için Isermann algoritması

Isermann, 2.4.5'de açıklanan darboğaz ulaştırma problemindeki tanımlarından yararlanarak maliyet ve zaman enküçüklemeli ulaştırma problemi tanımlamaktadır. Isermann'a göre ilk olarak doğrusal olmayan, darboğaz zamanı enküçüklemeleme amacı, doğrusallaştırılmalıdır. Isermann bu problemi yine standart ulaştırma problemi olarak tanımlamakta (Lexicographic Transportation Problem-LTP) ve probleme ait amaç fonksiyonları şöyle tanımlamaktadır:

$$enkz_1-\sum_{(i,j)\in Y} c_{ij}.x_{ij}$$

$$Lexminz_2 = \sum_{r=1}^{s} e_r \cdot \sum_{(i,j)\in\gamma_r} x_{ij}$$

Isermann ayrıca d_{ij} olarak da bir sütun vektörü tanımlamaktadır.

$$d_{ij} = \begin{pmatrix} c_{ij} \\ e_r \end{pmatrix} \qquad (i,j)\in\gamma_r \ , \ r=1,2,\ldots,s$$

Isermann'ın algoritmasına ait akış şeması şekil 3.3'de verilmiştir.

Algoritma:

1.Adım: LTP metodu için bir başlangıç temel uygun çözüm bul.

2.Adım: $enkDZ = enbt_{ij}$ $\{(i,j)|x_{ij}>0\}$ eşitliği uyarınca DZ değerini bul ve T matrisinden DZ'ye en yakın ve küçük olan $\bar{t}$ değerini alt sınır olarak belirle. $(\bar{t}<DZ)$

$\gamma_1 = \{(i,j)|t_{ij}>DZ\}$

$\gamma_2 = \{(i,j)|t_{ij}=DZ\}$

$\gamma_3 = \{(i,j)|t_{ij}=\bar{t}\}$

$\gamma_4 = \{(i,j)|t_{ij}<\bar{t}\}$

olarak alt kümelere ayır. γ_1 alt kümesi için birim vektör e_1, γ_2 alt kümesi için birim vektör $e_2, \ldots,$ γ_4 alt kümesi için birim vektör e_4'dür. Bu ilişkilerden yararlanarak

$$\forall(i,j)\in\gamma_r \quad için \quad d_{ij} = \begin{pmatrix} c_{ij} \\ e_r \end{pmatrix} \ r=1,2,3,4$$

vektörlerini türet.

3.Adım: LTP'yi önceden belirlenen bir yöntem ile çöz.

4.Adım: $\bar{X}^1 = \{\bar{x}^1_{ij}\}$ LTP için eniyi çözümdür $\bar{X}^1$ çözümü DUP için başlangıç çözümü oluşturur.

5.Adım: B kümesindeki (i,j) ikilileri için $d_{ij}=u_i+v_j$ ilişkisini kullanarak u_i , v_j değerlerini bul.

6.Adım: $\gamma\backslash B$ kümesindeki (i,j)'ler için

$z_{ij}=d_{ij}-\bar{u}_i-\bar{v}_j$ değerlerini hesapla.

7.Adım: $z^1_{ij}\geq 0 \Rightarrow$ geçerli çözüm ikinci amaç fonksiyonuna göre eniyi çözümdür, 10. adıma git. Diğer durumlarda (k,l) olarak tanımlanabilecek yeni değişken ikililerini aşağıdaki eşitliğe göre tanımla ve 9. adıma git. Eğer eşitlikte verilen seçim kuralı uygulanamıyorsa 8. adıma git.

$$-z^1_{kl}/z^{*r}_{kl}=enk\{-z^1_{ij}/z^{*r}_{ij} \mid z^*_{ij}\leq 0, Z^{*r}_{ij}<0, (i,j)\in\gamma\backslash B\}$$

8.Adım: $z_{kl}=lexmin\{z_{ij} \mid z_{ij}\leq 0, (i,j)\in\gamma\backslash B\}$

kuralı gereğince z^*_{ij} 'lar içinden seçim yap. Böylece belirlenen x_{kl} değişkeni temele girecek değişkendir ve temelde $\bar{x}_{kl}$ değerini alır. Eğer $\bar{x}_{kl}=0$ ise 9. adıma git.

$\bar{x}_{kl}>0$ ise geçerli temel uygun çözüm eniyi darboğaz zamanı ve eniyi darboğaz taşımayı sağlar. Toplam maliyet değerinin artışı, akış vektöründeki diğer bileşenlerin azalması nedeniyle dikkate alınmaz , 10. adıma git.

9.Adım: x_{kl} değeri temele girecek yeni değişkendir. Yeni çözümü bul. 5. adıma git. 7. adımda (k,l) ikilisi olarak birden fazla sayıda ikili tanımlanmışsa, herbir ikili için bütün komşu temel uygun çözümleri belirle ve hepsine 5. adımı uygula.

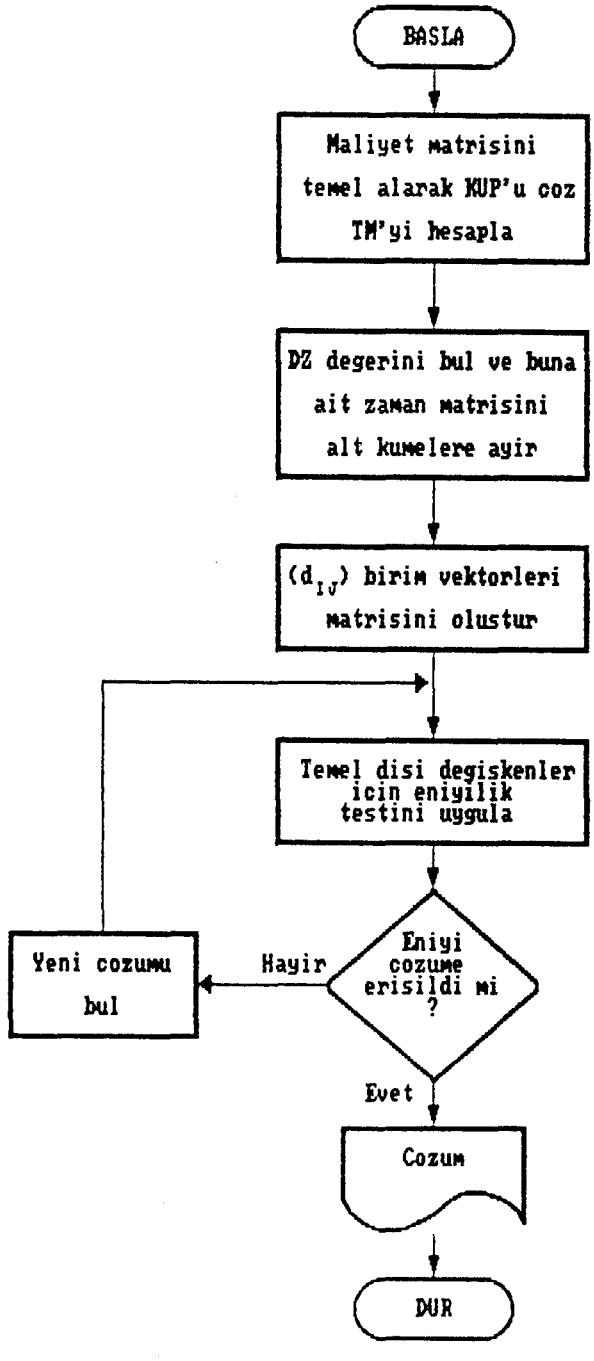
10.Adım: DUP için bütün etkin temel uygun çözümler bulunmuştur. Burada iki durum söz konusu olabilir:

i-) Her temel uygun çözümün DUP için sadece bir tane komşu temel uygun çözümü bulunmuştur. Yani p=1,2,...,q-1 olmak üzere x^p, x^{p+1} komşu çözümlerdir. O halde

$$\bar{S}=\bigcup_{p=1}^{q-1}\{\lambda.x^p+(1-\lambda).x^{p+1} \mid 0\leq\lambda\leq 1\}$$

kümesi DUP'un bütün etkin çözümlerini içerir.

ii-) Bir temel uygun çözümün 7. ve 8. adımlardaki seçim kuralları gereğince 2 ya da daha fazla sayıda komşu temel uygun çözümü vardır. Komşu temel uygun çözümlerin bütün dışbükey bileşimleri, bütün etkin çözümlerin bulunduğu kümenin uygun birer alt kümesi olabilir. Bu durumda izlenecek en basit yol, DUP için bulunmuş etkin çözümler kümesinin tamamını (tanımlayarak) hesaplamaktır.



Sekil 3.3: TM-DZ Isermann'ın Algoritmasına Ait Akış Seması.

4. TOPLAM MALİYET-DARBOĞAZ ZAMAN TABANLI İKİ AMAÇLI ULAŞTIRMA PROBLEMİ İÇİN BİR ALGORİTMA

Toplam maliyet ve Darboğaz zaman, TM-DZ tabanlı iki amaçlı ulaştırma problemlerinde, amaçların birbirleriyle çelişmesi nedeniyle, her iki amacı da eniyileyen tek bir çözüm bulunamamakta, buna karşılık sıralı çözümler kavramından yararlanılarak, ardışık çözümler üretilmekte ve elde edilen seçenekler karar vericiye sunulmaktadır. İki amaçlı ulaştırma problemleri ile ilgilenen araştırmacılar, genellikle sıralı çözümlerin nasıl üretileceği konusunda çalışmalar yapmışlar ve çeşitli yaklaşımlar önermişlerdir. Geetha-Vartak tarafından yayınlanan makalelerde ise (Geetha and Vartak, 1989), üç boyutlu atama problemleri incelenmiş ve bu problem için üretilen sıralı çözümler arasından bir tanesinin seçilerek, karar vericiye önerilebilecek çözüm olarak belirlenmesine ilişkin, bir yöntem geliştirilmiştir.

Çalışmanın bu bölümünde, TM-DZ tabanlı iki amaçlı bir ulaştırma problemi için sıralı çözümleri bularak, bunlar arasından üstün olanlarını seçen bir algoritma geliştirilmektedir. Ayrıca Geetha-Vartak tarafından üç boyutlu atama problemleri için önerilen ve bu çalışmada iki amaçlı ulaştırma problemlerine uyarlanan üstün çözümler arasından bir tanesini seçme yöntemi de algoritmaya eklenmekte ve ilk kez iki amaçlı bir ulaştırma problemi için karar vericiye önerilecek bir çözümün belirlenmesi olanağı sağlanmaktadır. Geliştirilen algoritmanın Turbo Pascal programlama dili ile yazılan programın listesi de sunularak, rassal olarak türetilen örnek problemler, geliştirilen algoritma ile çözüme kavuşturulmakta ve elde edilen sonuçlar tartışılmaktadır.

Algoritmada sıralı çözümler, KUP çözüm tekniği olan MODI metodu ve DUP çözüm tekniği olan Primal algoritma birarada kullanılarak elde edilmektedir. Daha sonra elde edilen sıralı çözümler arasından üstün çözüm ikilileri seçilerek, üstün (TM,DZ) çözüm ikililerinden oluşan bir

küme elde edilmekte ve iki amaçlı ulaştırma problemlerine uyarlanan seçim yöntemi uygulanarak karar vericiye bir çözüm ikilisi önerilmektedir.

4.1. Sıralı Çözümler Arasındaki İlişkiler

TM-DZ tabanlı iki amaçlı bir ulaştırma probleminde, birinci amaç fonksiyonu toplam maliyeti, ikinci amaç fonksiyonu ise darboğaz zamanı enküçüklemektir. Problem sadece birinci amaç fonksiyonu temel alınarak çözülürse, bilinen KUP incelenmiş olur ve modele ait maliyet matrisine göre eniyi toplam taşıma maliyetini sağlayan çözüm bulunur. Diğer yandan problem sadece ikinci amaç fonksiyonu temel alınarak çözülürse, DUP problemi incelenmiş olur ve bu kez de modele ait zaman matrisine göre eniyi darboğaz zamanı sağlayan çözüm elde edilir. Büyük bir olasılıkla, eniyi toplam maliyetin bulunduğu çözüme ait darboğaz zaman değeri, problem için eniyi DZ değeri olmayacak; benzer şekilde eniyi darboğaz zaman değerinin elde edildiği çözüme ait toplam maliyet değeri de, eniyi TM değeri olmayacaktır. Kısaca KUP ve DUP'un ayrı ayrı çözülmesinden sonra elde edilen (TM,DZ) ikilileri, farklı ikililer olacaktır. Bunların yanısıra probleme ait hem toplam maliyet değeri eniyi olmayan hem de darboğaz zaman değeri eniyi olmayan ikililerin de olması olasılığı vardır. Bu tür ikililer ara çözümlere ait ikililerdir. Sözü edilen bütün bu ikililerin elde edilmesi için farklı yöntemler kullanılabilir. Ancak sonuçta aralarında herhangi bir üstünlük ilişkisi tanımlanmadan elde edilen bu çözüm ikililerine, sıralı çözüm ikilileri denir.

4.1.1. Üstün çözüm ikilisi kavramı

Modelin birinci amaç fonksiyonu temel alınarak bulunan çözümüne ait maliyet-zaman ikilisi (TM^1, DZ^1) olsun. Bu çözüm ile elde edilen TM değeri aynı zamanda model için elde edilebilecek enküçük maliyet değeri olduğu için $TM^1 = TM^*$ olarak gösterilebilir. Benzer şekilde modelin

ikinci amaç fonksiyonu temel alınarak bulunan çözümüne ait maliyet-zaman ikilisi ise (TM^{*},DZ^{*}) olsun. Burada da DZ^{*} olarak bulunan değer, problem için elde edilebilecek eniyi darboğaz zaman değeridir ve $DZ^{*}=DZ^{**}$ olarak ifade edilebilir. Kolaylıkla anlaşılabileceği gibi, iki farklı amaç fonksiyonunun ayrı ayrı eniyilenmesiyle elde edilen bu iki maliyet-zaman ikilisi, modelin çözümünde elde edilecek sıralı çözümler için uç noktaları oluşturmaktadır. Sıralı çözümler, her iki amaç fonksiyonuna göre eniyi TM veya DZ değerini sağlayamayan ve bulunan uç noktalar arasında yeralan çözümlerdir. Sıralı (TM,DZ) çözüm ikililerinden bazıları, diğerleriyle ikili olarak karşılaştırıldığında daha iyi TM veya DZ değerine sahip olabilir. Bu özelliğe sahip çözüm ikililerine, üstün çözüm ikilileri denir. Doğal olarak karar vericiye sunulacak çözüm kümesi sıralı çözüm ikililerinin bulunduğu küme değil, üstün çözüm ikililerinin bulunduğu kümedir. Çünkü üstün çözüm ikilileri, diğerlerinden daha iyi oldukları için seçilmişlerdir. Ayrıca önerilecek çözümün seçimi için de üstün çözüm ikililerinin bulunması gerekir. Bu nedenle sıralı çözüm ikililerinden oluşan çözüm kümesinden üstün çözüm ikililerini seçmek için bir yöntemin kullanılması gerekmektedir.

4.1.2. Sıralı çözümlerde üstünlük ilişkileri

İki amaçlı ulaştırma problemi için sıralı çözümler ve bunlara ait maliyet-zaman ikilileri bulunmuş olsun. Elde edilen çözüm kümesinden üstün çözüm ikililerini seçebilmek için ikililerin karşılıklı olarak incelenmesi gerekir. Teorik olarak bulunan ikililer arasında karşılaşılabilecek 9 farklı durum vardır: j>i iken bu farklı durumlar şöyle açıklanabilir:

1. $TM^{i}<TM^{j}$ iken $DZ^{i}<DZ^{j}$ ise (TM^{i},DZ^{i}) ikilisi üstündür.
2. $TM^{i}<TM^{j}$ iken $DZ^{i}>DZ^{j}$ ise üstünlük yoktur.
3. $TM^{i}<TM^{j}$ iken $DZ^{i}=DZ^{j}$ ise (TM^{i},DZ^{i}) ikilisi üstündür.
4. $TM^{i}>TM^{j}$ iken $DZ^{i}<DZ^{j}$ ise üstünlük yoktur.

5. $TM^{k} > TM^{j}$ iken $DZ^{k} > DZ^{j}$ ise (TM^{j}, DZ^{j}) ikilisi üstündür.
6. $TM^{k} > TM^{j}$ iken $DZ^{k} = DZ^{j}$ ise (TM^{j}, DZ^{j}) ikilisi üstündür.
7. $TM^{k} = TM^{j}$ iken $DZ^{k} < DZ^{j}$ ise (TM^{k}, DZ^{k}) ikilisi üstündür.
8. $TM^{k} = TM^{j}$ iken $DZ^{k} > DZ^{j}$ ise (TM^{j}, DZ^{j}) ikilisi üstündür.
9. $TM^{k} = TM^{j}$ iken $DZ^{k} = DZ^{j}$ ise çözümler aynıdır.

Kolaylıkla görülebileceği gibi sıralı çözümlerin arasından, üstün çözüm ikililerinin seçilebilmesi için yukarda tanımlanan 9 farklı durumun tek tek incelenmesi yeterli olacaktır.

4.2. Üstün Sıralı Çözüm İkililerinin Bulunması

İki amaçlı bir ulaştırma problemi için bütün sıralı çözüm ikilileri bulunmuş olsun. Bu ikililer arasından seçilecek üstün çözüm ikilileri arasında

$$TM^{q} < TM^{q-1} \quad ve \quad DZ^{q} > DZ^{q-1} \quad (q = 1,2,\ldots,\varepsilon)$$

ilişkisi vardır. Yani üstün çözüm ikililerinden birincisinde toplam maliyetin eniyi değeri aldığı, sonuncusunda ise darboğaz zamanın en iyi değerini aldığı çözüm ikilileri bulunmuştur. Bulunan diğer çözümler ise hem eniyi TM, hem de eniyi DZ değerine sahip olmayan ara çözümlere ait ikililerdir. Burada üstün çözüm ikililerini belirlemek için 4.1.2'de açıklanan 9 farklı durum tek tek incelenmesine gerek yoktur.

4.2.1. Sıralı çözüm ikililerinin bulunması

Sıralı çözüm ikililerini bulmak için ilk adımda, maliyet matrisinden yararlanarak KUP çözülür. Elde edilen çözüme, zaman matrisinden karşı gelen darboğaz zaman değerini bulunarak, maliyet matrisi

$$c_{ij} = \begin{cases} c_{ij} & eğer \quad t_{ij} \leq DZ \\ M & eğer \quad t_{ij} \geq DZ \end{cases} \quad (M > 0 \ ve \ çok \ büyük)$$

ilişkisi ile indirgenir ve indirgenmiş matrise göre KUP yeniden çözülür. Böylece yeni bir çözüm ve bu çözüme karşı gelen (TM,DZ) değerleri belirlenmiş olur.

Ardıştırmalar, KUP için indirgenmiş matrislere göre, amaç fonksiyonunun değeri M'den az olan bir başka çözüm bulunamayıncaya kadar devam eder. Bu şekilde elde edilen çözümler, sıralı (TM,DZ) ikililerinden oluşan ilk kümeyi verir.

Tanımlanan maliyet matrisini indirgeme yöntemi incelendiğinde algoritmanın her ardıştırmasından sonra elde edilecek darboğaz zaman değerinin azalacağı kolaylıkla görülebilir. Yani elde edilen darboğaz zaman değerleri arasında, $DZ^1>DZ^2>...>DZ^k$ ilişkisi olacaktır.

Yukarda açıklanan yaklaşımla elde edilen toplam maliyet değerinin, her ardıştırma sonrasında artması veya en azından bir önceki ardıştırmada elde edilen değer ile aynı kalması beklenir. Çünkü maliyet matrisi üzerinde hiç bir indirgeme işlemi yapılmadan elde edilen toplam maliyet değeri (TM^1), model için elde edilebilecek enküçük değerdir ve ilerleyen ardıştırmalar boyunca bir daha bu değerden daha küçük bir toplam maliyet değerine erişilemeyeceği açıktır. Kısaca 1. ardıştırma ve sonrasında elde edilecek TM değerleri, TM^1'e eşit veya ondan daha büyük olmak zorundadır. Bu nedenlerle toplam maliyet değerleri arasında da $TM^1 \leq TM^2 \leq ... \leq TM^k$ ilişkisinin olması beklenir. TM ve DZ değerleri arasında olması beklenen bu ilişkiler, üstün çözüm ikililerinin elde edilmesinde kolaylık sağlar.

4.2.2. Üstün sıralı ikililerin seçimi

4.2.1'de tanımlanan maliyet matrisini indirgeme yönteminin kullanılmasıyla elde edilecek sıralı çözüm ikilileri arasında, j>i için, hiçbir şekilde $DZ^i<DZ^j$, $DZ^i=DZ^j$ ve $TM^j<TM^i$ ilişkileri gerçekleşmeyecektir. Bu nedenle de 4.1.2'de açıklanan karşılaşılabilir durumlardan; birinci eşitsizlik nedeniyle 1., 4. ve 7. durumların; ikinci eşitlik nedeniyle 3.,6. ve 9. durumların; ve j>i koşulu nedeniyle de 5. durumun varolması söz konusu değildir. Bu nedenle geriye kalan iki durumun, elde edilen sıralı çözüm ikilileri için incelenmesiyle üstün çözüm

ikililerini seçmek mümkün olacaktır. Üstün çözüm ikililerini bulmak için incelenmesi gereken diğer iki durum şunlardır:

i- $TM^{k}<TM^{j}$ iken $DZ^{k}>DZ^{j}$: Bu zaten beklenen durumdur ve ikililer arasında üstünlük yoktur, ikisi de geçerliliğini korur.

ii- $TM^{k}=TM^{j}$ iken $DZ^{k}>DZ^{j}$: (TM^{j},DZ^{j}) ikilisi üstündür, diğeri kümeden çıkarılır.

Böylece elde edilen sıralı çözümler arasında sadece ikinci kritere göre üstün çözümleri aramak, yeterli olacaktır.

Burada model için bütün üstün çözüm ikililerinin bulunup bulunmadığı çok önemlidir. Çünkü eğer modele ait bazı üstün çözüm ikilileri bulunamamışsa; uygulanacak üstün çözüm ikilileri arasından bir tanesini seçme yöntemiyle, yanlış çözümlerin önerilmesi olasılığı ortaya çıkar. Bu nedenle uygulanacak yöntem sonucunda ilk olarak bütün üstün çözüm ikililerinin bulunacağı gösterilmelidir.

Modelin hiçbir indirgeme işlemine tabi tutulmadan bulunan ilk çözümüne ait ikilileri (TM^{1},DZ^{1}), maliyet matrisinin DZ^{1} değerine göre indirgenmesinden sonra bulunan çözüm ikilileri de (TM^{2},DZ^{2}) olsun. TM^{2} değeri maliyet matrisi indirgendikten sonra bulunan eniyi toplam taşıma maliyetidir. Diyelimki DZ^{1} ile DZ^{2} değerleri arasında $\overline{DZ}$ gibi bir başka darboğaz zaman değerine sahip olan ve $DZ^{1}>\overline{DZ}>DZ^{2}$ ilişkisinin gerçeklendiği bir çözüm bulunsun. Bu çözüme ait olarak hesaplanacak $\overline{TM}$ değeri hiçbir zaman TM^{2} değerinden daha küçük bir değerde olamaz. Çünkü DZ^{1} değerine göre indirgenmiş matrisin toplam maliyet açısından eniyi çözümü daha önce de açıklandığı gibi TM^{2} değeridir. Bulunan $\overline{TM}$ değeri ise TM^{2} değerinden daha büyük veya ona eşit olmak zorundadır. Eğer $\overline{TM}$ değeri TM^{2} değerinden daha büyük ise $\overline{TM}>TM^{2}$ ve $\overline{DZ}>DZ^{2}$ durumu sözkonusu demektir. Bu

durumda (TM^{q},DZ^{q}) ikilisi üstündür. Eğer $\overline{TM}$ değeri TM^{q} değerine eşit ise $\overline{TM}=TM^{2}$ ve $\overline{DZ}>DZ^{2}$ durumu sözkonusu demektir. Bu durumda da yine (TM^{q},DZ^{q}) ikilisi üstündür.

Görülebileceği gibi sıralı çözümlerin bulunmasında izlenen yöntem ile, bütün üstün çözüm ikililerine ulaşılmaktadır. Ayrıca yöntem, üstün çözüm ikililerinin elde edilmesi için incelenecek durumlarla da çelişmemektedir. Bu nedenle üstün çözüm ikililerinin arasından karar vericiye önerilmek üzere bir tanesini seçme yönteminin uygulanması halinde hatasız sonuç alınacaktır.

4.3. Önerilecek Üstün Sıralı Çözümün Bulunması

Daha önce de değinildiği gibi, S.Geetha ve M.N.Vartak 1989 yılında yayınladıkları iki ayrı makale ile üç boyutlu Atama problemlerinde, sıralı çözümler içinden bir tanesini karar vericiye sunulmak üzere belirleyecek bir yöntem geliştirmişlerdir (Geetha and Vartak, 1989). Bu yöntemde ana fikir, elde edilen sıralı çözümler arasından üstün olanlarını diğerlerinden ayırarak yeni bir çözüm kümesi oluşturmak ve birim zaman gecikmenin toplam maliyete yansıması enküçük olan çözümü, önerilecek çözüm olarak belirlemektir. Buna göre darboğaz zamandaki her bir birimlik artışa karşılık toplam maliyetindeki artışı en az olan çözüm ikilisi, tercih edilmesi gereken çözüme ait ikilidir. Kısaca, Geetha-Vartak üstün sıralı çözümleri seçme yöntemi;

TM^{i}: i. çözüme karşı gelen toplam maliyet

DZ^{i}: i. çözüme karşı gelen darboğaz zaman

olmak üzere aşağıdaki gibi açıklanabilir:

i- Belirlenen s tane üstün çözüm ikilisi arasında

$$R_p=\frac{TM^{q-1}-TM^{q}}{DZ^{q}-DZ^{q-1}}$$

eşitliği gereğince s-1 tane analiz oranı bul.

ii- Analiz oranları (R_p) arasından enküçük olanını seç. Önerilecek çözüm (TM^{p+1},DZ^{p+1}) ikilisine ait çözümdür.

Geetha-Vartak tarafından üç boyutlu atama problemleri için önerilen bu yaklaşım, kolaylıkla iki amaçlı ulaştırma problemleri için de uygulanabilir. Çünkü, üç boyutlu atama problemlerinde de üstün çözüm ikilileri daha önceden herhangi bir yöntemin uygulanmasıyla bulunmuş sıralı (TM,DZ) ikilileri elde edilmekte ve seçim yöntemi bu ikililer üzerinde uygulanmaktadır. Bu yapısal benzerlik, Geetha-Vartak yönteminin TM-DZ tabanlı iki amaçlı ulaştırma problemi için de uygulanmasını mümkün kılmaktadır. Bu nedenle; bu çalışmada geliştirilen algoritmada, önerilecek çözümü belirlemek için yukarda açıklanan adımlar uygulanacaktır.

4.4. Geliştirilen Algoritma

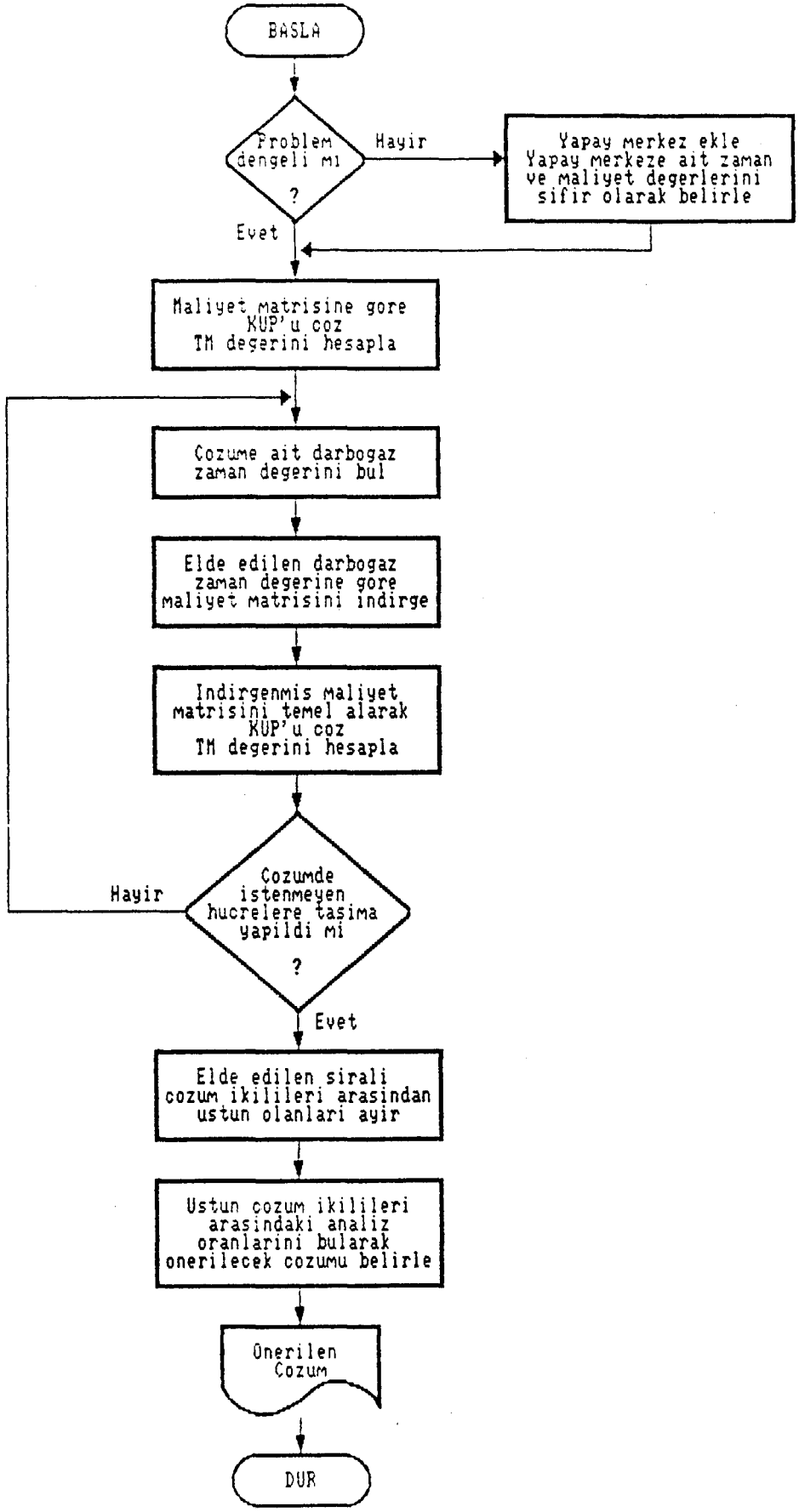
İki amaçlı ulaştırma problemleri ile çalışmalar yapan Srinivasan-Thompson, TM-DZ tabanlı iki amaçlı ulaştırma problemi için üç farklı algoritma geliştirmiştir. Bu algoritmalarda, probleme ait zaman ve maliyet matrisleri ilişkilendirilerek problem Ford-Fulkerson algoritması ile çözülmektedir. Sıralı çözümlerin bulunması ise maliyet matrisini önceki çözümlerden elde edilen darboğaz zaman değerine göre indirgeyip problemi bu indigenmiş matrisle yeniden çözerek sağlanmıştır.

Bu çalışmada geliştirilen algoritmada ise zaman ve maliyet matrisleri arasındaki ilişkilendirme Srinivasan-Thompson tarafından tanımlanan şekilde gerçekleştirilmekte, fakat problemin çözümü, KUP'u çözmek için yaygın olarak kullanılan, MODI metodu ile sağlanmaktadır. MODI metodunun seçilmesindeki temel nedenler, metodun kolay anlaşılır olması, modelin çözümünü bulabilmek için serim modeli biçiminde ifade edilmesine gerek duyulmayışıdır. Maliyet matrisinin indirgenmesinde ayrıca Primal algoritmadan da yararlanılmaktadır. Primal algoritmanın seçilmesindeki neden de, Primal algoritma

yaklaşımının sıralı çözüm ikililerinin elde edilmesinde çok etkin olması ve kolay anlaşılıp, uygulanabilir bir algoritma olmasıdır.

Geliştirilen algoritma, ilk olarak 4.2.1'de açıklanan ilişkilendirme gereği sıralı çözüm ikililerini bulmakta daha sonra 4.2.2'de aktarılan üstün çözüm ikililerini seçme yöntemine göre üstün çözümlerin bulunduğu kümeyi belirlemektedir. Son olarak 4.3'de açıklanan önerilecek çözümü bulma yöntemi uygulanmakta ve karar vericiye sunulacak bir çözümün elde edilmesini sağlamaktadır.

KUP'un çözümüyle ilgili karşılaşılabilecek tek sorun modelin dengelenmemiş olması halidir. KUP için dengelenmemiş model ile karşılaşıldığında modele yapay merkez ya da bölge eklenerek işlemlere devam edilir. Yapay noktaya ait maliyet değerleri ise sıfır olarak alınır. Benzer düşünce DUP için de geçerlidir ve çözüme başlamadan önce model dengelenerek yapay noktaya ait zaman değerleri sıfır olarak atanır. Algoritmada da dengelenmemiş bir model ile karşılaşıldığında, gereksinim doğrultusunda bir yapay nokta eklenmekte ve bu noktaya ait maliyet ve zaman değerleri sıfır olarak atanmaktadır. Algoritmaya ait akış şeması şekil 4.1'de verilmiştir.



Sekil 4.1: Belirlenen algoritmanın akış şeması.

4.4.1. Algoritmanın adımları

Algoritmaya göre, üretim merkezlerinin kapasiteleri, dağıtım bölgelerinin talepleri, maliyet ve zaman matrisleri bilinen iki amaçlı bir ulaştırma problemi için yapılacak işlemler, adımlar halinde şöyle açıklanabilir:

1.Adım: Toplam kapasite ve toplam talepler arasında denklik olup olmadığını incele. Denklik varsa 2. adım git. Yoksa gereksinime göre yapay üretim merkezi veya yapay dağıtım bölgesi oluştur. Yapay noktaya ait maliyet ve zaman matrisindeki değerleri sıfır olarak al ve 2. adıma git.

2.Adım: Modeldeki maliyet matrisini temel alarak KUP'u çöz. Bulunan toplam maliyet değerine TM^1 ve zaman matrisinden çözüme karşı gelen darboğaz zaman değerine DZ^1 ismini ver. r=1 yap.

3.Adım: $c_{ij}^{r+1} = \begin{cases} c_{ij}^{r} & eger \ \ t_{ij} < DZ \\ M & eger \ \ t_{ij} \geq DZ \end{cases}$ $(M>0 \ ve \ cok \ buyuk)$

dönüşümünü gerçekleştir.

4.Adım: c_{ij}^{r+1} matrisinden yararlanarak KUP'u çöz ve toplam maliyet değerini bul. TM^{r+1} ile göster.

5. Adım: a-) Eğer $TM^{r+1} < M$ ise bulunan çözüme zaman matrisinden karşı gelen DZ^{r+1} değerini bul. r=r+1 ve 3. adıma dön.

b-) Eğer $TM^{r+1} \geq M$ ise işlemler bitmiştir, 6. adıma git.

6.Adım: Elde edilen r tane sıralı çözümü enküçük maliyet değerli ikiliden başlayarak artan sırada listele. İkililer (TM^q, DZ^q) q=1,2,...,r biçimindedir. $DZ^q > DZ^{q+1}$ iken $TM^{q+1} = TM^q$ ise $(TM^{q+1} > DZ^{q+1})$ ikilisi üstündür. $(TM^q > DZ^q)$ ikilisi kümeden çıkar. Bu işlemleri bütün ikililer için yaparak s tane üstün çözüm ikilisinin bulunduğu U kümesini oluştur.

7.Adım: U={(TM^1,DZ^1),(TM^2,DZ^2),...,(TM^s,DZ^s)} dir.

İkililer arasında $TM^q<TM^{q+1}$ ve $DZ^q>DZ^{q+1}$ (q=1,2,...,s) ilişkisi vardır.

$$R_D=\frac{TM^{q+1}-TM^q}{DZ^q-DZ^{q+1}}$$

eşitliği ile s-1 tane analiz oranı belirle.

$$R_{p^*}=\min_{1\le p\le s-1}\{R_p\}$$

eşitliği gereğince bir analiz oranını seç. Buna karşı gelen (TM^{p+1},DZ^{p+1}) ikilisine ait çözüm karar vericiye sunulacak eniyi çözümdür.

4.4.2. Algoritmanın İrdelenmesi

Geliştirilen algoritma incelendiğinde, iki önemli açıdan algoritmanın etkin olduğu anlaşılacaktır. Algoritmada, TM-DZ tabanlı iki amaçlı bir ulaştırma problemi için ilk olarak sıralı çözüm ikilileri bulunmakta daha sonra bu ikililer arasından üstün olanlar seçilerek, karar vericiye sunulacak bir tek çözüm belirlenmektedir. Sıralı çözüm ikililerinin elde edilmesinde Primal algoritmadaki, maliyet matrisini herhangi bir çözümdeki darboğaz zaman değerine göre indirgeme yöntemi benimsenmiş, indirgenen her maliyet matrisine MODI metodu uygulanarak, sıralı çözüm ikililerinin elde edilmesi sağlanmıştır. Primal algoritmanın diğer DUP çözüm yöntemlerine göre önemli bir üstünlüğü olan sıralı çözüm ikililerinin bulunmasının sağlaması ve indirgeme işlemlerinde kolay anlaşılır olması, geliştirilen algoritmaya da yansımıştır. Bu nedenle algoritmadaki sıralı ikilileri bulma yönteminin anlaşılır ve etkin olduğu söylenebilir. Öte yandan aynı yöntem üstün çözüm ikililerinin de bulunmasında kolaylık sağlamaktadır. Yöntem sonucunda elde edilen sıralı çözüm ikilileri arasında olması gereken karşılıklı ilişkiler, 4.1.2'de açıklanan sıralı ikililer arasındaki 9 farklı durumun tek tek incelenmesi gereğini ortadan kaldırmakta ve sadece bir durumun incelenmesi sonucunda, kolaylıkla üstün çözüm ikililerinin bulunmasını sağlamaktadır. Bu da

algoritmanın etkin olmasındaki ikinci önemli nedendir. Ayrıca ilk kez iki amaçlı bir ulaştırma problemi için karar vericiye önerilecek bir çözümün belirlenmesi olanağı da yine bu algoritma ile sağlanmaktadır.

Algoritmanın diğer üstünlüğü ise uygulama kolaylığıdır. Günümüzde KUP çözüm tekniği olan MODI yöntemine ait pekçok yazılım bulunmaktadır. Bu yazılımlardan herhangi birinin kullanılmasıyla algoritma büyük boyutlu problemler için de uygulanabilecektir. Diğer yandan algoritmada tanımlanan indirgeme işleminin ve diğer adımların basitliği, kolaylıkla bir paket programın oluşturulmasına olanak sağlamaktadır. Nitekim; algoritmaya ait bilgisayar programının çıktısı da, Ek-III'de sunulmaktadır. Program, kullanacak kişinin bilgisayar bilmesine gerek olmayacak şekilde bir paket program biçiminde düzenlenmiş ve işlemler, menüler aracılığıyla sağlanmıştır.

Geliştirilen algoritmanın önemli bir özelliği de bu alanda geliştirilmiş diğer algoritmalarla karşılaştırma olanağının bulunmasıdır. Özellikle algoritmanın hızlılığı, etkinliği ve kolay anlaşılabilir olması karşılaştırmalar yapmak için uygundur ve algoritmanın üstünlüğünü ortaya koyacak özelliklerdir. Ayrıca indirgenen maliyet matrisine sadece MODI metodunun uygulanması da gerekmez. İstenen herhangi bir KUP çözüm tekniği, algoritmadaki MODI metodu ile değiştirilerek yeni yaklaşımlara uyarlanması mümkündür. Hatta problemin serim modeli halinde ifade edilerek Ford-Fulkerson algoritması gibi bir yöntemle de çözülmesi mümkündür. Bu nedenle algoritma, yapılacak değişikliklere karşı oldukça esnektir ve bu tür yaklaşımlarla yeni algoritmaların geliştirilmesine uygundur. Bu da algoritmanın diğer bir kayda değer özelliğini sergilemektedir.

4.5. Geliştirilen Programın Yapısı

Geliştirilen algoritma, Turbo Pascal programlama dili kullanılarak programlanmış ve TM-DZ tabanlı iki amaçlı ulaştırma problemlerinde sıralı çözüm ikililerini ve üstün çözüm ikililerini bulan, ayrıca önerilmek üzere bir tane çözümü seçen etkin bir paket program biçiminde düzenlenmiştir. Program çalıştırıldıktan sonra bir ana menü gelmekte ve kullanıcının isteğine göre dallanma yapılarak modelin girişi ve çözümü sağlanmaktadır. Ana menü 10 bölümden oluşmaktadır:

1. Yeni Problem Girisi
2. Problem Kaydetme
3. Disketten Problem Okuma
4. Rassal Veri Türetme
5. Problemin Çözümü
6. Sonuçlarin Gösterilmesi
7. Verilerin Tabloda Gösterimi
8. Sonucun Tabloda Gösterimi
9. Sonuclarin Yazdırılması
10. Programdan Çıkıs

Bu seçenekler, ana menüde belirtilen yön tuşlarının kullanılmasıyla işleme girmekte kullanıcı, ekranda görüntülenen sorulara yanıt vererek etkileşimli çalışma sağlanmaktadır. Programa ait menüler EK-1'de, program listesi ise EK-2'de verilmiştir. Rassal olarak türetilen bir probleme ait çözümler de EK-3'de sunulmuştur.

4.5.1. Programda veri girişi

Herhangi bir iki amaçlı ulaştırma probleminin girilebilmesi için kullanıcıya iki seçenek sunulmuştur:

i- Kullanıcının kendi problemine ait verileri varsa:

Bu durumda 1 nolu seçenek tercih edilmelidir. Ekranda önce üretim merkezi ve dağıtım bölgelerinin sayıları, sonra da kapasite, talep, taşıma maliyetleri ve taşıma zamanları değerlerinin ne olacağı tek tek sorularak, kullanıcıdan verilerini girmesi istenir. Verilerin hatasız olarak

girilmesiyle ekranda ana menü görüntülenir. Bu aşamada kullanıcı isterse 7. seçeneği seçerek veri girişinde hata yapıp yapmadığını kontrol edebilir. Eğer hata varsa, verilerde değişiklik yapılmasının istenip istenmediğine dair soru 'E' ile yanıtlandırılarak, istenen herhangi bir veri değiştirilebilir. Değişiklik yapılmayacaksa klavyeden 'H' girilerek tekrar ana menüye dönüş sağlanır.

ii- Kullanıcının bir probleme ait verileri yoksa:

Bu durumda gerek programın çalışmasını denetlemek ve gerekse de çeşitli özelliklerdeki iki amaçlı ulaştırma problemlerini incelemek için 4 nolu Rassal veri türetme seçeneği seçilmelidir. Bu seçim ile ekranda, türetilecek problem için üretim merkezi ve dağıtım bölgesi sayılarının ne olması gerektiği, kapasite, talep, taşıma maliyeti ve taşıma zamanı gibi değerler için alt ve üst sınırların neler olması gerektiği sorulacaktır. Kullanıcının dilediği değerleri girmesi ile veriler üniform dağılıma uygun olarak türetilir ve ana menüye dönülür. Ana menüye dönüş, veri giriş işlemlerinin bittiğini işaret etmektedir. Bundan sonra istenirse türetilen veriler 7 nolu seçenek ile görülüp değiştirilebilir.

4.5.2. Problem çözümü

Başarılı bir veri girişinden sonra problem çözümü için 5 nolu seçenek tercih edilmelidir. Çözümler, 'ENTER' tuşuna her basışta peş peşe sıralı çözümler olarak ekranda görüntülenir. Her çözümde, ekranda görüntülenen çözümün kaçıncı sıralı çözüm olduğu, kaç ardıştırma yapılarak çözüme erişildiği, çözüme ait toplam maliyet değeri ve darboğaz zaman değeri ekrana görüntülenir. Çözümlerde, $m + n - 1$ tane karar değişkeninin aldığı değer, üretim merkezlerinden dağıtım bölgelerine yapılacak taşıma olarak görüntülenir. Herhangi bir ardıştırma sırasında dejenere çözüm ile karşılaşılsa bile, rassal olarak temelde olamayan bazı karar değişkenlerine sıfır değeri atanmakta ve çözümde, $m + n - 1$ tane karar değişkeninin bulunması sağlanmaktadır. Bütün sıralı çözümlerin tek tek

listelenmesinden sonra elde edilen sıralı çözüm ikilileri toplu olarak ekranda listelenir. Bu listeden yararlanılarak üstün çözüm ikilileri bulunur ve ekranda gösterilir. Sonraki aşamada karar vericiye önerilecek çözümün numarası ile (TM,DZ) değerleri ve çözüme ait karar değişkenlerinin değerleri görüntülenir.

5. seçeneğin kullanılarak bir problemin tekrar tekrar çözülmesi olasıdır. Ayrıca büyüklüğü en fazla 7x9 boyutuna kadar olan ulaştırma problemleri için eniyi çözüm değerleri 8. seçeneğin seçilmesi ile tablo formatında da görülebilmektedir. Probleme ait bütün sıralı çözümlerin görüntülenmeden, önerilecek çözüm elde edilmek isteniyorsa 6. nolu seçenek seçilmelidir.

4.5.3. Programda uygulama kolaylıkları

Uygulama boyunca karşılaşılacak sürekli veri girişi problemi, 2. ve 3. seçeneklerle önlenmiştir. 2.seçenek ile veri girişi tamamlanan bir problem, istenen kütük ismiyle, istenen diskete kaydedilebilir. Problem verilerine yeniden erişmek için 3. seçeneğin seçilmesi ve kütük isminin verilmesi yeterli olacaktır.

Bir diğer uygulama kolaylığı da problem çözümündeki sonuçların sadece ekranda görüntülenmeyip, istenirse bir kütüğe ASCII kodu ile kaydedilebilmesidir. Bu sayede probleme ait çıktıları düzenleyebilme olanağı ve raporlama kolaylığı getirilmiştir.

5. SONUÇ

Darboğaz ulaştırma problemi ve iki amaçlı ulaştırma problemleri, pek çok alanda karşılaşılabilecek nitelikte, önemli karar verme problemleridir. Günümüze kadar bu karar problemleri ile ilgili çok sayıda çözüm algoritması geliştirilmiştir. Yeni yaklaşımların geliştirileceği ve bu alandaki gelişmelerin durmayacağı açıktır.

Bu çalışmayla sözü edilen iki karar probleminin teknik yazımda, oldukça ayrıntılarıyla ele alındığı ve bir dizi çözüm tekniklerinin geliştirilmiş olduğu anlaşılmıştır. Gelişim süreci içerisinde önemli yer tuttuğuna inanılan bazı algoritmalar, çalışmaya dahil edilerek, bu sayede her iki karar probleminin de tarihsel gelişimlerindeki mantık akışı da ortaya konmuştur.

İki amaçlı ulaştırma problemi için geliştirilen teknikler incelenerek Toplam maliyet-Darboğaz zaman tabanlı iki amaçlı bir ulaştırma problemi için etkin bir şekilde sıralı çözümler üreten ve bu çözümler arasından üstün olanlarını kolaylıkla ayıklayarak karar vericiye önerilecek seçeneği belirleyen bir yaklaşımın olmadığı incelenen kaynaklar itibariyle görülmüştür. Çalışma, bu konu üzerinde yoğunlaştırılarak, bir yöntem geliştirilmiş ve işlemler algoritmik yapıya kavuşturulmuştur. Geliştirilen algoritmanın yapısı, diğer çözüm teknikleri ile hızlılık, anlaşılırlık ve etkinlik açısından karşılaştırılabilecek şekilde düzenlenmiştir. Bu çalışmayla, ilk kez iki amaçlı bir ulaştırma problemi için, elde edilen üstün çözüm ikilileri arasından karar vericiye sunulmak üzere bir tanesini belirleyecek olan bir yöntem, Geetha-Vartak ikilisinin üç boyutlu atama problemleri için yaptıkları çalışmalardan esinlenilerek, probleme uyarlanmıştır.

Turbo Pascal programlama dili kullanılarak, algoritma için bir paket program yazılmış; işlemler, kullanıcının bilgisayar hakkında bilgisi olmasını gerektirmeyecek şekilde menülerle düzenlenmiştir. Kullanılan programlama dilinin etkinliği, geliştirilen paketin hızına da

yansıyarak algoritmanın etkinliğini arttırmıştır. Program, 50x50 boyutundaki iki amaçlı bir ulaştırma problemini çözebilecek kapasitededir. Gerekli eklentilerin yapılması ile geliştirilecek yazılımda, hem hız hem de işlem boyutlarının arttırılması yönlü değişiklikleri sağlamak mümkündür. Ayrıca rassal veri türetme aşamasında da, sadece üniform dağılıma bağlı kalınmayarak, diğer dağılımlara göre de veri türetilmesini sağlamak mümkündür. Geliştirilen yazılıma iki önemli işlem seçeneği eklenerek, bu alanda çalışacak araştırmacılara da belli sınırlar içinde kolaylık sağlanmak istenmiştir. Araştımacılara kolaylık sağlaması öngörülen birinci işlem seçeneği, rassal veri türetme olanağıdır. Rassal verilerin türetilmesinde, kullanıcıdan probleme ait kapasite, talep, taşıma maliyeti ve taşıma zamanı değerlerinin alt ve üst sınırları sorulmakta, böylece istenen sayısal özelliklere sahip iki amaçlı ulaştırma problemlerinin rassal olarak türetilmesi olanağı verilmektedir. Bu olanak, farklı sayısal özelliklere sahip iki amaçlı ulaştırma problemleri arasında ne gibi ilişkilerin bulunacağına dair yapılacak bir araştırmada, araştırmacıya büyük kolaylık sağlayacaktır. Diğer yandan, ikinci önemli işlem seçeneği olarak da, problem çözümüne ait sonuçların sadece ekranda görüntülenmeyip, istenirse bir kütükte saklanması olanağı sağlanmıştır. Bu da özellikle raporlama açısından büyük kolaylık getirecektir.

Ulaştırma problemi, doğrusal programlamanın her zaman en çok ilgiyi gören ve uygulama alanı bulan altbaşlığı olmuştur. Darboğaz ulaştırma problemi ve iki amaçlı ulaştırma problemleri için de, şimdiye kadar görülen ilginin devam edeceğini ve yeni yaklaşımların geliştirileceğini söylemek yanlış olmayacaktır. Nitekim 1959 yılından bu yana yapılan çalışmalarda, bu konuya olan ilginin hiç azalmadığı ve özellikle son yıllarda da artarak devam ettiği, yayınlanan makalelerden anlaşılmaktadır. Bu nedenle konunun güncel ve gelişmeye uygun olduğunu söylemek yerinde olacaktır.

Konu ile ilgili bundan sonraki gelişmelerin, özellikle darboğaz zaman süresince yapılacak olan darboğaz taşıma miktarının azaltılmasına yönelik olacağı söylenebilir. Hatta yapılan taşımalarda, taşıma maliyetinin söz konusu olduğu kadar, gecikmelerden doğacak maliyetlerin de varolacağı söylenebilir. Bu durumdan darboğaz zaman değerinin nasıl etkileneceğinin incelenmesi ayrı bir araştırma konusu olacak kadar nitelikli ve geniştir. Benzer şekilde üstün çözüm ikililerinin arasından, darboğaz taşıma miktarını da gözönünde bulundurarak, hangi çözümün seçileceğine dair çeşitli analizler ve araştırmalar yapmak mümkündür. Dahası DUP ve iki amaçlı ulaştırma problemleri için yeni çözüm yaklaşımlarının geliştirilmesine yönelik çalışmalar, araştırmacıları beklemektedir. Çözümlere ait duyarlılık analizlerinin nasıl yapılacağı ve hangi yorumlara ulaşılacağı sorusu da gündemdedir.

Bu çalışmayla, DUP ve iki amaçlı ulaştırma problemi, bundan sonra bu alanda araştırma yapacak kişilere çok yardımcı olacak şekilde ayrıntılarıyla anlatılmış, TM-DZ tabanlı iki amaçlı bir ulaştırma problemi için etkin bir algoritma geliştirilerek, algoritmaya ait bilgisayar programı verilmiştir. Konu, gelişiminin başından beri gördüğü ilgiyi korumakta ve gelecekte araştırmacıları ilgilendirecek pekçok soruyu beraberinde taşımaktadır. Bu alanda ki araştırmalarla, yeni bazı problem türlerinin tanımlanabileceği, farklı yaklaşımların oluşturulabileceği, öncelikle hizmet işletmelerinde verimliliği arttırıcı uygulamaların yapılabileceğini söylemek mümkündür.

KAYNAKLAR DİZİNİ

Ahuja R.K., 1986, Algorithms for the minimax transportation problem, NRLQ, 33, 725-739

Arsham H. and Kahn A.B., 1989, A simplex-type algorithm for general transportation problems: an alternative to stepping stone, J. Opl. Res. Soc., 40, 6, 581-590

Bayburan B., 1988, Turbo Pascal, Beta Basım Yayım Dağıtım A.S., 311 s.

Datta N., 1984, Algorithm to determine on initial efficient basic solution for a linear fractional multiple objective transportation problem, CCERO, 26, 1/2, 127-136

Davidson G., 1982, Practical Pascal Programs, McGraw-Hill, California, 205 p.

Dearing P.M., Newruck F.C., 1979, A capacitated bottleneck facility location problem, Management Science, 25, 11, 1093-1104

Dengs V., 1980, Assignment and matching problems: Solution methods with fortran-programs, Springer-Verlag Berlin Heidelberg, 145

Finke G. and Smith P.A., 1978, Primal equivalents to the threshold algorithm, Op. Res. Verf.,31, 185-198

Garfinkel R.A. and Rao M.R., 1971, The bottleneck transportation problem, NRLQ, 18, 465-472

Geetha S. and Vartak M.N., 1989, Time-cost trade-off in a three dimensional assignment problem, EJOR, 38, 255-258

Geetha S. and Vartak M.N., 1989, Time-cost trade-off analysis in some constrained assignment problems, J. Opl. Res. Soc., 40, 1, 97-101

Grabowski W., 1964, Problem of transportation in minimum time, Bulletin De L'academie Polonaise Des Sciences, 12, 2, 107-108

Hammer P.L., 1971, Communication on "The bottleneck transportation problem" and "Some remarks on the time transportation problem"., NRCQ, 18, 487-490

Hammer P.L., 1969, Time minimizing transportations problems, NRLQ, 16, 345-357

Isermann H., 1984, Linear bottleneck transportation problems, Asia-Pacific Journal of Operational Research, 1, 38-52

KAYNAKLAR DIZINI (devam ediyor)

Jamsa K., Nameroff S., 1989, Örneklerle turbo pascal, (Çev. G. Banger), Bilim Teknik Yayınevi, 514 s.

Khanna S., Bakhshi H.C., Arora S.R., 1983, Time minimization transportation problem with restricted flow, CCERO, 25, 1/2, 65-74

Lee M.S. and Moore L.J., 1973. Optimizing transportation problems with multiple objectives, AIIE Transactions, 5, 4, 333-338

Malhotra R., 1983, On Hammer's method of finding a least cost optimal solution to a time minimization problem, CCERO, 25, 1/2, 75-80

Nagelhout R. and Thamson G.L., 1984, A study of the bottleneck single source transportation problem, Comput & Op. Res., 11, 1, 25-36

Phillipps D.T., Ravindran A. and Solberg J., 1976, Operations Research: principles and practice, John Wiley Sons & Inc., Canada, 585 p.

Price W.L., 1971, Graphs and networks, Butterworth & Co Ltd., London, 108 p.

Ramakrishnan C.S., 1988, An improvement to goyal's modified VAM for the unbalanced transportation problem, J. Opl. Res. Soc., 39, 6, 609-610

Russel R.A., Klingman D.D. and Portow-Navid P., 1983, An efficient primal approach to bottleneck transportation problems, NRLQ, 30, 13-35

Satır A. and Kırca Ö., 1988, A heuristic for obtaining an initial solution for the transportation problem, EURO IX, TIMS XXVIII, International Conference, July 6-8.1988, Paris

Scheid, Francis, 1982, Computers and Programming, McGraw Hill Co., New York, 402 p.

Smith D.V., 1982, Network optimisation practice, A Computational Guide, John Wiley & Sons, New York 237 p.

Srinivasan V. and Thompson G.L., 1976, Algorithms for minimizing total cost, bottleneck time and bottleneck shipment in transportation problems, NRLQ, 23, 567-595

Szwarc W., 1971, Some remarks on the time transportation problems, NRLQ, 16, 483-485

Taha H.A.,1971, Operations Research an introduction, Mac Millan Publishing Co. Inc., New York, 129, 702 p.

EK I: Programa ait menüler

```
          IKI AMAÇLI ULASTIRMA PROBLEMI

   1  Yeni Problem Girisi
   2  Problem Kaydetme
   3  Disketten Problem Okuma
   4  Rassal Veri Türetme
   5  Problemin Çözümü
   6  Sonuçlarin Gösterilmesi
   7  Verilerin Tabloda Gösterimi
   8  Sonucun Tabloda Gösterimi
   9  Sonuclarin Yazdırılması
   0  Programdan Çıkıs

Yön Tusları ve [ENTER] ile Seçiminizi Belirtin..!
```

```
    SONUCLARIN GÖSTERILMESI

   1  Ekranda Göster

   2  Yazıcıdan Döküm

   3  Kütüge Kaydet

    Seçiminizi Belirtin..!
```

EK II: Geliştirilen Algoritmanın Bilgisayar Programı

```
     Program Iki_Amacli_Ulastirma_Modeli;
     uses crt,printer;
     const
       max=40;
type
    MaxAralik=0..max;
    CiftAralik=0..102;
    Elemanlar= Record
                Visited,Stone:Boolean;
                Maliyet:Real;
                Miktar:Real;
                Zaman:Real;
    End;
     type
    String79=string[79];
    Cevaplar=(CEVAPSIZ,YONLER,TUS,RETURN);
    Hareketler=(YOK,SOL,SAG,YUKARI,ASAGI);
     type
     MenuKayit = Record
           SecenekSayisi:Shortint;
           Genislik:Shortint;
           Secenekler:array [1..14] of char;
           Tanimlama:array [1..14] of string79;
           Baslik:string79;
           Mesaj:string79;
     End;
     Yol = Record
        icord:MaxAralik;
        jcord:MaxAralik;
End;
     UlastirmaBoyutu=Array[MaxAralik,MaxAralik]                of
Elemanlar;
HeaderArray= Array[MaxAralik] of Real;
TraceArray= Array[CiftAralik] of Yol;
     Var
    Tablo:UlastirmaBoyutu;
    RowHead,ColumnHead:HeaderArray;
    Source,Destination:HeaderArray;
    LastSource,LastDestin: 1..51;
    TotalSource,TotalDestin:Real;
    s,t,row,col:integer;
    LeastIndex:Real;
    Trace:TraceArray;
    NumStones,LastStone,count:CiftAralik;
    LeastAmount:Real;
    Secim,Cevap,Sonuc:char;
    TotalCost,MaxTime,BigM:Real;
    Ardistirma,CozumNo:Shortint;
    Son:Text;
     const
      AnaMenu : MenuKayit = (
          SecenekSayisi:10;
          Genislik:50;
          Secenekler:'1234567890    ';
          Tanimlama:('Yeni Problem Girisi',
                     'Problem Kaydetme',
                     'Disketten Problem Okuma',
                     'Rassal Veri Türetme',
                     'Problemin Çözümü',
                     'Sonuçlarin Gösterilmesi',
                     'Verilerin Tabloda Gösterimi',
                     'Sonucun Tabloda Gösterimi',
```

```
                    'Sonuclarin Yazdırılması',
                    'Programdan Çıkıs','','','','');
        Baslik:'IKI AMAÇLI ULASTIRMA PROBLEMI';
          Mesaj:'Yön  Tusları ve  [ENTER] ile  Seçiminizi
Belirtin..!');
    const
     SonucMenu : MenuKayit = (
        SecenekSayisi:3;
        Genislik:30;
        Secenekler:'123             ';
        Tanimlama:('Ekranda Göster',
                    'Yazıcıdan Döküm',
                                                'Kütüge
Kaydet','','','','','','','','','','','');
        Baslik:'SONUCLARIN GÖSTERILMESI';
        Mesaj:'Seçiminizi Belirtin..!');
    Procedure SetVideo(Atribute:Shortint);
var
   Blink,Bold:shortint;
Begin
     Blink:=(Atribute and 4)*4;
     if (Atribute and 2) = 2 then
        Begin
             Bold:=(atribute and 2) * 7;
             TextColor(1+Blink+Bold);
             TextBackGround(3);
        End
     Else
        Begin
             Bold:=(atribute and 2) * 5 div 2;
             TextColor(7+Blink+Bold);
             TextBackGround(0);
        End;
End;
    Procedure PutString (OutString:string79;
                    Line,Col,Atribute:Shortint);
Begin
     SetVideo(Atribute);
     GotoXY(Col,Line);
     Write(OutString);
     SetVideo(0);
End;
    Procedure PutMesaj (OutString:string79;
                    Line,Col:Shortint);
Begin
     GotoXY(Col,Line);
     ClrEol;
     PutString(OutString,Line,Col,3);
End;
    Procedure PutCenter(OutString:string79;
                    Line,Atribute:Shortint);
Begin
       PutString(OutString,Line,40-Length(OutString) div  2
,Atribute);
End;
    Procedure Cerceve(SatirSayisi,SutunSayisi:ShortInt);
Var
   SolSutun,UstSatir,Sutun,Satir:Shortint;
Begin
     if SatirSayisi>20 Then SatirSayisi:=20;
     if SutunSayisi>73 Then SutunSayisi:=73;
     SetVideo(3);
     UstSatir:=12-SatirSayisi div 2 - 3;
     if UstSatir<1 Then UstSatir:=1;
     SolSutun:=40-SutunSayisi div 2-1;
     if SolSutun<1 then SolSutun:=1;
```

```
      GotoXY(SolSutun,UstSatir);
      Write('',chr(218));
      For Sutun:=1 to SutunSayisi+2 do
          Write(Chr(196));
      Write(Chr(191),'');
      For Satir:= UstSatir+1 to UstSatir+SatirSayisi+3 do
          Begin
               GotoXY(SolSutun,Satir);
               Write('',Chr(179),'');
               GotoXY(SolSutun+Sutunsayisi+3,Satir);
               WriteLn('',Chr(179),'');
          End;
      GotoXY(SolSutun,UstSatir+SatirSayisi+4);
      Write('',chr(192));
      For Sutun:=1 to SutunSayisi+2 do
          Write(Chr(196));
      Write(Chr(217),'');
      SetVideo(0);
End;
    Procedure Cerceveiki(x1,y1,x2,y2:Shortint);
Var i:Shortint;
Begin
      GotoXY(x1,y1); Write(#201);
      For i:=x1+1 to x2-1 do write(#205);
      Write(#187);
      For i:=y1+1 to y2-1 do
          begin
               GotoXY(x1,i); Write(#186);
               GotoXY(x2,i); Write(#186);
          end;
      GotoXY(x1,y2); Write(#200);
      For i:=x1+1 to x2-1 do write(#205);
      Write(#188);
End;
    Procedure MenuyuGoster(Menu:MenuKayit);
Var
   SolSutun,Ustsatir,BosSatir,i:integer;
Begin
      Clrscr;
      BosSatir:=1;
      With menu do
           begin
              if SecenekSayisi < 7 then BosSatir:=2;
              UstSatir:=10-(SecenekSayisi*BosSatir) div 2 -
                        (2-Bossatir)*SecenekSayisi mod 2;
              SolSutun:=41-Genislik div 2;
              Cerceve(SecenekSayisi*BosSatir+3,Genislik);
              PutCenter(Baslik,UstSatir+1,1);
              for i:=1 to SecenekSayisi do
                 begin
                   PutString(Secenekler[i],UstSatir+i*BosSa
tir+2,SolSutun+2,1);
                                                    PutString('
'+Tanimlama[i],UstSatir+i*BosSatir+2,SolSutun+4,0);
                 end;

PutCenter(Mesaj,UstSatir+SecenekSayisi*BosSatir+4,2);
           end;
end;
    Procedure CevapOku(Var Cevap:Cevaplar;
                       Var Yon:Hareketler;
                       Var Secim:char);
const
      zil=7;
      Enter=13;
      Esc=27;
      Saga=77;
```

```
      Sola=75;
      Asagiya=80;
      Yukariya=72;
var
   BasilanTus:Char;
Begin
      Cevap:=CEVAPSIZ;
      Yon:=YOK;
      Secim:=' ';
      Repeat
            BasilanTus:=ReadKey;
            if BasilanTus=#0 then
               Begin
                     Cevap:=YONLER;
                     BasilanTus:=ReadKey;
                     if Ord(BasilanTus)=Sola then Yon:=SOL
                           else if Ord(BasilanTus)=Saga  then
Yon:=SAG
                     else if BasilanTus=#72 then Yon:=YUKARI
                     else if BasilanTus=#80 then Yon:=ASAGI
                              else
                                    begin
                                          Cevap:=CEVAPSIZ;
                                          Write(chr(zil));
                                    end;
                End
            else if Ord(BasilanTus)=Enter  then Cevap:=RETURN
            else
                begin
                     cevap:=TUS;
                     Secim:=UpCase(BasilanTus);
                end;
      Until cevap<>CEVAPSIZ;
End;
    Procedure SecimiBul(Menu:MenuKayit;
                        Var Secim:Char);
Var
   AktifSecim,BosSatir,IlkSatir,TanimSutunu,i:shortint;
    Yanit:Cevaplar;
   OkYonu:Hareketler;
    Buldum:Boolean;
Begin
  with Menu do
       begin
         AktifSecim:=1; {SecenekSayisi;}
         for i:=1 to SecenekSayisi do
             if secim=secenekler[i] then AktifSecim:=i;
                if SecenekSayisi>7  then  Bossatir:=1  else
BosSatir:=2;
                   IlkSatir:=12-(SecenekSayisi*BosSatir)  div
2-(2-BosSatir)*SecenekSayisi mod 2;
          TanimSutunu:=47-Genislik div 2;
          repeat
            PutString(Tanimlama[AktifSecim],IlkSatir+AktifS
ecim*BosSatir,TanimSutunu-1,6);
            GotoXY(TanimSutunu-5+Length(Mesaj),IlkSatir+Sec
enekSayisi*BosSatir+2);
            CevapOku(Yanit,Okyonu,Secim);
            PutString(Tanimlama[AktifSecim],IlkSatir+AktifS
ecim*BosSatir,TanimSutunu-1,0);
            Case Yanit of
               YONLER: begin
                                       if (OkYonu=ASAGI)  and
(AktifSecim=SecenekSayisi) then AktifSecim:=1
                                  else if  (OkYonu=ASAGI) then
AktifSecim:=AktifSecim+1
                                  else if  (OkYonu=YUKARI) and
```

```
(AktifSecim=1) then AktifSecim:=SecenekSayisi
                                    else if  (OkYonu=YUKARI) then
AktifSecim:=AktifSecim-1;
                                end;
                    TUS: begin
                                 Buldum:=False;
                                 For i:=1 to SecenekSayisi do
                                     if Secim=Secenekler[i] then
                                          Begin
                                               Buldum:=true;
                                               AktifSecim:=i;
                                          end;
                                  if Buldum then Yanit:=RETURN else
write(chr(7));
                            end;
                     RETURN: Secim:=Secenekler[AktifSecim];
                end;
                PutString(Tanimlama[AktifSecim],IlkSatir+AktifS
ecim*BosSatir,TanimSutunu-1,1);
           until Yanit=RETURN;
        end;
end;
    Procedure InputData;
Var
   i,j:Byte;
   InputNum:Real;
Begin
    ClrScr;
          Write('Üretim      Merkezi    Sayısı    <50   :    ');
Readln(LastSource);
           Write('Dagıtım    Bölgesi   Sayısı    <50   :    ');
Readln(LastDestin);
    Writeln;
    TotalSource:=0;
    For i:=1 to LastSource do
        Begin
             Write(i,'. Üretim Merkezinin Kapasitesi : ');
             Readln(InputNum);
             Source[i]:=InputNum;
             TotalSource:=TotalSource+InputNum;
        End;
     TotalDestin:=0;
     WriteLn;
     For j:=1 to LastDestin do
         Begin
              Write(j,'. Dagıtım Bölgesinin Talebi : ');
              Readln(Destination[j]);
              TotalDestin:=TotalDestin+Destination[j];
         End;
     Write(#7);
End;
    Procedure InputCost;
Var
   i,j:Byte;
   InputNum:Real;
Begin
    ClrScr;
          Writeln('--------------------  Tasıma   Maliyetleri
--------------------');
     WriteLn;
     For i:=1 to LastSource Do
       Begin
         For j:=1 to LastDestin Do
             Begin
                  Write('Üretim  Merkezi ',i,''den Dagıtım
Bölgesi ',j,''e : ');
                  Readln(InputNum);
```

```
                    Tablo[i,j].Maliyet:=InputNum;
                    Tablo[i,j].Stone:=false;
                    Tablo[i,j].Visited:=false;
                    Tablo[i,j].Miktar:=0;
                End;
        End;
        Write(#7);
End;
    Procedure InputTime;
Var
   i,j:byte;
   InputNum:Real;
Begin
     ClrScr;
              WriteLn('------------------   Tasıma   Zamanları
------------------');
     WriteLn;
     For i:=1 to LastSource Do
       Begin
         For j:=1 to LastDestin Do
             Begin
                 Write('Üretim  Merkezi ',i,''den Dagıtım
Bölgesi ',j,''e : ');
                 ReadLn(InputNum);
                 Tablo[i,j].Zaman:=InputNum;
             End;
         WriteLn;
       End;
       Write(#7);
End;
    Procedure BaslangicTablo;
Var
   i,j:Byte;
Begin
     For i:=1 to LastSource do
       begin
         For j:=1 to LastDestin do
           begin
             Tablo[i,j].Miktar:=0;
             Tablo[i,j].Stone:=False;
             Tablo[i,j].Visited:=False;
             If Tablo[i,j].Maliyet>=BigM then

Tablo[i,j].Maliyet:=Tablo[i,j].Maliyet-BigM;
           end;
       end;
     TotalCost:=0;
End;
    Procedure InputRandom;
Var
 i,j:Byte;
   a,b,c,d,e,f:integer;
Begin
     ClrScr;
     WriteLn('Verilerin Rassal Olusturulması için');
     WriteLn('Düzgün Dagılım Fonksiyonu Kullanılmaktadır.');
     WriteLn; WriteLn; WriteLn;
       Write('Üretim   Merkezi        Sayısı   <50   :    ');
ReadLn(LastSource);
          Write('Dagıtım   Bölgesi  Sayısı   <50   :    ');
ReadLn(LastDestin);
     WriteLn;WriteLn;
       Write('Kapasite ve Taleplerin  Degisim Aralıgı  (a-b)
:');ReadLn(a,b);
     WriteLn;
       Write('Tasıma Maliyetlerinin Degisim  Aralıgı   (a-b)
:');ReadLn(c,d);
```

```
     WriteLn;
      Write('Tasıma  Zamanlarının Degisim Aralıgı       (a-b)
:');ReadLn(e,f);
     WriteLn;
     TotalSource:=0;
     For i:=1 to LastSource do
          Begin
               Source[i]:=a+Random(b-a);
               TotalSource:=TotalSource+Source[i];
          End;
     TotalDestin:=0;
     For j:=1 to LastDestin do
          Begin
               Destination[j]:=a+Random(b-a);
               TotalDestin:=TotalDestin+Destination[j];
          End;
     For i:=1 to LastSource Do
          For j:=1 to LastDestin Do
              Begin
                   Tablo[i,j].Maliyet:=c+Random(d-c);
                   Tablo[i,j].Stone:=false;
                   Tablo[i,j].Visited:=false;
                   Tablo[i,j].Miktar:=0;
              End;
     For i:=1 to LastSource Do
          For j:=1 to LastDestin Do
                   Tablo[i,j].Zaman:=e+Random(f-e);
End;
     Procedure CizgiCiz;
Var
   i,j:MaxAralik;
Begin
      For j:=1 to LastDestin+1 do
          Write(Son,'----------+');
      WriteLn(Son,'----------');
End;
     Procedure ListSolution;
Var
   i,j:Byte;
Begin
      if (LastSource>7) or (LastDestin>10) then
          begin
               WriteLn(#7);
               exit;
          end;
      ClrScr;
      Write(Son,'S \ D |');
      For j:=1 to LastDestin do
          Write(Son,'  D',j,'   |');
      WriteLn(Son,' Kapasite');
      CizgiCiz;
      For i:=1 to LastSource do
          Begin
               Write(Son,' S',i,'    |');
               For j:=1 to LastDestin do
               Write(Son,Tablo[i,j].Maliyet:3:0,('|',)Tablo[
i,j].Zaman:3:0,'|');
               WriteLn(Son,Source[i]:6:0);
               Write(Son,'      |');
               For j:=1 to LastDestin do
                   if Tablo[i,j].Miktar>0 Then
                      Write(Son,Tablo[i,j].Miktar:4:0,'   |')
                   else
                      Write(Son,' ':4,'   |');
               WriteLn(Son);
               CizgiCiz;
          End;
```

```
      Write(Son,'Talep |');
      For j:=1 to LastDestin do
          Write(Son,Destination[j]:6:0,'|');
      WriteLn(Son,TotalSource:4:0,'\',TotalDestin:4:0);
      WriteLn(Son);
      WriteLn(Son,'Toplam Maliyet :',TotalCost:10:2);
      if sonuc='3' then
            WriteLn(Son)
      else
           begin
              GotoXY(1,25);
                        Write('Hazır  Olduğunuzda  Bir  Tusa
Basınız!..');
              Repeat
                  Cevap:=ReadKey;
              Until Cevap<>'';
              ClrScr;
           end;
End;
    Procedure ReadData;
Var
   F:Text;
   FileName:String[20];
   i,j:Byte;
   r:real;
   k,t:integer;
   Deger1,Deger2:String[10];
Begin
    window(50,10,80,16);
    ClrScr;
    cerceveiki(1,1,31,6);
    GotoXY(3,1);
    WriteLn(' Disk(et)ten Program Okuma ');
    window(52,12,77,14);
    WriteLn('Veri Kütügünün Adı ?');
    Write('_____________________');
    GotoXY(1,2);
    Readln(FileName);
    Assign(F,FileName);
    Reset(F);
    if IOResult<>0 Then
       Begin
            ClrScr;
            WriteLn('Kütük Diskette Bulunamadı');
            GotoXY(24,30);
            Write('Bir Tusa Basınız!..');
            Repeat
               Cevap:=ReadKey;
            Until Cevap<>'';
            window(1,1,80,25);
            Exit;
       End;
    ReadLn(F,Deger1,Deger2);
    Val(Deger1,t,k);
         if  k=0  then  LastSource:=t   Else  Write('Hatali
Veriler!..');
    Val(Deger2,t,k);
         if  k=0  then  LastDestin:=t   Else  Write('Hatali
Veriler!..');
    TotalSource:=0;
    For i:=1 to LastSource do
      Begin
           ReadLn(F,Deger1);
           Val(Deger1,r,k);
           if k=0 then
              begin
                  Source[i]:=r;
```

```
                    TotalSource:=TotalSource+Source[i];
                end
            Else Write('Hatalı Veriler!..');
        End;
    TotalDestin:=0;
    For j:=1 to LastDestin do
      Begin
            ReadLn(F,Deger2);
            Val(Deger2,r,k);
            if k=0 then
               begin
                    Destination[j]:=r;
                    TotalDestin:=TotalDestin+Destination[j];
               end
            Else Write('Hatalı Veriler!..');
        End;
    For i:=1 to LastSource Do
        For j:=1 to LastDestin Do
           Begin
                ReadLn(F,Deger1,Deger2);
                Val(Deger1,r,k);
                if k=0 then
                   Begin
                        Tablo[i,j].Maliyet:=r;
                        Tablo[i,j].Miktar:=0;
                        Tablo[i,j].Stone:=False;
                        Tablo[i,j].Visited:=False;
                   end
                Else Write('Hatali Veriler!..');
                Val(Deger2,r,k);
                  if  k=0 then Tablo[i,j].Zaman:=r  Else
Write('Hatali Veriler!..');
           End;
    Close(F);
    window(1,1,80,25);
End;
    Procedure Dummy;
Var
   i,j:byte;
Begin
     If TotalSource > TotalDestin Then
        Begin
             LastDestin:=LastDestin+1;
             For i:=1 to LastSource do
                 Begin
                      Tablo[i,LastDestin].Maliyet:=0;
                      Tablo[i,LastDestin].Stone:=false;
                      Tablo[i,LastDestin].Miktar:=0;
                      Tablo[i,LastDestin].Visited:=false;
                      Tablo[i,LastDestin].Zaman:=0;
                 End;
                    Destination[LastDestin]:=TotalSource  -
TotalDestin;
             TotalDestin:=TotalSource;
        End
     Else
        Begin
             LastSource:=LastSource+1;
             For j:=1 to LastDestin do
                 Begin
                      Tablo[LastSource,j].Maliyet:=0;
                      Tablo[LastSource,j].Stone:=false;
                      Tablo[LastSource,j].Miktar:=0;
                      Tablo[LastSource,j].Visited:=false;
                      Tablo[LastSource,j].Zaman:=0;
                 End;
             Source[LastSource]:=TotalDestin - TotalSource;
```

```
                TotalSource:=TotalDestin;
          End;
End;
     Procedure ToplamMaliyet;
Var
     i,j:Byte;
Begin
      TotalCost:=0;
      For i:=1 to LastSource do
        For j:=1 to LastDestin do
            Begin
                                        If  (Tablo[i,j].stone)   and
(Tablo[i,j].Maliyet>0) then

TotalCost:=TotalCost+Tablo[i,j].Miktar*Tablo[i,j].Maliyet;
            end;
End;
     Procedure EnbuyukZaman;
Var
    i,j:MaxAralik;
Begin
      MaxTime:=0;
      For i:=1 to LastSource do
          For j:=1 to LastDestin do
              Begin
                                           If  Tablo[i,j].Stone  and
(MaxTime<Tablo[i,j].Zaman) Then
                      Begin
                           MaxTime:=Tablo[i,j].Zaman;
                      End;
                End;
End;
     Procedure NorthWest;
Var
    i,j:Byte;
    Srem,Drem:Real;
Begin
      i:=1;
      j:=1;
      Srem:=Source[i];
      Drem:=Destination[j];
      While (i<=LastSource) and (j<=LastDestin) Do
           Begin
                If Srem>Drem Then
                   Begin
                        Tablo[i,j].Miktar:=Drem;
                        Tablo[i,j].Stone:=true;
                        Srem:=Srem-Drem;
                        j:=j+1;
                                          If j<=LastDestin  Then
Drem:=Destination[j];
                   End
                Else
                   If Srem<Drem Then
                      Begin
                           Tablo[i,j].Miktar:=Srem;
                           Tablo[i,j].Stone:=true;
                           Drem:=Drem-Srem;
                           i:=i+1;
                                          If j<=LastSource  Then
Srem:=Source[i];
                      End
                   Else
                      Begin
                           Tablo[i,j].Miktar:=Srem;
                           Tablo[i,j].Stone:=true;
                           j:=j+1;
```

```
                                If j<=LastDestin Then
                                   Begin
                                        Tablo[i,j].Stone:=true;
                                        Drem:=Destination[j]
                                   End;
                                i:=i+1;
                                        If i<=LastSource  Then
Srem:=Source[i]
                          End
               End
End;
     Procedure DoRow(i,PrevCol:MaxAralik);forward;
Procedure DoCol(PrevRow,j:MaxAralik);
Var
    i:Byte;
Begin
      i:=0;
      While (i<LastSource) and (Count<NumStones) do
               Begin
                    i:=i+1;
                    if Tablo[i,j].stone and (i<>PrevRow) Then
                       Begin
                            Count:=Count+1;

RowHead[i]:=Tablo[i,j].Maliyet-ColumnHead[j];
                            DoRow(i,j);
                       End;
               End;
End;
     Procedure DoRow;
Var
    j:Byte;
Begin
      j:=0;
      While (j<LastDestin) and (Count<NumStones) do
               Begin
                    j:=j+1;
                    if Tablo[i,j].stone and (j<>PrevCol) Then
                       Begin
                            Count:=Count+1;

ColumnHead[j]:=Tablo[i,j].Maliyet-RowHead[i];
                            DoCol(i,j);
                       End;
               End;
End;
     Procedure Improveindex;
Var
    i,j:Byte;
    index:real;
Begin
      LeastIndex:=0;
      For i:=1 to LastSource do
          for j:=1 to LastDestin do
              Begin
                    if not Tablo[i,j].stone then
                       begin

index:=Tablo[i,j].Maliyet-Rowhead[i]-Columnhead[j];
                                if index<leastindex then
                                   begin
                                        leastindex:=index;
                                        row:=i;
                                        col:=j;
                                    end;
```

```
                        end;
                end;
end;
    Function
NextColumn(i,prevcol:MaxAralik):boolean;forward;
Function NextRow(prevrow,j:MaxAralik):boolean;
Var
    i:Byte;
    token:boolean;
Begin
      i:=1;
      token:=false;
      repeat
            if (Tablo[i,j].stone and not Tablo[i,j].visited)
Then
               Begin
                     Tablo[i,j].visited:=true;
                     token:=NextColumn(i,j);
                     if token then
                        Begin
                              LastStone:=Laststone+1;
                              Trace[LastStone].icord:=i;
                              Trace[Laststone].jcord:=j;
                              if Tablo[i,j].Miktar<LeastAmount
Then

LeastAmount:=Tablo[i,j].Miktar;
                        End;
                     Tablo[i,j].visited:=false;
                End;
            i:=i+1;
      Until (i>LastSource) or token;
      NextRow:=token;
End;
    function NextColumn;
Var
    j:Byte;
    token:boolean;
Begin
      j:=1;
      token:=false;
      Repeat
            if (Tablo[i,j].stone and not Tablo[i,j].visited)
Then
               Begin
                     Tablo[i,j].visited:=true;
                     if Tablo[row,j].stone then
                        begin
                              trace[1].icord:=row;
                              trace[1].jcord:=j;

LeastAmount:=Tablo[row,j].Miktar;
                              trace[2].icord:=i;
                              trace[2].jcord:=j;
                              token:=true;
                        end
                     else
                        begin
                              token:=NextRow(i,j);
                              if token then
                                 begin
                                       Laststone:=Laststone+1;

trace[laststone].icord:=i;

trace[laststone].jcord:=j;
                                    end;
```

```
                        end;
                    Tablo[i,j].visited:=false;
                end;
            j:=j+1;
      until (j>lastdestin) or token;
      nextcolumn:=token;
End;
    Procedure NextSolution;
Var
   Step:CiftAralik;
   double:boolean;
   x,y:Byte;
Begin
      double:=false;
      Tablo[row,col].Miktar:=ABS(LeastAmount);
      Tablo[row,col].stone:=true;
      for step:=1 to laststone do
          begin
              LeastAmount:= -LeastAmount;
               x:=trace[step].icord;
               y:=trace[step].jcord;

Tablo[x,y].Miktar:=ABS(Tablo[x,y].Miktar+LeastAmount);
                  if (not double)  and (Tablo[x,y].Miktar=0)
then
                  begin
                       double:=true;
                       Tablo[x,y].stone:=false;
                  end;
          end;
End;
    Procedure Optimum;
Var
   best:boolean;
   joke:boolean;
Begin
      Ardistirma:=1;
      best:=false;
      while not best do
            begin
                 Ardistirma:=Ardistirma+1;
                 RowHead[1]:=0;
                 count:=0;
                 DoRow(1,0);
                 ImproveIndex;
                 if LeastIndex<0 then
                    begin
                         LastStone:=2;
                          joke:=NextRow(row,col);
                         NextSolution;
                    end
                 else
                    best:=true;
            end;
            ToplamMaliyet;
            EnbuyukZaman;
            LeastAmount:=0;
end;
    Procedure Degisiklik;
Var
  i,j,satir,sutun:Byte;
  istek:Char;
  YeniDeger,Fark:Real;
Begin
      Repeat
       GotoXY(1,23);
      Write('Degisiklik Nerde?  [ M / Z / K / T / Cıkıs ]');
```

```
      ClrEol;
      GotoXY(1,24);
      ClrEol;
      istek:=ReadKey;
      istek:=UpCase(istek);
      Case istek of
           'M' : begin
                      GotoXY(1,24);
                      Write('satir <',LastSource:2,' : _____
sutun <',
                                  LastDestin:2  ,' : _____  Yeni
Maliyet : ______');
                      GotoXY(13,24);
                      Repeat
                            Read(satir);
                      until satir<=LastSource;
                      GotoXY(32,24);
                      repeat
                            Read(sutun);
                      until sutun<=LastDestin;
                      GotoXY(54,24);
                      ReadLn(YeniDeger);
                      Tablo[satir,sutun].Maliyet:=Yenideger;
                      ToplamMaliyet;
                end;
           'Z' : begin
                      GotoXY(1,24);
                      Write('satir <',LastSource:2,' : _____
sutun <',
                                  LastDestin:2  ,' : _____  Yeni
Zaman : ______');
                      GotoXY(13,24);
                      Repeat
                            Read(satir);
                      until satir<=LastSource;
                      GotoXY(32,24);
                      repeat
                            Read(sutun);
                      until sutun<=LastDestin;
                      GotoXY(52,24);
                      ReadLn(YeniDeger);
                      Tablo[satir,sutun].Zaman:=Yenideger;
                      EnbuyukZaman;
                end;
           'K' : begin
                      GotoXY(1,24);
                      Write('satir <',LastSource:2,' : _____
Yeni Kaynak : ______');
                      GotoXY(13,24);
                      Repeat
                            Read(satir);
                      until satir<=LastSource;
                      GotoXY(35,24);
                      ReadLn(YeniDeger);
                      Fark:=Source[satir]-YeniDeger;
                      Source[satir]:=YeniDeger;
                      TotalSource:=TotalSource+Fark;
                end;
           'T' : begin
                      GotoXY(1,24);
                          Write('sütun <',LastDestin:2  ,' :
_____  Yeni Talep : _____');
                      GotoXY(13,24);
                      repeat
                            Read(sutun);
                      until sutun<=LastDestin;
                      GotoXY(34,24);
                      ReadLn(YeniDeger);
```

```
                        Fark:=Destination[sutun]-YeniDeger;
                        Destination[sutun]:=YeniDeger;
                        TotalDestin:=TotalDestin+Fark;
                    end;
            'C' : Exit;
           Else Write(chr(7));
      end;
      If TotalDestin <> TotalSource then Dummy;
      until istek='C';
End;
    Procedure VeriListesi;
Var
   i,j:Byte;
Begin
      ClrScr;
         WriteLn(Son,LastSource,'X',LastDestin,'  Modelinin
Verileri.');
      Writeln(Son);
        Writeln(Son,'                                        Tasıma
Tasıma');
         Writeln(Son,'Dagıtım                                Maliyeti
Zamanı');

WriteLn(Son,'------------------------------------------------------');
      For i:=1 to LastSource do
        Begin
          Write(Son,'S',i,'(',Source[i]:3:0,') -> ');
          For j:=1 to lastdestin do
            Begin
                  if j>1 then Write(Son,'            ');
                  if i=1 then

Write(Son,'D',j:2,'(',Destination[j]:3:0,')')
                       else
                           Write(Son,'D',j:2,'      ');
                   Write(Son,Tablo[i,j].Maliyet:10:2);
                   WriteLn(Son,Tablo[i,j].Zaman:10:2);
                   WriteLn(Son);
             end;
        if sonuc='3' then
             WriteLn(Son)
        else
           begin
              GotoXY(1,25);
                   Write('Hazır  Oldugunuzda  Bir  Tusa
Basınız!..');
              Repeat
                  Cevap:=ReadKey;
              Until Cevap<>'';
              ClrScr;
           end;
         end;
End;
    Procedure ListData;
Var
   i,j:Byte;
   istek:Char;
Begin
      ClrScr;
      Write(Son,'S \ D |');
      For j:=1 to LastDestin do
          Write(Son,'  D',j,'  |');
      WriteLn(Son,' Kapasite');
      CizgiCiz;
      For i:=1 to LastSource do
          Begin
```

```
                Write(Son,' S',i,'    |');
                For j:=1 to LastDestin do
                  begin
Write(Son,Tablo[i,j].Maliyet:2:0,Tablo[i,j].Zaman:4:0,'|');
                  end;
                WriteLn(Son,Source[i]:6:0);
                CizgiCiz;
          End;
      Write(Son,'Talep  |');
      For j:=1 to LastDestin do
          Write(Son,Destination[j]:6:0,'|');
      WriteLn(Son,TotalDestin:6:0,'\',TotalSource:6:0);
          if sonuc='1' then
           begin
             GotoXY(1,23);
              Write('Degisiklik  Yapmak Istermisiniz?..[E/H]
');
             Istek:=ReadKey;
             istek:=UpCase(istek);
             If istek='E' then Degisiklik;
           end
      else WriteLn(Son);
End;
    Procedure Print;
Var
   i,j:Byte;
   OneCost:real;
Begin
      ClrScr;
      WriteLn(Son,CozumNo,'. Cözüm (',Ardistirma,')');
      Writeln(Son);
      Writeln(Son,'          Yapılacak        Birim      Toplam
Zaman');
      writeln(Son,'Dagıtım      Tasıma      Maliyet     Maliyet
Degerleri ');
      WriteLn(Son,'-----------------------------------------------
-----------');
      For i:=1 to LastSource do
        Begin
          For j:=1 to lastdestin do
            Begin
                                If  (Tablo[i,j].stone)   {and
(Tablo[i,j].Maliyet>0)} then
                 Begin

Onecost:=Tablo[i,j].Miktar*Tablo[i,j].Maliyet;
                                    Write(Son,'S',i,'  ->','
D',j,Tablo[i,j].Miktar:10:2);
                  Write(Son,Tablo[i,j].Maliyet:11:2);
                  Write(Son,onecost:11:2);
                  Write(Son,Tablo[i,j].Zaman:11:2);
                       if (MaxTime=Tablo[i,j].Zaman)  then
WriteLn(Son,' <==') else WriteLn(Son);
                 end;
            end;
        End;
      WriteLn(Son,'-----------------------------------------------
-----------');
       Writeln(Son,'                                Toplam  Maliyet
:',totalcost:10:2);
      if sonuc='3' then
            WriteLn(Son)
      else
          begin
            GotoXY(1,25);
                     Write('Hazır  Oldugunuzda  Bir  Tusa
```

```
Basınız!..');
              Repeat
                  Cevap:=ReadKey;
              Until Cevap<>'';
              ClrScr;
           end;
end;
    Procedure SaveData;
Var
   F:Text;
    FileName:String[20];
    i,j:Byte;
Begin
    window(50,9,80,15);
    ClrScr;
    cerceveiki(1,1,31,6);
    GotoXY(3,1);
    WriteLn(' Disk(et)e Program Kaydetme ');
    window(52,11,77,13);
    WriteLn('Veri Kütügünün Adı?');
    Write('_____________________');
    GotoXY(1,2);
    Readln(FileName);
    window(1,1,80,25);
    Assign(F,FileName);
    Rewrite(F);
    WriteLn(F,LastSource:10,LastDestin:10);
    For i:=1 to LastSource do
        WriteLn(F,Source[i]:10:2);
    For j:=1 to LastDestin do
        WriteLn(F,Destination[j]:10:2);
    For i:=1 to LastSource Do
        For j:=1 to LastDestin Do

WriteLn(F,Tablo[i,j].Maliyet:10:2,Tablo[i,j].Zaman:10:2);
    Close(F);
End;
    Procedure DarBogaz;
Var
    i,j,k:integer;
    C,T:Array[1..120] of Real;
    Procedure ikililer;
Var
    i:integer;
Begin
     ClrScr;
     WriteLn(Son,'Cözüm       Toplam      Enbüyük');
     WriteLn(Son,'   No      Maliyet        Zaman');
     WriteLn(Son,'-------------------------------');
     for i:=1 to CozumNo do
         writeLn(Son,i:5,C[i]:12:2,T[i]:12:2);
     if sonuc='3' then
           WriteLn(Son)
     else
          begin
            GotoXY(1,25);
                   Write('Hazır  Oldugunuzda  Bir  Tusa
Basınız!..');
              Repeat
                  Cevap:=ReadKey;
              Until Cevap<>'';
           end;
end;
    Procedure UstunCozum;
Var
    i,j,n:Byte;
    C2,T2:Array[1..100] of Real;
```

```
    Begin
       i:=1;
       C2[i]:=C[i];
       T2[i]:=T[i];
       for j:=2 to CozumNo do
         begin
           if C2[i]=C[j] Then
              begin
                  C2[i]:=C[j];
                  T2[i]:=T[j];
              end
            else
              begin
                    i:=i+1;
                   C2[i]:=C[j];
                   T2[i]:=T[j];
              end;
          end;
       CozumNo:=i;
       For j:=1 to CozumNo do
           Begin
                C[j]:=C2[j];
                T[j]:=T2[j];
           end;
End;
    Procedure OnerilenCozum;
Var
   Rq:Array[1..120] of real;
   i,j,q:Byte;
   enkRq:Real;
Begin
     For i:=1 to CozumNo-1 do
       begin
         Rq[i]:=abs((C[i+1]-C[i])/(T[i+1]-T[i]));
       end;
     enkRq:=Rq[1];
     q:=0;
     For i:=1 to CozumNo-1 do
         if enkRq>Rq[i] Then
            begin
                 enkRq:=Rq[i];
                 q:=i;
            end;
     WriteLn(son);
     GotoXY(1,22);
           Writeln(son,'Önerilen   ÇÖzüm  No  :   ',(q+1),'
('',C[q+1]:8:2,',',T[q+1]:6:2,' )');
     MaxTime:=T[q+1];
     CozumNo:=q+1;
     if sonuc='3' then
           WriteLn(Son)
     else
         begin
           GotoXY(1,25);
                      Write('Hazır  Oldugunuzda  Bir  Tusa
Basınız!..');
           Repeat
                Cevap:=ReadKey;
           Until Cevap<>'';
           ClrScr;
         end;
End;
```

```
    Procedure IstenenCozum;
Var
   i,j:Byte;
Begin
     BaslangicTablo;
     For i:=1 to LastSource do
         For j:=1 to LastDestin do
             if (Tablo[i,j].Zaman>MaxTime)  then

Tablo[i,j].Maliyet:=Tablo[i,j].Maliyet+BigM;
     NorthWest;
     Optimum;
     Print;
End;
    Begin {Darbogaz}
         Repeat
           EnbuyukZaman;
           C[CozumNo]:=TotalCost;
           T[CozumNo]:=MaxTime;
           CozumNo:=CozumNo+1;
           For i:=1 to LastSource do
               For j:=1 to LastDestin do
                 begin
                   Tablo[i,j].Visited:=False;
                   if Tablo[i,j].Miktar<0.01 Then
                       Begin
                           Tablo[i,j].Miktar:=0;
                           Tablo[i,j].Visited:=False;
                           Tablo[i,j].Stone:=False;
                       end;
                    If (Tablo[i,j].Zaman>=MaxTime)  and
(Tablo[i,j].Maliyet<BigM) then

Tablo[i,j].Maliyet:=Tablo[i,j].Maliyet+BigM;
                 end;
           Optimum;
           if TotalCost<BigM then print;
      Until TotalCost>BigM;
      CozumNo:=CozumNo-1;
      ikililer;
      if CozumNo>1 then
         begin
             UstunCozum;
             if CozumNo>1 Then
                begin
                     ikililer;
                    OnerilenCozum;
                end;
         end;
      IstenenCozum;
      For i:=1 to LastSource do
          For j:=1 to LastDestin do
              If Tablo[i,j].Maliyet>=BigM then

Tablo[i,j].Maliyet:=Tablo[i,j].Maliyet-BigM;
End;
Procedure SonucYaz;
Var
   FileName:String[20];
Begin
      MenuyuGoster(SonucMenu);
      SecimiBul(SonucMenu,Sonuc);
      Case Sonuc of
          '1' :  Begin
                   Close(Son);
                   Assign(Son,'con');
                   Rewrite(Son);
```

```
                End;
          '2' :  Begin
                   Close(Son);
                   Assign(Son,'Lst');
                   Rewrite(Son);
                 End;
          '3' :  Begin
                      window(40,9,80,15);
                      ClrScr;
                      cerceveiki(1,1,41,6);
                      GotoXY(7,1);
                      WriteLn(' Sonuclari Disk(et)e Kaydetme
');
                      Window(42,11,77,13);
                      WriteLn('Sonuc Kütügünün Adı?');
                      Write('_____________________');
                      GotoXY(1,2);
                      Readln(FileName);
                      window(1,1,80,25);
                      Close(Son);
                      Assign(Son,FileName);
                      Rewrite(Son);
                 End;
           Else Write(chr(7));
        End;
End;
    {Program}
Begin
     CozumNo:=1;
     BigM:=100000000.0;
     LastSource:=1;
     LastDestin:=1;
     TotalSource:=0;
     TotalDestin:=0;
     Tablo[1,1].Maliyet:=0;
     Tablo[1,1].Miktar:=0;
     Tablo[1,1].Visited:=False;
     Tablo[1,1].Stone:=False;
     LeastAmount:=0;
     Sonuc:='1';
     Assign(son,'con');
     ReWrite(son);
  Repeat
        MenuyuGoster(AnaMenu);
        SecimiBul(AnaMenu,Secim);
        Case secim of
          '1' :  Begin
                   InputData;
                   InputCost;
                   InputTime;
                 End;
          '2' :  SaveData;
          '3' :  ReadData;
          '4' :  InputRandom;
          '5' :  Begin
                    if  (TotalSource=0) and (TotalDestin=0)
then
                      begin
                         ClrScr;
                         Close(Son);
                         exit;
                      end;
                        If TotalDestin <> TotalSource  then
Dummy;
                    if TotalCost>1 then
                      begin
                         BaslangicTablo;
```

```
                                CozumNo:=1;
                             end;
                          NumStones:=LastSource+LastDestin-1;
                          NorthWest;
                          Optimum;
                          Print;
                          Darbogaz;
                       End;
              '6' :  Print;
              '7' :  Begin
                           if (LastDestin>9)  or (LastSource>9)
Then
                              VeriListesi
                           else
                              ListData;
                       End;
              '8' :  ListSolution;
                   '9' :  SonucYaz;
                   'O' :   begin
                             Close(son);
                             ClrScr;
                             Halt;
                    end;
                Else Write(chr(7));
          End;
      until secim='O';
end.
```

EK III: Algoritmanın Uygulaması

Baslangıç Veriler. (Maliyet-Zaman)

S \ D	D1		D2		D3		D4		D5		D6		Kapasite
S1	5	9	9	25	6	11	5	8	5	13	0	0	21
S2	6	20	7	12	6	20	6	12	8	7	0	0	36
S3	7	25	8	12	8	15	7	22	9	6	0	0	36
S4	9	24	6	29	6	11	9	20	8	9	0	0	25
S5	8	8	7	26	5	24	9	10	9	29	0	0	22
S6	5	15	8	18	6	28	8	14	9	14	0	0	37
S7	7	11	6	19	9	5	5	12	5	24	0	0	21
Talep		34		34		28		29		28		45	198\ 198

1. Cözüm (15)

Dagıtım	Yapılacak Tasıma	Birim Maliyet	Toplam Maliyet	Zaman Degerleri	
S1 -> D4	14.00	5.00	70.00	8.00	
S1 -> D5	7.00	5.00	35.00	13.00	
S2 -> D2	9.00	7.00	63.00	12.00	
S2 -> D3	6.00	6.00	36.00	20.00	
S2 -> D4	15.00	6.00	90.00	12.00	
S2 -> D6	6.00	0.00	0.00	0.00	
S3 -> D6	36.00	0.00	0.00	0.00	
S4 -> D2	25.00	6.00	150.00	29.00	<==
S5 -> D3	22.00	5.00	110.00	24.00	
S6 -> D1	34.00	5.00	170.00	15.00	
S6 -> D6	3.00	0.00	0.00	0.00	
S7 -> D5	21.00	5.00	105.00	24.00	
		Toplam Maliyet :	829.00		

2. Cözüm (5)

Dagıtım	Yapılacak Tasıma	Birim Maliyet	Toplam Maliyet	Zaman Degerleri	
S1 -> D4	14.00	5.00	70.00	8.00	
S1 -> D5	7.00	5.00	35.00	13.00	
S2 -> D2	21.00	7.00	147.00	12.00	
S2 -> D4	15.00	6.00	90.00	12.00	
S3 -> D2	13.00	8.00	104.00	12.00	
S3 -> D6	23.00	0.00	0.00	0.00	
S4 -> D3	6.00	6.00	36.00	11.00	
S4 -> D6	19.00	0.00	0.00	0.00	
S5 -> D3	22.00	5.00	110.00	24.00	<==
S6 -> D1	34.00	5.00	170.00	15.00	
S6 -> D6	3.00	0.00	0.00	0.00	
S7 -> D5	21.00	5.00	105.00	24.00	<==
		Toplam Maliyet :	867.00		

3. Cözüm (7)

Dagıtım	Yapılacak Tasıma	Birim Maliyet	Toplam Maliyet	Zaman Degerleri	
S1 -> D5	21.00	5.00	105.00	13.00	
S2 -> D3	10.00	6.00	60.00	20.00	
S2 -> D4	26.00	6.00	156.00	12.00	
S3 -> D2	13.00	8.00	104.00	12.00	
S3 -> D4	3.00	7.00	21.00	22.00	<==
S3 -> D6	20.00	0.00	0.00	0.00	
S4 -> D3	18.00	6.00	108.00	11.00	
S4 -> D5	7.00	8.00	56.00	9.00	
S5 -> D6	22.00	0.00	0.00	0.00	
S6 -> D1	34.00	5.00	170.00	15.00	
S6 -> D6	3.00	0.00	0.00	0.00	
S7 -> D2	21.00	6.00	126.00	19.00	
		Toplam Maliyet :	906.00		

4. Cözüm (3)

Dagıtım	Yapılacak Tasıma	Birim Maliyet	Toplam Maliyet	Zaman Degerleri	
S1 -> D5	21.00	5.00	105.00	13.00	
S2 -> D3	7.00	6.00	42.00	20.00	<==
S2 -> D4	29.00	6.00	174.00	12.00	
S3 -> D2	13.00	8.00	104.00	12.00	
S3 -> D5	3.00	9.00	27.00	6.00	
S3 -> D6	20.00	0.00	0.00	0.00	
S4 -> D3	21.00	6.00	126.00	11.00	
S4 -> D5	4.00	8.00	32.00	9.00	
S5 -> D6	22.00	0.00	0.00	0.00	
S6 -> D1	34.00	5.00	170.00	15.00	
S6 -> D6	3.00	0.00	0.00	0.00	
S7 -> D2	21.00	6.00	126.00	19.00	
		Toplam Maliyet :	906.00		

5. Cözüm (4)

Dagıtım	Yapılacak Tasıma	Birim Maliyet	Toplam Maliyet	Zaman Degerleri	
S1 -> D5	21.00	5.00	105.00	13.00	
S2 -> D2	7.00	7.00	49.00	12.00	
S2 -> D4	29.00	6.00	174.00	12.00	
S3 -> D2	6.00	8.00	48.00	12.00	
S3 -> D3	3.00	8.00	24.00	15.00	
S3 -> D5	7.00	9.00	63.00	6.00	
S3 -> D6	20.00	0.00	0.00	0.00	
S4 -> D3	25.00	6.00	150.00	11.00	
S5 -> D6	22.00	0.00	0.00	0.00	
S6 -> D1	34.00	5.00	170.00	15.00	
S6 -> D6	3.00	0.00	0.00	0.00	
S7 -> D2	21.00	6.00	126.00	19.00	<==
		Toplam Maliyet :	909.00		

6. Cözüm (3)

Dagıtım	Yapılacak Tasıma	Birim Maliyet	Toplam Maliyet	Zaman Degerleri	
S1 -> D5	21.00	5.00	105.00	13.00	
S2 -> D2	28.00	7.00	196.00	12.00	
S2 -> D4	8.00	6.00	48.00	12.00	
S3 -> D2	6.00	8.00	48.00	12.00	
S3 -> D3	3.00	8.00	24.00	15.00	<==
S3 -> D5	7.00	9.00	63.00	6.00	
S3 -> D6	20.00	0.00	0.00	0.00	
S4 -> D3	25.00	6.00	150.00	11.00	
S5 -> D6	22.00	0.00	0.00	0.00	
S6 -> D1	34.00	5.00	170.00	15.00	<==
S6 -> D6	3.00	0.00	0.00	0.00	
S7 -> D4	21.00	5.00	105.00	12.00	
		Toplam Maliyet :	909.00		

7. Çözüm (5)

Dagıtım	Yapılacak Tasıma	Birim Maliyet	Toplam Maliyet	Zaman Degerleri	
S1 -> D1	12.00	5.00	60.00	9.00	
S1 -> D3	3.00	6.00	18.00	11.00	
S1 -> D5	6.00	5.00	30.00	13.00	<==
S2 -> D2	28.00	7.00	196.00	12.00	
S2 -> D4	8.00	6.00	48.00	12.00	
S3 -> D2	6.00	8.00	48.00	12.00	
S3 -> D5	22.00	9.00	198.00	6.00	
S3 -> D6	8.00	0.00	0.00	0.00	
S4 -> D3	25.00	6.00	150.00	11.00	
S5 -> D1	22.00	8.00	176.00	8.00	
S6 -> D6	37.00	0.00	0.00	0.00	
S7 -> D4	21.00	5.00	105.00	12.00	

Toplam Maliyet : 1029.00

8. Çözüm (3)

Dagıtım	Yapılacak Tasıma	Birim Maliyet	Toplam Maliyet	Zaman Degerleri	
S1 -> D1	18.00	5.00	90.00	9.00	
S1 -> D3	3.00	6.00	18.00	11.00	
S2 -> D2	28.00	7.00	196.00	12.00	<==
S2 -> D4	8.00	6.00	48.00	12.00	<==
S3 -> D2	6.00	8.00	48.00	12.00	<==
S3 -> D5	28.00	9.00	252.00	6.00	
S3 -> D6	2.00	0.00	0.00	0.00	
S4 -> D3	25.00	6.00	150.00	11.00	
S5 -> D1	16.00	8.00	128.00	8.00	
S5 -> D6	6.00	0.00	0.00	0.00	
S6 -> D6	37.00	0.00	0.00	0.00	
S7 -> D4	21.00	5.00	105.00	12.00	<==

Toplam Maliyet : 1035.00

Sıralı Çözümler:

Çözüm No	Toplam Maliyet	Enbüyük Zaman
1	829.00	29.00
2	867.00	24.00
3	906.00	22.00
4	906.00	20.00
5	909.00	19.00
6	909.00	15.00
7	1029.00	13.00
8	1035.00	12.00

Üstün Çözümler:

Çözüm No	Toplam Maliyet	Enbüyük Zaman
1	829.00	29.00
2	867.00	24.00
3	906.00	20.00
4	909.00	15.00
5	1029.00	13.00
6	1035.00	12.00

Önerilen Çözüm No : 4 (909.00, 15.00)