HAVA HAREKÂT GÖREVLERİNDE FİLOLARIN HEDEFLERE ATANMASININ MODELLENMESİ ve GERÇEKLEŞTİRİMİ

DESIGN and IMPLEMENTATION of an ASSET – TARGET ALLOCATION SYSTEM for AIR TASKING ORDERS

ŞÜKRÜ TİKVEŞ

Hacettepe Üniversitesi

Lisanüstü Eğitim — Öğretim ve Sınav Yönetmeliğinin

BİLGİSAYAR MÜHENDİSLİĞİ Anabilim Dalı İçin Öngördüğü

YÜKSEK LİSANS TEZİ

olarak hazırlanmıştır

2007

Fen Bilimler Entistüsü Müdürlüğü'ne,

Bu çalışma jürimiz tarafından **BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI'-nda YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Başkan _____
Prof. Dr. A. Ünal Yarımağan

Üye (Danışman) _____
Yrd. Doç. Dr. Kayhan İmre

Üye (Eş Danışman) _____
Prof. Dr. Faruk Polat

Üye _____
Yrd. Doç. Dr. Mustafa Ege

Üye _____
Doç. Dr. Göktürk Üçoluk

Bu tez . . . / . . . / . . . tarihinde Enstitü Yönetim Kurulunca kabul edilmiştir.

Prof. Dr. Erdem Yazgan

Enstitü Müdürü

HAVA HAREKÂT GÖREVLERİNDE FİLOLARIN HEDEFLERE ATANMASININ MODELLENMESİ ve GERÇEKLEŞTİRİMİ

ŞÜKRÜ TİKVEŞ

ÖZ

Bu çalışmada, hava operasyonları için askeri kaynak planma problemi ve probleme yönelik üretilmiş çözüm yöntemleri araştırılmıştır. Çeşitli problem büyüklük ve karmaşıklıkları için bu yöntemlerin uygulanabilirlik ve başarımları incelemiş, ayrıca gerçek zaman kısıtlarının bulunması durumunda bu yöntemlerden ne şekilde faydalanabileceği anlatılmıştır. Çalışmanın katkısı olarak, bu tür problemlerin çözümü için bir çatı hazırlanmış ve çeşitli durumlar için uygun olan yöntem çözümlemeleri sunulmuştur.

Askeri kaynak planlama problemleri, silahlı kuvvetlerin bulundurduğu kaynakları, gerçekleştirmesi gereken görevlere en verimli olarak ataması ile ilgilidir. Tanım olarak, eniyileme problemleri içinde, kombinasyonel optimizasyon dalına ve karmaşıklık olarak $NP - Complete$ kümesine dahildirler. Ayrıca, düşman savunma birimlerinin bulunması, görevlerin birbirleriyle ilişkilerinin kapsama eklenmesi, vb şartlara göre çeşitli şekillde değişebilen problem formülasyonları da bulunmaktadır.

Geliştirilen olası çözümlerin kıyaslanabilmesi için, yöntemlerin tümüne ortak bir problem tanımı oluşturulmuş ve tüm çözücülerin çalıştırılabileceği yazılım çatısı hazırlanmıştır. Yapılan deneyler sonucunda ise, görev sayısı olarak büyük ve karmaşıklığı yüksek girdiler için genetik algoritma kullanmanın, diğer hallerde ise dallan ve sınırla ile tam arama yapmanın uygun olduğu görülmüştür.

Anahtar Kelimeler: Askeri kaynak planlama, kombinasyonel optimizasyon, genetik algoritma, tam arama

Danışman: Yrd. Doç. Dr. Kayhan İmre, Hacettepe Üniversitesi, Bilgisayar Mühendisliği Bölümü

Eş-danışman: Prof. Dr. Faruk Polat, Orta Doğu Teknik Üniversitesi, Bilgisayar Mühendisliği Bölümü

DESIGN and IMPLEMENTATION of an ASSET – TARGET ALLOCATION SYSTEM for AIR TASKING ORDERS

ŞÜKRÜ TİKVEŞ

ABSTRACT

This thesis considers the resource allocation problem and proposed solutions for air force military operations. The applicability of various methods depends on the size of the problem and the complexity (i.e.: resource availability) of the input configuration. In thıs thesıs, a framework has been desıgned to analyse the methods in different conditions.

The military resource allocation problem tries the optimize to allocation of military assets to fulfill a given list of objectives as effectively and efficiently as possible. The formulation of a specific problem may be adopted to include enemy defences or inter-mission dependencies into account. Yet, all the studied forms of the problem are classified under combinatorial optimization branch of mathematical optimization problems.

The problem complexity has been analysed, and shown to be in $NP - Complete$ set. Also the suitability of solvers for real time constraints have been discussed.

In order to compare different methods, a subset of the problem has been defined and a framework with the implementation of specific solvers has been prepared. The analysis of solver peformance show that genetic algorithms are appropriate for large and complex problem instances. Exhaustive search with branch-and-bound method is sufficient for the other instances of the problem.

Keywords: Military resource planning, asset allocation, combinatorial optimization, genetic algorithm, exhaustive search

Advisor: Assoc. Prof. Dr. Kayhan İmre, Hacettepe University, Department of Computer Engineering

Co-advisor: Prof. Dr. Faruk Polat, Middle East Technical University, Department of Computer Engineering

TEŞEKKÜR

Yazar, yüksek lisans çalışmasını gerçekleştirken gösterdikleri katkılardan dolayı, aşağıdaki kişilere teşekkür etmeyi borç bilmektedir:

ERSİN ER, yazarın dertlerini dinleyip, sıkılmadan yardımcı olduğu ve tez metnini sayılmayacak kadar çok kez gözden geçirdiği için;

ÖNER ÜNAL, kendi yoğun zamanlarında bile arkadaşlarına vakit ayırdığı ve Hacettepe Üniversitesi Bilgisayar Mühendisliği Bölümü'nü daha canlı hale getirdiği için;

KEREM ERZURUMLU, heyecanı ve yazarın tez haricindeki birikiminin sağlanmasına gösterdiği önemli katkısından dolayı.

İÇİNDEKİLER DİZİNİ

Sayfa

ÖZ i

ABSTRACT ii

TEŞEKKÜR iii

ŞEKİLLER DİZİNİ vi

ÇİZELGELER DİZİNİ vii

1 GİRİŞ 1
1.1 Hava Operasyonları Askeri Kaynak Planlama Problemi . . . 2
1.1.1 Genel problemler . . . 3
1.1.2 Düşman savunma sistemlerinin bastırılması . . . 4
1.1.3 Girdi varsayımları . . . 4
1.2 Daha Önce Yapılmış Çalışmalar . . . 5
1.3 Kullanılan Problem Tanımı ve Girdi Biçimi . . . 6
1.4 Kullanılan Çözüm Biçimi . . . 8
1.5 Çözüm Yöntemi Beklentileri . . . 9
1.5.1 Tamsayı çözüm üretebilme . . . 9
1.5.2 Verimlilik (*efficiency*) . . . 9
1.5.3 Etkililik (*effectiveness*) . . . 9
1.5.4 Problemin modellenmesi . . . 10
1.5.5 Tekrar kullanılabilirlik . . . 10
1.6 Çözüm Yöntemlerinin Tanıtımı . . . 10
1.6.1 Tam arama . . . 10
1.6.2 Aç gözlü algoritmalar . . . 12
1.6.3 Genetik algoritmalar . . . 13
1.6.4 Ağ Akışı . . . 13
1.7 Deney Yöntemi . . . 14
1.8 Girdi Üretimi . . . 14
1.8.1 Girdi üretimi kıstasları . . . 15
1.8.2 Tam arama girdi kümesi . . . 15
1.8.3 Genel girdi kümesi . . . 15

2 KAPSAMLI ALGORİTMA TANITIMI 16
2.1 Aç Gözlü Çözücü Algoritması . . . 16
2.2 Tam Arama . . . 16
2.3 Genetik Algoritma . . . 17
2.3.1 Temel genetik algoritma tanımı . . . 17
2.3.2 Özel genom türleri . . . 19
2.3.3 Özel algoritma türleri . . . 19
2.3.4 Kullanılan genom yapısı . . . 20
2.3.5 Kullanılan uygunluk değerlendirme işlevi . . . 21

2.4 Ağ Akış Problemleri 21
2.4.1 Genel yapı 22
2.4.2 En fazla akım 22
2.4.3 En düşük maliyetli akım 26

3 DENEY SONUÇLARI ve DEĞERLENDİRİLMESİ 29
3.1 Tam Arama Sonuçları 29
3.2 Etkililik Karşılaştırması 31
3.3 Yüksek Kapasite Oranlı Girdilerin İncelenmesi 35
3.4 Genel Karşılaştırma 35
3.5 Genetik Algoritma Parametrelerinin İncelenmesi 36
3.6 Aç Gözlü Algoritmaların Karşılaştırılması 39
3.7 Değerlendirme 39

4 Gerçek Zaman Kısıtları 42

5 Açık Konular 43
5.1 Gerçek hayata uygun girdiler 43
5.2 Eksik yöntemlerin tamamlanması 43
5.3 Yöntemlerin bir arada kullanılması 43
5.4 Tahmin karmaşıklığının incelenmesi 44
5.5 Çok amaçlı arama 44

6 SONUÇLAR ve TARTIŞMA 46

KAYNAKLAR 47

EKLER 49

A PROBLEM KARMAŞIKLIĞININ ÇÖZÜMLENMESİ 50
A.1 Problemin Karar Verme Problemi Olarak Tanımlanması 50
A.2 Polinomsal Zamanlı İndirgeme 51

B DENEY YAZILIMI KULLANIM BİLGİLERİ 53
B.1 Projeler 53
B.2 Rastgele Girdi Üretme Parametreleri 53
B.3 Girdiler 55
B.4 Algoritma Ayarları 56
B.5 Algoritma İşletme 56
B.6 Sonuçların Çıkarılması 57

C TERİMLER SÖZLÜĞÜ 58
C.1 İngilizce — Türkçe 59
C.2 Türkçe — İngilizce 60

ŞEKİLLER DİZİNİ

Sayfa

1.1 Problem veri yapısı 7
1.2 Örnek problem olgusu 8

2.1 Aç gözlü çözücü algoritması 17
2.2 Tam arama algoritması 18
2.3 Genom değerlendirme işlevi 21
2.4 Örnek en kısa uzunluklu yol problem olgusu 22
2.5 Örnek en fazla akım problem ağı 23
2.6 Örnek artık ağ 24
2.7 Farklı arttıran yol seçimları sonucunda oluşan akımlar 24
2.8 Genel arttıran yol en fazla ağ akışı çözüm algoritması 24
2.9 Ön akım - ittirme en fazla ağ akışı çözüm algoritması 27
2.10 Örnek en düşük maliyetli akım problemi. 28
2.11 Eksi maliyetli çevrimlerin kaldırılması en düşük maliyetli akım algoritması 28

3.1 Tam arama zaman kullanım grafiği (girdi kümesi 1) 30
3.2 Tam arama zaman kullanım grafiği (logaritmik ölçekli, girdi kümesi 1) . . . 31
3.3 Girdi büyüklüğüne göre dallan ve sınırla yöntemi zaman kullanım grafiği (girdi kümesi 2) 32
3.4 Kapasite oranına göre dallan ve sınırla yöntemi zaman kullanım grafiği (girdi kümesi 2) 33
3.5 Kapasite oranına göre yöntemlerin verimlilik oranları (girdi kümesi 2) . . . 33
3.6 Kapasite oranına göre yöntemlerin verimlilik oranları (girdi kümesi 2, 3 ve 4) 34
3.7 Kapasite oranına göre yöntemlerin etkinlik oranları (girdi kümesi 2, 3 ve 4) 34
3.8 Aç gözlü arama ve ağ akışı zaman kullanım grafiği (girdi kümesi 4) 37
3.9 Genetik algoritma zaman kullanım grafiği (girdi kümesi 4) 37
3.10 Farklı genetik algoritma türleri için etkililik grafiği (girdi kümesi 4) 39
3.11 *SteadyState* algoritmasının etkillik grafiği (girdi kümesi 4) 40
3.12 Farklı genetik algoritmalar için zaman kullanım grafiği (girdi kümesi 4) . . 40
3.13 Farklı sıralama türlerine göre aç gözlü algoritma etkililiği (girdi kümesi 2, 3 ve 4) 41

B.1 Proje görünümü 54
B.2 Görev türü tanımı 54
B.3 Girdi üretme parametresi oluşturma 55
B.4 Girdi nesneleri 55
B.5 Algoritma ayarları 56
B.6 Tek bir algoritmanın ayar ayrıntıları 56
B.7 Tüm algoritmaların çalıştırılması ve sonuçların elde edilmesi 57
B.8 Çözücü sonuçlarının öğrenilmesi 57

ÇİZELGELER DİZİNİ

Sayfa

3.1 Tam arama karşılaştırmalı deney sonuçları (girdi kümesi 1) 30
3.2 Tam arama karşılaştırmalı deney sonuçları (girdi kümesi 2) 32
3.3 Yüksek kapasite oranlı girdiler deney sonuçları (girdi kümesi 3) 35
3.4 Düşük kapasite oranlı girdiler deney sonuçları (girdi kümesi 4) 35
3.5 Genetik algoritma parametreleri karşılaştırması (girdi kümesi 4) 38

1 GİRİŞ

Hava operasyonları askeri planlama problemleri, operasyonların planlanmasının farklı aşamalarında karşılaşılan çeşitli alt problemleri içermektedir. Bu problemler temel olarak olası hedeflerin ve düşman savunma sistemlerinin tespiti (istihbarat), olası hedeflere karşı görevlerin belirlenmesi, eldeki kaynakların bu görevlere verimli bir şekilde atanması (kaynak planlama), bu görevlerin birbirleri ile ilişkileri gözönüne alınarak sıralanması başlıklarında toplanabilir.

Kaynak planlama, eldeki kaynakların en verimli atanması olarak yöneylem araştırması (*OR, operations research*) kapsamında da tanımlanmaktadır[HL05]. Yöneylem araştırması disiplinler arası bir çalışma sahasıdır. Bu bilim dalı ikinci dünya savaşı sırasında silahlı kuvvetlerin kaynaklarının verimli kullanılması amacıyla matematiksel yöntemlerin değerlendirilmesiyle başlamıştır[HL05]. Matematiksel eniyileştirme ve kaynak planlama ile ilgilendiği göz önüne alındığında, problemlerinin hem bilgisayar bilimcilerinin, hem de endüstüri mühendisilerinin alanlarına girmesi doğal olmaktadır.

Hava kuvvetlerin'n askeri operasyonları için kaynak planlama problemlerinin formülasyonu genel olarak tamsayı doğrusal programlama (*integer linear programming*) şeklinde yapılmaktadır[LCB04]. Tamsayı doğrusal programlama çözüm yöntemleri (*simplex* ve dallan ve sınırla ikilisi) ise en kötü durumda üssel karmaşıklıktadırlar. Bu şekilde, çoğu zaman tam çözüm makul sürede bulunabilse dahi, etkililikteki – doğru sonucu bulabilmedeki – belirsizlik ve çalışma zamanlarının üssel karmaşıklıkta olmasından dolayı bilgisayar bilimlerinde genel olarak tercih edilmemektedir.

Probleme, genel sezgisel yöntemlerle yaklaşmak veya problemi ağ akışı veya taşıma problemi olarak modellemek daha iyi bir üst sınırda karmaşıklığa ulaşmamıza yardımcı olacaktır.

Yukarıda sayılan yaklaşım ve yöntemler, kombinasyonel optimizasyon adı verilen eniyileme problem kümesine dahildir. Kombinasyonel optimizasyon, $NP - Hard$ olan eniyileme problemleri ile ilgilidir.

Bu çalışma, normalde farklı formülasyonları olan yöntemleri, bir arada, aynı problem kümesi üzerinde kullanılabilir halde geliştirmeyi amaçlamaktadır. Bunun sonucunda da, probleme yalnızca tek bir yöntem ile yaklaşmak yerine, birden çok

yöntemi bir çatı altında toplayıp, çoklu yöntemle yaklaşmanın daha uygun olacağı gösterilemeye çalışılacaktır.

İlgili yöntemlerin tanımı ve probleme uyarlanması ise bu bölümün alt kesimlerinde ve Bölüm 2 içerisinde açıklanmıştır.

1.1 Hava Operasyonları Askeri Kaynak Planlama Problemi

Hava operasyonlarında kaynakların hedeflere atanıp görevlerin tanımlanması; görevlerin arasındaki ilişkilerin de göz önüne alınıp zamana göre planlanması ve bir kısım görevlerin başarılı veya başarısız olarak sonuçlanmasına göre planların değiştirilmesi halen etkin olarak üzerinde çalışılan problemlerdir[ATZ04].

Bu çalışma kapsamında temel olarak saldırı kaynaklarının hedeflere atanması problemi üzerinde durulmuştur. Bu problem, Li ve diğerleri tarafından "saldırı kaynaklarının (*asset*) paketler halinde gruplanıp, bu grupların hedeflere ve savunma kaynaklarına güç potansiyelini verimli şekilde kullanarak atanması" olarak tanımlanmıştır[LCB04].

Hava kuvvetlerinin ülke genelinde üslere dağılmış olarak çeşitli kaynakları bulunmaktadır. Bu kaynaklardan ilgi alanımıza girenler şunlardır:

- **Filolar**

  Her bir üs içerisindeki temel muharip birim filo olarak adlandırılmakdır. Filolar üslerde konuşlanan uçaklar ve bunlara bağlı pilotlardan oluşmaktadır.

  Uçakların kabiliyeti taşıyabildikleri yük miktarı ve normal uçuş menzillerine göre değişmektedir.

- **Pilotlar**

  Temel olarak hava kuvvetlerinin görevleri hava-hava ve hava-kara olarak ikiye ayrılabilir[1]. Bunları gerçekleştirecek pilotların eğitimi ise görev türlerine göre farklılık göstermektedir. Sonuç olarak, farklı türdeki hedefler için pilot seçimi başarı şansını doğrudan etkilemektedir.

- **Mühimmat**

[1] TSK bünyesinde hava-deniz görevleri de bulunmaktadır. Fakat bu çalışma, hava-kara ve bunuları destekleyen hava-hava görevleri ile ilgilenmektedir.

Uçakların temel görevi, üslerinden hedeflerine bir miktar patlayıcı mühimmat taşımaktır. Mühimmatın tahrib kabiliyeti kütleleriyle orantılıdır.

İstenen tahrip gücünü sağlamak için gereken uçak miktarı, uçak türünün taşıma kapasitesi ve uzun mesafede silah yerine kısmen yedek yakıt tankı kullanımına göre belirlenmektedir.

- **Diğer kaynaklar**

  Görevlerin gerçekleştirilmesinde bu temellerin dışında da göreve özel bir kısım ek kaynakların (örneğin SEAD[2] desteği) bulundurulması da gerekebilmektedir.

1.1.1 Genel problemler

- **İstihbarat ve hedef belirleme**

  Hava kuvvetlerinin ilgili birimleri daimi olarak düşman kaynaklarını takip ederek, siyasi iradenin isteyebileceği çeşitli amaçlara yönelik hedef belirleme faaliyetinde bulunur.

- **Paket oluşturma**

  Bir görevi tamamlamak için kullanılacak kaynakların toplamı paket olarak adlandırılır. Paket tanımı içerisinde filolardan seçilen uçaklar, mühimmat ve ek kaynaklar bulunmaktadır.

- **Makul paketlerin seçimi**

  Normal şartlarda eldeki kaynakların belirlenen tüm hedefleri karşılaması mümkün olmamaktadır. Bu sebeple hedeflerin önceliklendirilmesi ve aralarındaki ilişkilerin belirlenmesi; bunun sonucunda ise makul bir hedef alt kümesinin operasyonlar için seçilmesi gerekmektedir.

Eğer bu tür sistemlerin tüm alt problemleri çözülmek istenirse, farklı türlerde çalışmalar yapmak gerekecektir. Örneğin burada verilen kapsamın tümünün gerçekleştirilmesi için kısıt tatmini (*constraint satisfaction*), kombinasyonel optimizasyon, ve yapay zekâ planlama (AI planning) yöntemlerinin kullanılması olasıdır.

[2] *SEAD*: Supression of enemy air defences

1.1.2 Düşman savunma sistemlerinin bastırılması

Görev planlamada, hedefleri tek başına düşünmek yeterli değildir. Hedefi tahrip edecek gücün yanında, düşmanın hava savunma sistemlerini etkisiz hale getirecek desteğin bulundurulması da gerekmektedir. Bunlar, hava devriye (*interceptor*) birimlerine karşı, hava-hava savaşma kabiliyetli filolar veya yerden havaya savunma sistemleri için, SEAD uçakları olabilir.

1.1.3 Girdi varsayımları

Bu çalışma kapsamında birden çok çözüm yönteminde aynı girdi kümesinin kullanılabilmesi için çeşitli varsayım ve sadeleştirmelerde bulunulmuştur.

Öncelikle bir filo içinde tek tür uçak ve kabiliyeti birbirine yakın pilotlar bulunduğu varsayılmıştır. Bütün bir filo kapsamında uçaklarına ve pilotlarına bakılarak, her farklı görev türü için birer kabiliyet değeri saklanmaktadır. Bu varsayım, farklı türde pilotlar içeren filolar mantıksal alt filolara bölünerek gerçek dünyaya uyarlanabilir.

Aynı zamanda hedef noktasında tek bir görev bulunduğu kabul edilmiştir. Birden çok göreve konu olan hedefler, aynı noktadaki sanal hedeflerle karşılanabilir. Bir hedefin türüne göre filoların sağladığı kabiliyet değiştiği için bu kabul modeli daha sade hale getirmektedir.

Görevler arasındaki bağımlılık ve zaman ilişkileri modellenmemiştir. Problem dahilinde yalnızca kaynak atama sorunu çözülmeye çalışılmıştır. Ayrıca problem tek bir sorti ve her göreve tek bir filo atanacak şekilde düşünülmektedir. Birden çok sorti için birleşik bir model (bir filonun birden çok kopyasının bulunması) veya çok seviyeli ayrı problem olguları kullanılarak yakınsama yapılabilir. Fakat ne yazık ki bu modeller denk değildir.

Çözücü algoritmalar, problem karmaşıklığını değiştirmediği için, yalnızca filo içindeki uçakları kaynak olarak kullanmaktadır. (Çözücüler tarafından gözardı edilse bile problem girdisi tanımında diğer tür kaynakların belirtilmesi olasıdır).

Bu varsayım ve kısıtılamalar, problem kapsamını fazla daraltmadan, gerçekleştirilmesi gereken algoritmaların önemli ölçüde sadeleştirilmesini ve kod tekrarının azaltılmasını, dolayısıyla çalışmanın sınanmasının kolaylaştırılmasını sağlamaktadır.

1.2 Daha Önce Yapılmış Çalışmalar

Griggs ve diğerleri, problemi hava kuvvetlerindeki farklı yaklaşım yöntemleri ile beraber açıklayıp, tamsayı karışık programlama (*mixed integer programming*) biçiminde modellemişlerdir [SJ97]. Li ve diğerleri ise, problemi 0/1 sırt çantası problemi (*0/1 knapsack problem*) şeklinde sadeleştirip, yine tamsayı karışık programalama ile çözerek, daha büyük girdilerde, en iyi çözüme $\%1$ yakınlıkta ulaşılabileceğini göstermiştir.

Ahuja ve diğerleri, benzer bir askeri kaynak planlama problemi olan silah hedef ataması (*WTA - weapon target allocation*) ile ilgilenmiştir. Bu çalışmada ilgilenilen problemden farklı olarak, silah hedef ataması, savunma kuvvetlerinin taarruzdan korunma için, düşman saldırı kuvvetlerine atanması ile ilgilenmektedir. Ahuja ve diğerlerinin çalışmasının önemli bir farkı da, kazancı en fazlalaştırmak yerine, riski en az hale getirmeye çalışılmasıdır [AKJO03].

Önerdikleri çözüm, önce bir arama alt sınırı bulup, dallan ve sınırla (*branch and bound*) yöntemi ile bir geniş komşuluklu arama uzayının taranmasıdır. Arama alt sınırı olarak, doğrusal programlama, tamsayı karışık programlama, en düşük maliyetli akım ve en yüksek getiri tabanlı aç gözlü yöntem kullanmışlardır. Sınırlama yöntemi olarak ise, yine tamsayı karışık programlama, en düşük maliyetli alım, en yüksek getiri yöntemlerinden ve ayrıca özel olarak tanımladıkları melez bir yöntemden faydalanmışlardır. Çalışma sonucunda ise büyük girdi kümeleri için dahi problemin gerçek zamanda çözülebileceğini öne sürmüşlerdir[AKJO03].

Metler ve Preston ise aynı problemi savunma birimlerin kurtulma şansının en fazlalaştırılması olarak modellemiştir. Ayrıca taarruz kuvvetlerinden bir grubun tümüyle engellenememesi durumunda başarılı olacaklarını, yani hedeflerde başarının ya tam ya da sıfır şeklinde olduğunun kabulünü yapmışlardır. Bu modeli doğrusal progralama ve aç gözlü bir yöntemle çözmüşlerdir[Met89].

Lee ve diğerleri probleme soylulaştırma (*eugenics*) destekli genetik algoritma ile yaklaşmıştır. Genetik algoritma ve *simulated annealing* yöntemlerini teker teker ve bir arada gerçekleştirmiş, ayrıca genetik algoritmanın aç gözlü bir soylulaştırma yöntemi ile desteklenmiş halini eklemişlerdir. Çalışmaları sonucunda genetik algoritmanın, problem bilgisi dahilindeki soylulaştırma ile geliştirilmesiyle en iyi sonucun alınabileceğini göstermişlerdir[LSL03].

Yine Lee ve diğerleri, genetik algoritma ve karınca kolonisi eniyileştirme yöntemlerini karşılaştırmış ve karınca kolonisi eniyileştirmesinin, silah hedef ataması problemi için daha hızlı sonuca ulaştığını bulmuşlardır[LSL02].

Aberdeen ve diğerleri tümüyle farklı olarak, görev sırası planlaması problemi içinde kaynak ihtiyaçlarını da dahil ederek, zaman ve görev bağımlılıklarını da gözeterek, bir model önermişlerdir. Önerdikleri modeli gerçek zamanlı LRTDP (*labeled real time dynamic programming*) algoritması ile çözmüşlerdir[ATZ04].

1.3 Kullanılan Problem Tanımı ve Girdi Biçimi

Problem temel olarak, verilen olası bir görev listesi içinden, eldeki kaynaklara göre en yüksek verimi elde edecek bir alt küme bulmak şeklinde çözülmektedir. En yüksek verim, karşılanan hedeflerin getirilerinin toplamı olarak tanımlanmaktadır.

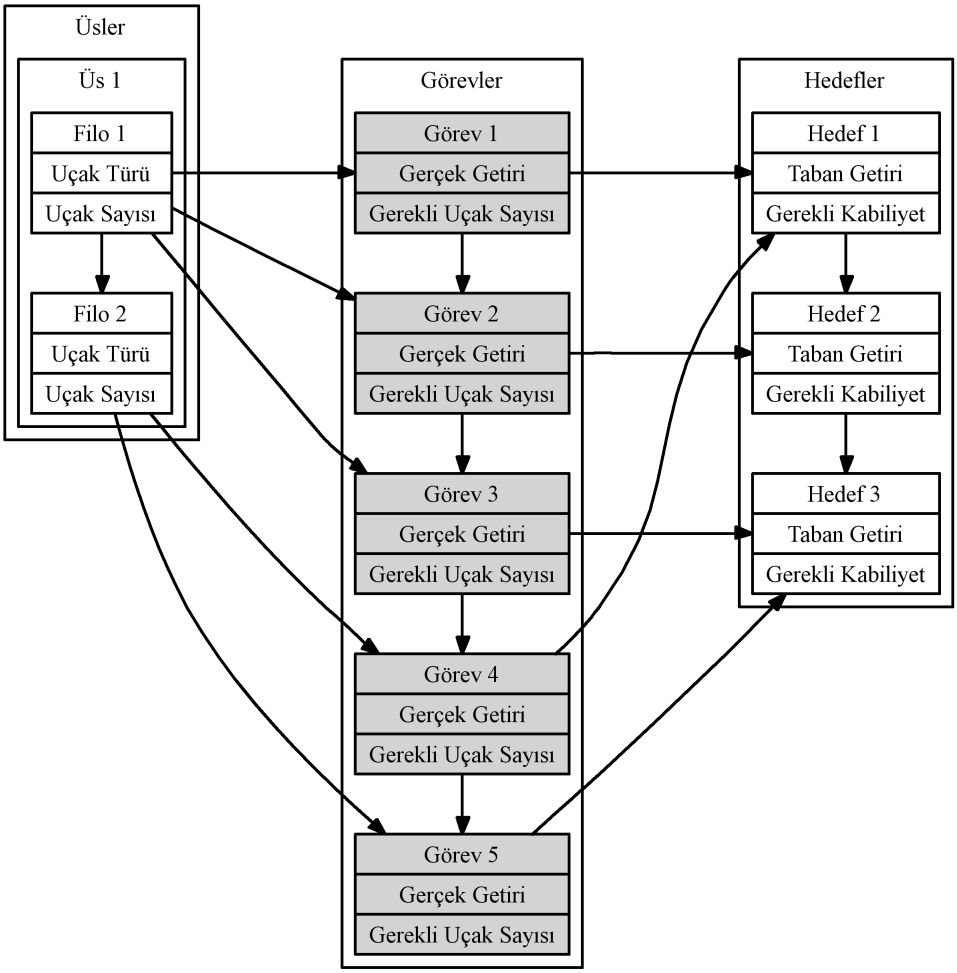
Problemde amaç, hedeflerin gerçekleştirilmesidir. Hedeflerin konumları, türleri ve taban getiri değeri bulunmaktadır. Bu bilgilere göre her bir görev içerisinde bir net getiri değeri ve risk hesaplanmaktadır. Bir hedef için tek bir görev seçilmesi yeterlidir, çünkü birden çok görev atanması toplam getiri değerini yükseltmeyecektir.

Hedeflerin ayrıca kaynak ihtiyacı da bulunmaktadır. Bu kaynaklar filo kabiliyeti ve diğer kaynaklar olarak belirlenip, girdi bilgisinde verilen filo tanımları ile karşılanmaktadır. Hedefler için üretilen görev tanımlarında filo kabiliyetleri, her bir filo için, görevi karşılamasına yeterli net uçak sayısına dönüştürülmektedir. Filolardaki uçak sayısının da sınırlı olduğu göz önüne alındığında, her bir filonun aynı anda gerçekleştirebileceği görev sınırı belirlenmektedir.

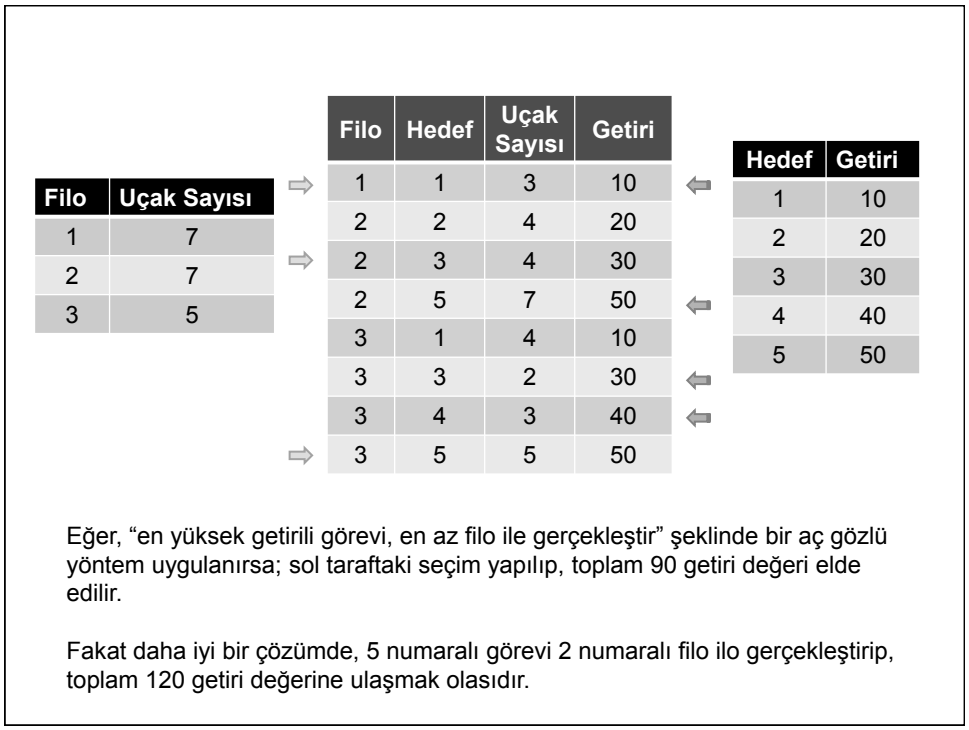
Diğer kaynaklar filoların ait olduğu üsler içinde tanımlanmıştır. Fakat tüm çözücü algoritmalara rahatca uyarlanamadıkları için çıktı içerisinde bu bilgiler göz ardı edilmektedir. Ayrıca risk değeri de kullanmamaktadır. Bu değerin şimdiki veya daha gelişmiş bir modele nasıl uyarlanabileceği Bölüm 5 içerisinde anlatılmaktadır.

Genel problem girdisi yapısını özetleyen bir çizim Şekil 1.1 içinde gösterilmiştir. Şekilde gösterildiği gibi, bir filo ve bir hedef arasında bağlantı kurulmasıya, görevler oluşturulmaktadır. Örnek bir problem olgusu ise Şekil 1.2 içinde verilmiştir.

Yukarıda anlatılan problemin, karar verme problemi karşılığı $NP - complete$ kümesine dahildir. Bu önermenin ispatı EK A içerisinde verilmiştir.



Şekil 1.1: Problem veri yapısı



Şekil 1.2: Örnek problem olgusu

1.4 Kullanılan Çözüm Biçimi

Tez kapsamındaki çözüm algoritmalarından, yukarıda verilen Filo, Hedef ve Görev tanımlarını içeren bir problem verildiğinde, toplam görev getirisini en yüksek yapacak bir alt küme belirlenmesi istenmektedir.

Çözücülerin çıktısı bu alt kümeyi tanımlayan bir bit dizisi şeklindedir. Dizi içindeki değerler ilgili görevin seçilip seçilmediğini göstermektedir.

Daha önceden de belirtildiği gibi, bir hedef tek bir sefer gerçekleştirilebileceği ve filoların uçak miktarı kısıtlı olduğu için; aynı hedefi gerçekleştirmek için birden çok görevi seçen veya bir filodan verebileceğinden fazla uçak talep eden çözümler geçersiz kabul edilmektedir (gerçekleştirilen yazılım bu tür geçersiz çözümleri otomatik olarak yakalayabilme yeteneğine sahiptir).

1.5 Çözüm Yöntemi Beklentileri

1.5.1 Tamsayı çözüm üretebilme

Gerçek hayatta bir göreve "yarım" uçak atanması gibi bir durum söz konusu olmamaktadır. Dolayısıyla her bir uçağın tam olarak bir göreve atanması veya atıl durması gerekmektedir. Bunun yanında görevlerin kısmen gerçekleştirilmesinin de herhangi bir fayda sağlamadığı kabul edildiğinden dolayı, örneğin 6 uçak isteyen bir göreve 5 uçak atanmasının herhangi bir anlamı bulunmamaktadır.

Fakat bu gereksinimden dolayı tam çözüm üretemeyen yöntemlerin tümüyle işe yaramadığı anlamı çıkarılmamalıdır. Çünkü yarım olan bir çözüm, başka bir algoritmanın girdisi olarak kullanılarak daha faydalı hale getirilebilir[AMO93].

1.5.2 Verimlilik (*efficiency*)

Çözüm algoritmalarının makul bir süre içerisinde ve kabul edilebilir bir bellek kullanımıyla çalışması beklenmektedir. Bu beklentinin tek istisnası, başvuru değerlerin üretilmesinde kullanılan tam arama yöntemidir. (Tam arama çıktısından deney sonuçlarının karşılaştırılmasında faydalanılmıştır. Fakat yöntem, tanımı gereği, tüm olası durumların üzerinden geçtiği için gerçek hayatta kullanılması öngörülmemektedir).

Buna ek olarak, gerçek zaman kısıtları bulunması durumunda, makul süre ihtiyacı belli bir kısıtlı zaman şekline gelmektedir. Bu tür kısıtlamalara, hangi algoritmaların ne şartlar altında uyabildiği Bölüm 4 içinde ayrıntılı olarak incelenmiştir.

1.5.3 Etkililik (*effectiveness*)

Algoritmaların hızlı çalışmasının yanında iyi sonuç üretmesi de beklenmektedir. Dolayısıyla yapılan tam aramaya göre ne kadar süre kazandırdıklarının yanında, en iyi çözümden hangi oranda taviz verdikleri gerçek hayatta çözüm yöntemi seçiminde tercih ölçütü olacaktır.

Bu amaçla, deney sonuçları içerisinde deneysel verilerden yola çıkılarak algoritmaların en iyi çözümden uzaklıkları modellenmeye çalışılmıştır.

1.5.4 Problemin modellenmesi

Algoritmanın elimizdeki probleme uyarlanabilmesi gerekmektedir. Yani her bir algoritmanın girdi içerisindeki verinin olabildiğince çok kısmını kullanması ve çıktı için tam olarak bir görev kümesi oluşturması istenmektedir (Tam görev kümesi oluşturamayanlar, yukarıda anlatıldığı üzere ayrı bir işlemden daha geçirilmelidir).

Girdiyi uyarlarken iki farklı yaklaşım söz konusudur. İlki, görevlerin birbirleriyle ilişkisi (her hedef için tek görev olması gibi) göz önünü alınmayıp doğrudan alt küme oluşturmaya çalışmaktır. İkincisi ise, probleme her bir hedef için birer görev seçimi, şeklinde yaklaşmaktır.

1.5.5 Tekrar kullanılabilirlik

Algoritma sonuçlarının ve mümkünse ara verilerinin, kısmen gerçekleşmiş veya başka bir sebepten dolayı değişmesi gereken bir problem tanımının düzeltilmesinde kullanılabilmesi ilgili durumlar için tercih sebebi olacaktır.

1.6 Çözüm Yöntemlerinin Tanıtımı

Aşağıdaki algoritmaların karmaşıklıkları tanımlanırken, F = filo sayısı , H = hedef sayısı , N = görev sayısı olarak kabul edilmiştir. Yapısı gereği $N = O(F \times H)$ olduğu hatırlanmalıdır.

1.6.1 Tam arama

Tam arama, olası tüm çözüm uzayının tüketilmesi şeklinde yapılan aramalardır. Tüm ihtimalleri denediği için her zaman en iyi sonucu bulmayı garantilemektedir. Fakat, bunun yanında kör bir arama yaptığı için gerçek hayatta kabul edilemeyecek zaman maliyetine sahiptir. Bu çalışmada ise, referans en iyi değerlerin bulunması amacıyla kullanılmaktadır.

Probleme eğer normal tanımı ile, yani ilişkiler gözetilmeksizin görev alt kümesi seçimi şeklinde, yaklaşırsak bu yöntem $O(2^N)$ karmaşıklığa sahiptir. Fakat bir hedef için en fazla tek bir görev şeçileceği bilindiğinden dolayı, hedefleri ayrı ayrı ele alarak $O(H^{N/H})$ veya $O(H^F)$ karmaşıklık elde edilebilir.

Ayrıca bir olası çözüm tam olarak üretildikten sonra, daha sonraki çözümlerin kısmen tamamlanmış hallerine bakarak, bunların tamamlanması durumunda eski bilinen en iyi çözümden daha iyi olma şansının olmadığını önceden kestirmek mümkündür. Bunun sonucu olarak, o ana kadarki en iyi çözüm değeri ve üzerinde çalışınan ara çözümün en iyi potansiyeli karşılaştırılarak sürekli olarak arama ağacını budamak olasıdır. Bu yönteme dallan ve sınırla (*branch-and-bound*) ismi verilmektedir.

Yapılan çalışmada bazı kesme sınır değeri belirleme yöntemleri de denenmiştir. Ayrıca sınama amacıyla bazı girdilerde hiçbir kesme olmadan da çalışılmıştır.

Aç gözlü sınır

Temel olarak daha sonradan gelecek tüm hedefler için en iyi görev seçeneğinin müsait olduğu varsayımı ile çalışır. Fakat bunun yanında asla gerçekleştirilemeyecek hedef bilgileri de eklenerek verimi arttırılabilir.

Örneğin kısmî çözümde yapılan harcamalardan sonra elde kalan en büyük filo içerisinde 10 uçak mevcutsa ve bir hedef, en az uçak isteyen görev seçeneğinde 11 uçak talep ediyorsa, bu hedefin kalan kaynaklarla hiçbir koşul altında gerçekleştirilmesi mümkün değildir. Bu bilgi kullanılarak, hedefin getirisinin sıfır olacağı önceden kesitirilebilir.

Sırt çantası sınırı (*knapsack bound*)

Sırt çantası sınırı, kalan problem kesiminin potansiyel değerini ölçmek için, açgözlü bir yöntem yerine, bu kesimin bir sırt çantası problemi olgusu olarak modelleme esasına dayanmaktadır.

Sırt çantası problemi, temel hali ile bir hırsızın sınırlı sığalı olan çuvalını önündeki değerli mallardan en verimli olarak nasıl dolduracağı sorununu çözmeye çalışır. Eğer sığa öğe büyüklüğüne tam bölünebilecek olsa en büyük getiri oranlı (getiri / büyüklük) malla doldurmak bariz çözümdür. Fakat atıl kalan sığa göz önüne alındığında, biraz daha az değerli fakat bu sözkonusu sığayı da dolduracak seçimler yapmak gerekmektedir.

Bunun yanında, eğer her bir maldan tek (veya kısıtlı sayıda) bulunuyorsa problem

'sınırlı sırt çantası problemi' (*bounded knapsack problem*) olarak adlandırılmaktadır-[Pis95].

Bu yöntem, aç gözlü yönteme göre daha ayrıntılı bir çözümleme yapsa da, iki sebepten dolayı zayıf kalabilmektedir. Öncelikle bu kesme algoritmasının kendi çalışma hızı, aç gözlü yönteme göre daha yavaştır. Ayrıca hedeflerin farklı filolarca gerçekleştirilmesi durumunda gereken uçak miktarı değiştiği için çuval büyüklüğünü uçak sayısı olarak vermek mümkün değildir. Bunun yerine filo kabiliyeti cinsinden verilmesi gerekmemektedir. Filo kabiliyeti rasyonel bir sayı olduğu için, talep edilen kabiliyetler aşağı ve sunulanlar yukarı doğru yuvarlanmak durumundadır, ki bu verimi azaltmaktadır.

Sıralama türleri

Eğer budama kullanılıyorsa, problemdeki görevlerin ele alınış sırası, budama verimini ve dolayısıyla zaman kullanımını etkilemektedir. Bunun sebebi daha iyi çözümlerin farklı bir sıralama ile erken bulunma olasılığının varlığıdır.

Gerçekleştirilen yazılım içinde üç farklı sıralama türü tanımlanmıştır. Bunlar en yüksek getirili görevlere göre, en yüksek getiri oranlı (getiri / filo gereksinimi) görevlere göre ve en az filo ihtiyacına göre sıralamaktır. Bu sıralama türlerinin birbirlerine göre avantaj ve dezavantajları Bölüm 3 içinde incelenmiştir.

1.6.2 Aç gözlü algoritmalar

Aç gözlü algoritmalar, tam aramadan faklı olarak, verdikleri kararları değiştirmemektedir. Bunun sonucunda da yalnızca tek bir çözüm üretirler.

Dolayısıyla, sıralama hariç $O(F + H)$, sıralama dahil $O(N \log N)$ zaman karmaşıklığında çalışıp, çok hızlı çözüm üretme imkânları vardır. Fakat, tercihlerin kalıcı olmasından dolayı, görev sıralaması algoritma tarafından bulunan çözümün etkliliğini doğrudan etkilemektedir.

Aç gözlü algoritma gerçekleştiriminin, yine tam aramada olduğu gibi, üç farklı sıralama yöntemi ile çalışma imkânı vardır. Bu sıralama yöntemleri, farklı girdi topolojileri için farklı başarım oranları sunmaktadır.

Aç gözlü yöntemin ürettiği çözümler tek başına değerli olmasa bile, diğer yöntemlerle beraber çalışarak veya onlar için bir karşılaştırma alt sınırı oluşturarak

anlamlı hale gelmektedir.

1.6.3 Genetik algoritmalar

Genetik algoritma, en iyileme problemlerinde kullanılan bir genel sezgisel arama yaklaşımdır. Probleme özel yol kullanılması yerine, genel bir yöntemle, çözüm uzayını verilen bir uygunluk işlevi yardımıyla aramayı sağlar.

Genetik algoritmalarda olası çözümlere birey denilmektedir. Bireyler soyut genom yapılarıyla tanımlanmaktadır. Algoritma sürekli olarak bir popülasyon içerisindeki bireyleri (birbirlerinden ve dışarıdan sağlanan bilgilerle) iyileştirmeye çalışarak ilerler.

Algoritmanın genel tanımında olası çözümlerin bulunduğu havuz (genom popülasyonu) rastgele üyelerden üretilse de, bu çalışmadaki gerçekleştirimde özelleştirilmiş genom türlerinin de katılımı sağlanmıştır. Özellikle başka bir çözüm yönteminin (örneğin aç gözlü algoritma) çıktısının eklenmesi, hem diğer çözüm yönteminin eksiklerini kapamakta ve hem de genetik algoritma çözümünün gücünü arttırmaktadır.

Rastgele ve diğer algoritmaların çıktılarından oluşan “genetik bilgi”, (*crossover*) çaprazlama ile belli bir süre iyileştirildikten sonra kaçınılmaz olarak, yeni bilgi akışı olmadığı için, yerel bir minimuma takılmak söz konusudur. Bu durumda genetik algoritma yönteminin mutasyon işlevi belli bir miktar çıkış olanağı sunmaktadır. Fakat daha fazla çeşitlilik sağlanması için ayrıca birden çok paralel popülasyonlu genetik algoritma türevi kullanma olanağı da sağlanmıştır.

Genetik algoritmaların istatistiksel yöntemlere dayanması özelliğinden dolayı her çıktısı aynı sonuca ulaşamamaktadır. Bu yüzden aynı girdi için birden çok kez yeniden çalıştırılması, iyi sonuç bulma olasılığını arttıracaktır.

Bu yöntem üzerine detaylı bilgi ve yerel olarak kullanılan genom yapısı Bölüm 2.3 içinde verilmiştir.

1.6.4 Ağ Akışı

Görevlerin gerçekleştirilmesinde esas amaç belli bir kısım kaynakların üslerden hedeflere taşınması olarak düşünülebilir. Bu durumda filolara kargo birimleri ve probleme taşıma problemi olarak bakmak da mümkündür.

Bu tür problemler ağ akışı altında incelenmektedir. Ağ akışı problemleri kombinasyonel optimizasyon kümesi içindedir. Ayrıca çözüm yöntemleri hem bilgisayar bilimcileri ve hem de yöneylem araştırmacılarının ilgi alanına girmektedir[Sch06].

Görevlerin seçimini en düşük maliyetli akım problemi biçiminde tanımlamak olasıdır. Fakat yapılan problem modellemesi, yani ağın topolojisi ile maliyet ve sığa değerlenin belirlenmesi, elde edilecek verimi doğrudan etkileyecektir.

Bu modellemede, problem tanımının uyumsuzluğundan dolayı iki sıkıntı ortaya çıkmaktadır. Öncelikle girdilerin tamsayı değerlerle belirtilememesi, öte yandan ağ akışı algoritmalarının tamsayı değişkenler için tanımlanmış olması, yuvarlamadan dolayı oluşan bir bilgi kaybını kaçınılmaz kılmaktadır.

Bunun yanında, problem yapısı gereği bir filo farklı yerlerde aynı birimle gösterilememektedir. Öncelikle farklı uçak türleri arasında birebir bir dönüşüm oranı (örneğin $3 \times F4 = 2 \times F16$ gibi) tanımı söz konusu değildir. Ayrıca mesafeye ve filo kabiliyetine bağlı olarak aynı tür uçakların bile farklı görevlerdeki değeri değişmektedir.

1.7 Deney Yöntemi

Çalışmadaki deneylerin girdilerini oluşturmak ve bu deneyleri gerçekleştirmek için özel hazırlanmış bir yazılım kullanılmıştır. Bu yazılımın teknik ayrıntıları ve kullanım bilgileri EK B içinde anlatılmaktadır.

1.8 Girdi Üretimi

Algoritmaların doğruluk sınamaları dışında, tüm başarım ölçümü ve karşılaştırmalar için yazılım tarafından üretilen rastgele girdilerden faydalanılmıştır.

Girdilerin hazırlanmasında üç temel parametre etkili olmaktadır. Bu parametreler, filo ve hedef sayıları, bunlara bağlı olarak görev sayısı ve son olarak sağlanan ve talep edilen kapasite oranıdır.

İlk ikisi, yani filo ve hedef sayıları çifti, veya görev sayısı farklı algoritmaların karmaşıklığını belirlemektedir. Kapasite oranı ise görevlerin ne kadarının başarılı olma olasılığının bulunduğunu göstermektedir. Örneğin, bir girdi, toplam 200 filo kapasitesi gerektiren hedef bulunduruyor ve aynı girdide sağlanan filo kapasiteleri toplamı 100 ise, büyük olasılıkla hedeflerin yarısı başarılamayacaktır.

1.8.1 Girdi üretimi kıstasları

Bir girdi kümesi ile tüm algoritmaların karşılaştırılması pratik olarak pek olanaklı değildir. Bunun önündeki en büyük engel görev sayısı yüksek olan girdiler üzerinde tam arama yapmanın makul olmayan zaman ihtiyacıdır. Ayrıca küçük girdiler de diğer algoritmaları sınamak için yeterli değildir.

Dolayısıyla ana veri kümesi tam aramanın bulunduğu ve bulunmadığı şeklinde ikiye bölünmüştür. Bunun yanında her çözücü algoritmanın kendi parametrelerinin etkilerinin gözlenmesi için özel sınamalar da yapılmıştır (örneğin genetik algoritma için popülasyon büyüklüğünün veya aç gözlü yöntemde sıralama kriterinin etkisini gözleme amaçlı deneyler).

Algoritmaların başarımlarını etkileyen diğer bir kıstasın, sağlanan ve istenen kapasite oranı olduğu belirlenmiştir. Buna bağlı olarak, girdi kümeleri kapasite oranlarına göre de ayrılmıştır.

1.8.2 Tam arama girdi kümesi

Diğer algoritmaların doğru sonuca uzaklığının bulunması ve tam aramanın karmaşıklığının modellenmesi için polinomsal olarak artan büyüklükte 6 girdilik bir küme kullanılmıştır. Bu girdilerde kapasite etkisinin en az olması için sağlanan ve talep edilen kapasite oranı $2/3$ ve 2 arasında olan girdiler tercih edilmiştir.

1.8.3 Genel girdi kümesi

Genel amaçlı karşılaştırma için üç farklı girdi kümesi kullanılmıştır. Kümeler görevlerin kapasite oranına göre seçilmiştir. Bunlar sağlanan kapasite oranının $2/3$'ten az olduğu düşük kaynak oranlı girdiler, kapasite oranının $2/3$ ve 2 aralığında olduğu normal kaynak oranlı girdiler ve kapasite oranının $3/2$'den yüksek olduğu yüksek kaynak oranlı girdilerdir.

2 KAPSAMLI ALGORİTMA TANITIMI

2.1 Aç Gözlü Çözücü Algoritması

Aç gözlü çözücü algoritması sade olmasına rağmen, genel yapısı diğer yöntemlerde de tekrarlandığı için önemlidir. Temel olarak belli bir sırada verilen tüm görevleri teker teker çözüm listesine eklemeyi deneme mantığı ile çalışmaktadır.

Şekil 2.1 içinde gösterilen bu algortima, yapılması mümkün olan her görevin yapılacağını kabul edecektir. Eğer herhangi bir görev kalan kaynaklarla gerçekleştirilemiyorsa onu atlayacak; daha önceki kabul edilenleri değiştirerek olası bir gelir arttırımını denemeyecektir.

Algoritma her görev için tek bir kez sabit zamanlı bir işlem yaptığından, algoritmanın ana gövdesinin karmaşıklığı $O(N)$ şeklindedir. Fakat önceden yapılan sıralama $O(N \log N)$ karmaşıklığında olacağından, eğer sıralama maliyeti başka bir yere aktarılmazsa, bu gerçekleştirimin karmaşıklığı da, baskın terim olan, $O(N \log N)$ olmaktadır.

2.2 Tam Arama

Tam arama, aç gözlü yöntemin çalışmasına ek olarak yaptığı seçimleri değiştirme ve bunun sonucunda her olası çözümü inceleyebilme özelliğine sahiptir. Bunu gerçekleştirirken her hedef için mümkün olan tüm görevleri teker teker deneyip, özyineli olarak kendini çalıştırmaktadır.

Şekil 2.2 içinde gösterilen tam arama gerçekleştirimi ayrıca herhangi bir noktada, önceden bulunmuşlardan daha iyi bir çözüm bulamayacağını anladığında o an aramakta olduğu dalı budama imkanına da sahiptir.

Burada **Sınır** işlevi, arama algoritması dışında soyut olarak tanımlandığı için, aynı iskeleti farklı sınırlama yöntemleri ile çalıştırmak mümündür. Ya da sürekli **YANLIŞ** döndürecek bir sınırlama işlev tanımı ile, budama yapmayan bir tam arama gerçekleştirimi elde edilir.

```
Algoritma AçGözlüÇözücü(TümGörevler, SıralamaYöntemi, Filo):
   Sırala(TümGörevler, SıralamaYöntemi)

   Çözüm = []

   Kalan[1:F] = Filo[1:F]->UçakSayısı
   Yapıldı[1:H] = 0

   Görev = [TümGörevler] İÇİN;
      EĞER Görev->UçakSayısı <= Kalan[Görev->Filo]
            VE Yapıldı[Görev->Hedef] = 0 İSE;
         Çözüm += [Görev]
         Yapıldı[Görev->Hedef] = 1
         Kalan[Görev->Filo] -= Görev->UçakSayısı
```

Şekil 2.1: Aç gözlü çözücü algoritması

2.3 Genetik Algoritma

Genetik algoritma, biyolojik evrim modelinin bilgisayarda benzetiminine (*simulation*) dayanmaktadır[For96]. Bu algoritma, tepe tırmanma (*hill climbing*) gibi yöntemlere göre, yerel en düşük noktalara daha az takılan bir genel sezgisel arama yöntemidir. Yöntemin en çok ayakta kalmış tanımı Holland tarafından yapılmıştır[For96]. Fakat ihtiyaçlara göre çok farklı değişikliklere uğramış halleri de kullanılmaktadır.

2.3.1 Temel genetik algoritma tanımı

Algoritmanın temel veri yapısı bir gen havuzudur (popülasyon). Bunun içerisinde, ikili bit dizilerinden oluşan genomlar halinde gösterilen bireyler bulunmaktadır. Algoritma, herbir adımda bu popülasyonun bireyleri üzerinde tanımlı bir kısım işlemleri kullanarak yeni bir popülasyon oluşturur. Sürekli olarak bu şekilde devam ederek bireyleri iyileştirmeye çalışır.

Genetik algoritma, başlangıçta tümüyle rastgele bireylerden oluşan bir popülasyon ile çalışmaya başlar. Herbir adımda elinde oluşan tüm bireyleri bir değerlendirme sürecinden geçirip, en uygun olanları bir sonraki nesli oluşturacak grup için seçer. Bu bireylerden en az biri tam uygunluğa ulaştığında veya en fazla adım sayısı aşıldığında ise algoritma sonlanır.

```
Algoritma TamArama(TümGörevler, SıralamaYöntemi, Filo, HedefSay):
   Sırala(TümGörevler, SıralamaYöntemi)
   TablolarıİlklendirTümGörevler, {Kalan}, {HedefGörevSeçeneği})

   Çözüm = []
   EnİyiGetiri = 0

   Ara(1, 0)

   Algoritma Ara(Hedef, ToplamGetiri):
      EĞER Hedef >= HedefSay İSE;
         EĞER ToplamGetiri > EnİyiGetiri İSE;
            EnİyiÇözüm = Çözüm
            EnİyiGetiri = ToplamGetiri

         GERİ DÖN

      EĞER Sınır(Hedef, ToplamGetiri) = DOĞRU İSE;
         GERİ DÖN

      Ara(Hedef + 1, ToplamGetiri)

      Görev = [HedefGörevSeçeneği[Hedef]] İÇİN;
         EĞER Görev->UçakSayısı <= Kalan[Görev->Filo] İSE;
            Çözüm += [Görev]
            Kalan[Görev->Filo] -= Görev->UçakSayısı

            Ara(Hedef + 1, ToplamGetiri + Görev->Getiri)

            Çözüm -= [Görev]
            Kalan[Görev->Filo] += Görev->UçakSayısı
```

Şekil 2.2: Tam arama algoritması

Bireylerin sonuca uygunluğunu değerlendirecek işlev, genom içinde kodlanmış veriyi çözüp problem veri yapısı haline getirir ve buradan aranan sonuca uzaklığını hesaplayıp bu değeri döndürür. Dolayısıyla algoritmayı kullanacak kişinin öncelikle kendi problem verisini ikili bit dizisi halinde modellemesi gerekmektedir.

Yeni popülasyonun oluşturulması sırasında, eski popülasyondan seçilen grup üzerinde iki farklı işlevden biri rastgele uygulanır. Bunlardan ilki iki farklı bireyi alıp, yerlerine yeni iki oğul birey üreten çaprazlama işlevidir. İkincisi ise bir bireyin rastgele bir bitini tersleyen mutasyon işlevidir. Yeni popülasyon bu işlevlerin çıktılarından oluşan bireylerden meydana gelir.

Çaprazlama sırasında bir bireyin genetik bilgisinin bir kısmı, bir başkasınınki ile yer değiştirilir. Oluşan iki yeni birey yeni popülasyona dahil edilir. Bu işlem farklı çözümlerdeki bilgilerin bir araya toplanmasını sağlar.

Mutasyon, bir bireyin genetik yapısının rastgele olarak *bozulması* olarak tanımlanmıştır, ve bireyin genomunun bir bitinin değillenmesi şeklinde gerçekleştirilir. Bu işlem ise genetik çeşitliliğin arttırılması ve yerel minimumların içinden çıkabilmesini sağlar.

İkili bit dizisi şeklindeki genomların kodladığı olası çözümler her zaman geçerli olmayabilir. Bu tür durumlarda uygunluk değerlendirme işlevinin hatalı kesimleri göz ardı edebilmesi veya göz ardı edilemiyorsa o çözüme cezalı kısmi bir puan vermesi beklenmektedir. Yine de geçersiz çözümlerin bulunamayacağı bir kodlama hazırlamak daha uygundur.

2.3.2 Özel genom türleri

Genetik algoritmanın değiştirilmesi ile ikili bit dizisi haricinde genom türleri kullanmak da mümkündür. Problemin ihtiyacına göre kayan noktalı sayı dizileri, iki veya daha yüksek boyutlu diziler, veya ağaç yapıları da genom olarak kullanılmaktadır.

Haliyle genetik işlevlerin (ilklendirme, çaprazlama, mutasyon) bu yeni veri yapısına göre uyarlanması gerekecektir.

2.3.3 Özel algoritma türleri

Genetik algoritmanın gücünün arttırılabilmesi için sık olarak kullanılan iki türevi bulunmaktadır. Bu türevler, bireylerin bir kısmının dokunulmadan öbür nesle

aktarılmasını ve paralel gen havuzları kullanılmasını destekler.

Her adımda gen havuzu yeni nesille değiştirildiği için bazı bilgilerin kaybolması söz konusu olabilir. “Steady State” şeklinde adlandırılan türevde, daha önceden belirlenen bir orandaki (olabildiğince en iyi) bireylerin bir sonraki nesle değiştirilmeden aktarılması sağlanır.

Tek bir gen havuzu ile, mutasyon kullanımına rağmen, yerel bir mimimuma takılma olasılığı vardır. Bunu aşmak için paralel popülasyonlar oluşturmak ve her adım sonunda bir kısım bireyleri bu popülasyonlar arasında göç ettirmek olasıdır. Ayrıca bu yöntem, genetik algoritmanın paralel bilgisayar mimarilerinde kullanımında da tercih edilmektedir.

2.3.4 Kullanılan genom yapısı

Bu çalışma kapsamındaki gerçekleştirimde, genom yapısı olarak sabit uzunluklu bir bit dizisi kullanılmıştır. Dizinin herbir elemanı bir görevin yapılmasına müsade edilip edilmediğini kodlamaktadır.

Düşük bit yoğunlukları için (H mertebesinde) bir çözümü tam olarak kodlayacak bilgi içermektedir. Daha yüksek yoğunluklarda ise, etkin durumda olan görevler içinden seçim yapılması gerekmektedir. Bu seçim halihazırdaki gerçekleştirimde aç gözlü algoritma ile yapılmaktadır.

Gösterimin eksikliği, normal durumda yüksek oranda atıl genetik bilgi içermesidir. Fakat tüm çözüm uzayını gösterebilme ve burada hızlıca hareket etme imkânı sağlamaktadır.

Kullanılmayan diğer iki genom yapısı seçeneği ise, hedeflere göre filo veya filolara göre hedef listesi tutulmasıdır. Her ikisinde de çözüm uzayını tümüyle gösterememe sıkıntısı vardır. Örnek olarak, bir görev için, 3 numaralı filo seçilmiş fakat aynı genom üzerindeki diğer görevler 3 numaralı filoyu tüketmişse, daha başka seçenekleri olsa bile, ilgili görev gerçekleştirilmeyecektir.

Eğer listelerde bir yerine birden çok seçenek saklanacak olursa, bu genom yapılarının gösterim gücü ilk tanımlanana denk hale gelecektir. Fakat bu durumda da iki boyutlu dizi ve devingen boylu yapı kullanımı gerekeceği için gerçekleştirim karmaşıklığı artacaktır. Ayrıca genetik işlevlerin (çaprazlama ve mutasyon) gücünün azalmaması için özel özen gösterilmesi de gerekecektir.

```
Algoritma GenetikAlgoritmaHazırlık(TümGörevler, SıralamaYöntemi)
   Sırala(TümGörevler, SıralamaYöntemi)

Algoritma UygunlukDeğeri(Genom):
   Kalan[1:F] = Filo[1:F]->UçakSayısı
   Yapıldı[1:H] = 0

   Değer = 0

   Görev = [TümGörevler] İÇİN;
      EĞER Genom(Görev) = 1
            VE Görev->UçakSayısı <= Kalan[Görev->Filo]
            VE Yapıldı[Görev->Hedef] = 0 İSE;
         Değer += Görev->Getiri

         Yapıldı[Görev->Hedef] = 1
         Kalan[Görev->Filo] -= Görev->UçakSayısı
```

Şekil 2.3: Genom değerlendirme işlevi

2.3.5 Kullanılan uygunluk değerlendirme işlevi

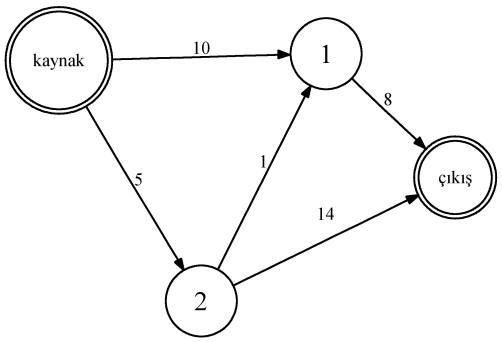
Kullanılan uygunluk değerlendirme işlevi, genom içinde kodlanmış etkin görev listesini aç gözlü algoritma ile bir alt problem şeklinde çözüp, oluşan getiri değerini döndürmektedir. İlgili kod Şekil 2.3 içerisinde gösterilmiştir.

2.4 Ağ Akış Problemleri

Ağ akış problemleri, temelde bulunan bir ağ yapısı üzerinde bir kısım malların belli bir yerden başka bir yere çeşitli kısıtlar ve hedefler dahilinde taşınması ile ilgili problemleri içermektedir. Bu problemlerin bilgisayar bilimleri, mühendislikler, yöneylem araştırması gibi alanlarda uygulamaları bulunmaktadır[AMO93].

Örnek olarak;

- Bir trafik yolu ağı kapasitesinin bulunması,
- Bir iş sürecinin veriminin arttırılması,
- Bir ağda iki nokta arasındaki en kısa mesafenin bulunması gibi problemler verilebilir.



Şekil 2.4: Örnek en kısa uzunluklu yol problem olgusu

2.4.1 Genel yapı

Ağ akış problemlerinin temelinde genellikle yönlü çizge biçimindeki bir veri yapısı kullanılmaktadır. Problemin yapısına bağlı olarak çizge üzerindeki düğümlerin sığası, kenarların maliyeti ve/veya sığası bulunabilmektedir. Ayrıca çoğu algoritma gerçekleştiriminde çizge üzerinde tüm düğümlerden diğer tüm düğümlere, her iki yönde de, — 0 sığalı bile olsa — kenar bulunduğu varsayılmaktadır.

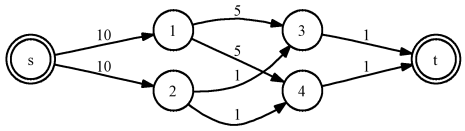
Bunun yanında algoritmalar çalışmaları sırasında düğümlere uzaklık veya benzeri başka bilgilerle etiket vermektedir.

2.4.2 En fazla akım

En fazla akım verilen ağ üzerinde bir kaynak düğümden bir çıkış düğümüne olabildiğince fazla miktarda taşıma yapma problemidir. Gerçek hayatta petrol hatlarının sığasının belirlenmesi gibi bariz olanları dışında, en büyük eşleştirmeyi bulma, veya zamanlama (*scheduling*) gibi uygulamaları da bulunmaktadır.

Problemin biçimsel tanımı şu şekildedir[AMO93]:

Tanım Sığalı bir $G = (D, K)$ ağı, her bir $(i, j) \in K$ kenarı için u_{ij} biçiminde negatif olmayan bir sığa değeri içerir.



Şekil 2.5: Örnek en fazla akım problem ağı

Tanım En fazla akım problemi, bir sığalı ağ üzerinde verilen s kaynak ve t çıkış düğümleri için aşağıdaki eniyileştirme problemi olarak tanımlanmıştır.

$MAX(v)$

$$\sum_{(j:(i,j)\in K)} x_{ij} - \sum_{(j:(j,i)\in K)} x_{ji} = \begin{cases} v, & i = s \\ 0, & \forall i \in D - (s,t) \\ -v, & i = t \end{cases}$$

$$0 \leq x_{ij} \leq u_{ij}, \forall (i,j) \in K.$$

Şekil 2.5 içerisinde örnek bir problem olgusu gösterilmektedir. Kenarlar üzerindeki etiketler ilgili sığa değerlerini belirtmektedir.

Artık ağ

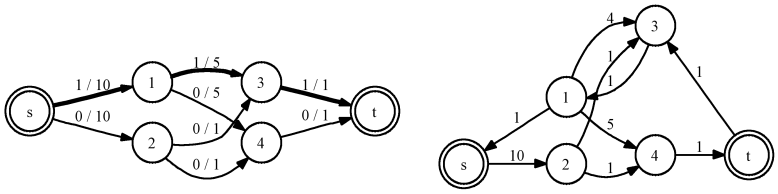
En fazla akım probleminin (ve benzer başka problemlerin) çözümünde kullanılan en önemli veri yapılarından biri artık ağ tanımıdır. Artık ağ, üzerinde belli bir miktar akım olan bir ağda ne kadar daha taşıma yapılabileceğini göstermektedir. Artık ağ üzerindeki taşıma sığaları $r_{ij} = u_{ij} - x_{ij}$ şeklinde tanımlanmaktadır. Buradaki yaklaşım, her bir kenar üzerindeki akımın kendi kalan sığasını azalttığı fakat ters yönde akım potansiyeli oluşturduğudur[AMO93].

Şekil 2.6 (a) içerisinde, daha önceden Şekil 2.5'de verilen ağ için geçerli bir akım ve (b) içerisinde ise bu akımdan sonra oluşan artık ağ gösterilmektedir.

Genel arttıran yol çözüm yöntemi

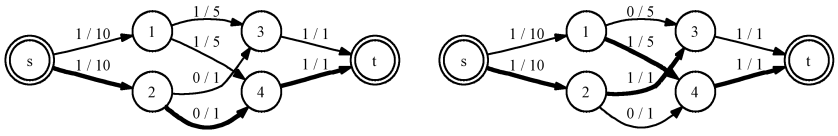
En fazla ağ akışı problemi için oluşturulan kolay anlaşılan bir yöntem sürekli olarak arttıran yolların bulunup toplanmasıdır.

Tanım Artık ağ üzerinde kaynak ve çıkışı bağlayan yollara arttıran yol adı verilmektedir. Arttıran yolun taşıma sığası, içindeki en az sığalı kenarınki kadardır[AMO93].



(a) Akım (b) Artık Ağ

Şekil 2.6: Örnek artık ağ



(a) s - 2 - 4 - t (b) s - 2 - 3 - 1 - 4 - t

Şekil 2.7: Farklı arttıran yol seçimları sonucunda oluşan akımlar

Örnek olarak Şekil 2.6 (b)'deki artık ağ üzerinde "s - 2 - 4 - t" yolu bir arttıran yoldur. Aynı şekilde "s - 2 - 3 - 1 - 4 - t" yolu da bir arttıran yoldur. İlkinin sonucunda oluşan durum Şekil 2.7 (a) içerisinde, ikincisinin seçimi sonucunda oluşan durum 2.7 (b) içerisinde gösterilmiştir.

Sürekli olarak arttıran yolları bulup, ağ akışına eklemek, en fazla ağ akışını tam olarak bulmayı sağlamaktadır[AMO93]. Bu yöntemi gerçekleştiren algoritma tanımı ise Şekil 2.8'de gösterilmiştir.

```
Algoritma EnFazlaAkım(Ağ, Kaynak, Çıkış):
   ArtıkAğ = Ağ
   Akım = []

   ArttıranYolVar(ArtıkAğ, Kaynak, Çıkış) OLDUĞU SÜRECE;
      Yol = ArttıranYolSeç(ArtıkAğ, Kaynak, Çıkış)
      Ekle(Akım, Yol)
      ArtıkAğ = Ağ - Akım

   Akım DEĞERİNİ DÖNDÜR;
```

Şekil 2.8: Genel arttıran yol en fazla ağ akışı çözüm algoritması

Polinomsal çalışma zamanı elde etme

Eğer n = düğüm sayısı, m = kenar sayısı, U en büyük sığa değeri ise, Şekil 2.8'de verilen algoritma an sade haliyle $O(nmU)$ zamanda çalışmaktadır. Bu sınır küçük veya sabit özellikli U değerleri için kabul edilebilir olsa bile, U değerinin yüksek olması durumunda üssel karmaşıklığa yol açmaktadır. Ayrıca sığa değerleri ve dolayısıyla U tamsayı değilse, algoritmanın sonlanmaması da mümkündür[AMO93].

Bu durumun sebebi, arttıran yol seçimi için belli bir ölçüt tanımlanmamış olmasıdır. Bunun sonucunda da N gibi çok yüksek sığalı bir yolu her seferinde 1 gibi küçük bir sığalı yol üzerinden N sefer geçerek doldurmak gibi durumlar ortaya çıkabilmektedir[EK72].

Algoritmayı polinomsal hale getirmek için, Ahuja ve diğerleri tarafından aşağıda yazılan yaklaşımlar olduğu belirtilmiştir[AMO93]:

- Daha yüksek sığalı arttıran yolların seçilmesi
- Arttıran yol seçiminde matematiksel bir değerlendirme yapılması
- Tümüyle farklı bir yöntem kullanılması

Bu çalışma kapsamında genel arttıran yol ağ akışı çözüm algoritması yalnızca başvuru olarak gerçekleştirildiği için ilk iki yöntem üzerinde durulmayacaktır.

Ön akım-ittirme yöntemi

Arttıran yol yöntemleri, ağın sınır kısıtlarını bozmadan sürekli olarak bir yol üzerinden akımı arttırmaktadır. Bunun yerine sınır kısıtlarını geçici olarak bozup, tüm bir yol değil sadece kenarlar üzerinden çalışarak en iyi çözümü aramak daha iyi sonuç verebilir.

Ön akım-ittirme yöntemi, gerçek hayatta boruların dolma şekline benzer bir yaklaşımla problemi çözmeyi amaçlamaktadır. Doğada, akışkanlar önce hızlı bir şekilde borulara nüfuz etmekte, tıkandığında ise başka yolları tercih etmekte veya geri doğru basınç oluşturmaktadır.

Ön akım-ittirme yönteminde de buna benzer şekilde başlangıçta ağa kaynaktan olabildiğince çok akım aktarılmaktadır. Daha sonra sürekli olarak ağın denge

kısıtlarını bozan yerler bulunup, o düğümlerde oluşan fazla akım başka yerlere aktarılmakta, eğer bu mümkün değilse kaynağa doğru geri gönderilmektedir.

Şekli canlandırmak için yukarıdan aşağıya doğru dizilmiş bir ağ yapısı düşünülebilir. Bu yapı üzerinde kaynak en yukarıda ve çıkış en aşağıda bulunmaktadır. Ara düğümler ise gergin olarak dizilmiştir.

Tanım Başlangıçta ağa aktarılan akım, ara düğümler için denge sınırını (giren ve çıkan akımın eşit olması) bozacaktır. Bu şekildeki düğümler etkin düğüm olarak tanımlanmaktadır.

Algoritmanın yaklaşımı sürekli olarak etkin düğümleri dengelemeye çalışmaktır. Yapılan işlem, eğer etkin düğümün, çıkışa kendinden daha yakın bir komşusu varsa, kendi fazlasının olabildiğince çok miktarını bu komşuya akıtmak; eğer bu türde hiçbir komşusu yoksa, makul bir akım oluşturabilmek için düğümün yüksekliğini arttırmaktır.

Düğümlerin yüksekliğini arttırmak, akımların geriye doğru, muhtemelen de yeniden kaynağa, dönmesini sağlamaktadır.

Algoritma ilk olarak Goldber ve Tarjan tarafından tanımlanmıştır[GT88]. Şekil 2.9 içerisinde ise bahsedilen ön akım ittirme en fazla ağ akışı algoritmasının bir gerçekleştirimi verilmiştir. Etkin düğüm belirlenmesinde kaynak ve çıkış düğümlerinin kapsam dışında olduğu unutulmamalıdır.

2.4.3 En düşük maliyetli akım

Tanım Bir ağ üzerinde en yüksek sığalı akımlar içinde en düşük maliyetli olanın bulunması, en düşük maliyetli akım problemi olarak tanımlanmaktadır.

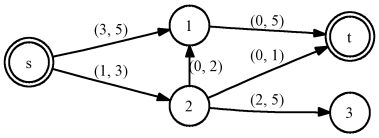
Eksi Maliyetli Çevrim Bulma

Tanım Bir ağ içinde, kenarların toplam maliyeti eksi olan döngülere, eksi maliyetli çevrim adı verilmektedir.

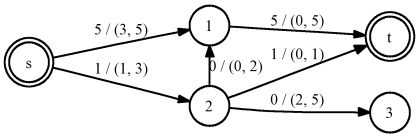
Bir artık ağ üzerinde eksi maliyetli bir çevrim bulunuyorsa, ilgili akım temeldeki ağ için bir en düşük maliyetli akım değildir. Çünkü bu çevrim üzerinden, sığasının izin verdiği miktarda akım aktararak, toplam akımı azaltmadan, toplam maliyeti düşürmek olasıdır.

```
Algoritma EnFazlaAkım(Ağ, Kaynak, Çıkış):
   Akım = []
   Yükseklik[1:n] = []
   Fazla[1:n] = 0

   YükseklikHesapla()

   ÖnAkım()

   EtkinDüğümVar(Ağ) OLDUĞU SÜRECE;
     Düğüm = EtkinDüğümSeç(Ağ)
     İttirVeyaYükselt(Düğüm)

   Akım DEĞERİNİ DÖNDÜR;

   Algoritma YükseklikHesapla():
      Yükseklik[1:n] = TümEnKısaYollar(Ters(Ağ), Çıkış)
      Yükseklik[Kaynak] = n + 1

   Algoritma İttirVeyaYükselt(Düğüm):
      Komşu = [Ağ->Komşular[Düğüm]] İÇİN;
        EĞER Yükseklik[Düğüm] > Yükseklik[Komşu] İSE;
          Miktar = MAX(Fazla[Düğüm], Ağ->Sığa[Düğüm, Komşu])

          Akım += [ (Düğüm, Komşu, Miktar) ]

          Fazla[Düğüm] -= Miktar
          Fazla[Komşu] += Miktar

          GERİ DÖN;

      Yükseklik[Düğüm] = MIN(Yükseklik[Ağ->Komşular[Düğüm]) + 1
```

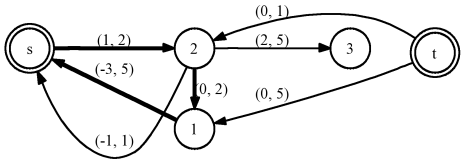
Şekil 2.9: Ön akım - ittirme en fazla ağ akışı çözüm algoritması



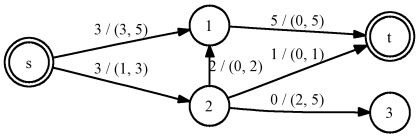
(a) Problem Olgusu



(b) En Yüksek Akım Çözümü



(c) Artık Ağ ve -2 maliyetli bir çevrim



(d) En Düşük Maliyetli Akım

(Maliyet ve sığa değerleri kenarlar üzerinde (maliyet, sıra) şeklinde gösterilmiştir).

Şekil 2.10: Örnek en düşük maliyetli akım problemi.

```
Algoritma EnDüşükMaliyetliAkım(Ağ, Kaynak, Çıkış):
   Akım = EnYüksekAkım(Ağ, Kaynak, Çıkış)

   EksiMaliyetliÇevrimVar(Ağ) OLDUĞU SÜRECE;
     Çevrim = EksiMaliyetliÇevrimSeç(Ağ)
     Akım += Çevrim

   Akım DEĞERİNİ DÖNDÜR;
```

Şekil 2.11: Eksi maliyetli çevrimlerin kaldırılması en düşük maliyetli akım algoritması

Teorem 2.1 *En düşük maliyetli bir akımın artık ağı üzerinde eksi maliyetli çevrim bulunmamaktadır. Çünkü bu tür bir çevrimin mevcut olması durumunda, bu çevrim kullanılarak daha düşük maliyetli bir akım elde edilir ve bu ilk akımın en düşük maliyetli olmasıyla çelişir.*

Teorem 2.1'nın sonucu olarak, bir en fazla akımdan sürekli olarak varolan eksi maliyetli çevrimleri çıkararak en düşük maliyetli akım elde edilebileceği ortaya çıkmaktadır. Şekil 2.7 üzerinde örnek bir en düşük maliyetli akım probleminin bu yaklaşım ile çözümü gösterilmiştir. İlgili algoritmanın tanımı ise Şekil 2.11 içinde verilmiştir.

3 DENEY SONUÇLARI ve DEĞERLENDİRİLMESİ

Algoritmaların birbirleriyle karşılaştırmasını gerçekleştirmek ve kendi parametrelerinin etkilerini gözlemlemek için, Bölüm 1.8 içinde tanımlandığı şekilde, veri kümeleri hazırlanmıştır. Bu veriler için uygulanabilir olan çözücülerin, çalışma zamanları ve buldukların çözümlerin getirileri ölçülüp, sonuçları bu bölüm içerisinde incelenmiştir.

3.1 Tam Arama Sonuçları

Tam arama yöntemini tüm girdiler üzerinde kullanmak, daha önceden belirtildiği gibi, pratik olarak mümkün değildir. Bunun yanında, tam aramanın mümkün olduğu girdi büyüklüklerini belirlemek, ve en azından bu büyüklüklerdeki girdilerde diğer algoritmaların etkililiğinin ölçülmesi için referans değerlerini hazırlamak amacıyla özel bir veri kümesi hazırlanmıştır. Bu küme için algoritmaların başarımları Çizelge 3.1 üzerinde verilmiştir.

Tam arama çalışma süresi, Şekil 3.2 içindeki grafikte de görüldüğü gibi üssel olarak artış göstermektedir[1]. Bu girdilerden en sonuncusu ise, diğer çözücülerle aynı sonucu bulmasına rağmen, 32 saatlik bir bekleme süresinden sonra, kendiliğinden tamamlanmadığı için erken olarak sonlandırılmıştır.

Burada karşılaşılan ilginç bir sonuç, dallan ve sınırla yöntemi kullanıldığında, tam aramanın en iyi başarım zamanlarını elde etmesidir[2]. Bu durum Şekil 3.1 içindeki grafikte de görülmektedir. Fakat ne yazık ki, elde edilen bu başarım ilerideki girdilerde sağlanamayacaktır.

Tam aramanın karmaşıklığından dolayı girdi büyüklükleri küçük tutularak deney yapılmıştır. Girdi büyüklüğünün artması ve ayrıca kapasite oranının düşmesi, dallan ve sınırla yönteminin de pratikte kullanılamaz hale gelmesine yol açmaktadır. Bahsedilen değişikliklerin yapıldığı ek bir veri kümesi kullanılarak yapılan bir çalışmanın elde edilen sonuçları Çizelge 3.2 içinde gösterilmiştir.

Bu ikinci veri kümesinde, dallan ve sınırla yönteminin zaman kullanımının, bazı

[1] Sonuçlardan artış oranın yaklaşık olarak $O(1.1^N)$ biçiminde olduğu gözlemlenmektedir. Fakat bu tahmin, biçimsel bir analizle yapılmadığı, için yalnızca bilgi amaçlıdır.

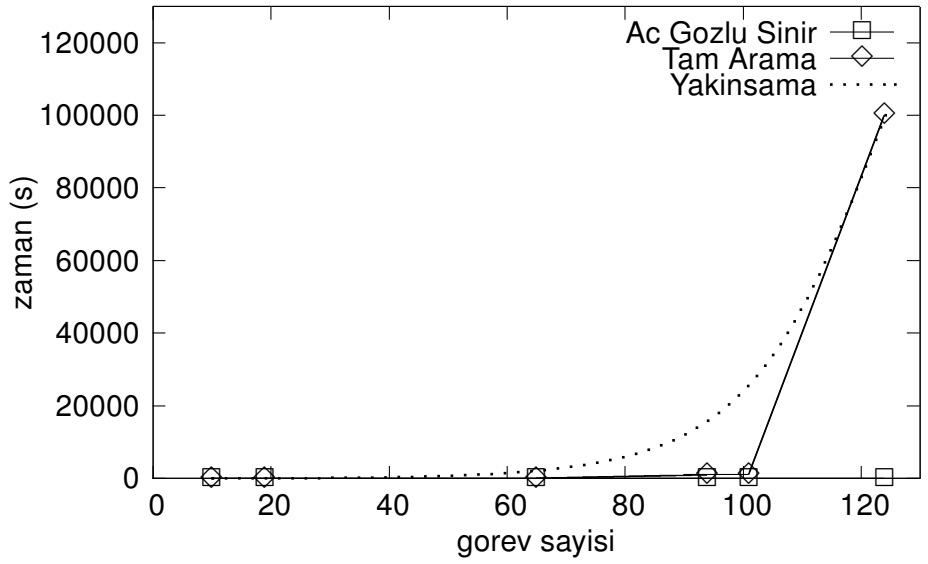
[2] Tam arama, aç gözlü algoritmaya göre bile daha hızlı çalışıyor gözükmektedir. Esasında bunun sebebi aç gözlü algoritmanın ilk olarak çalıması ve işletim ortamı maliyetlerinin üzerine yıkılmasıdır

Çizelge 3.1: Tam arama karşılaştırmalı deney sonuçları (girdi kümesi 1)

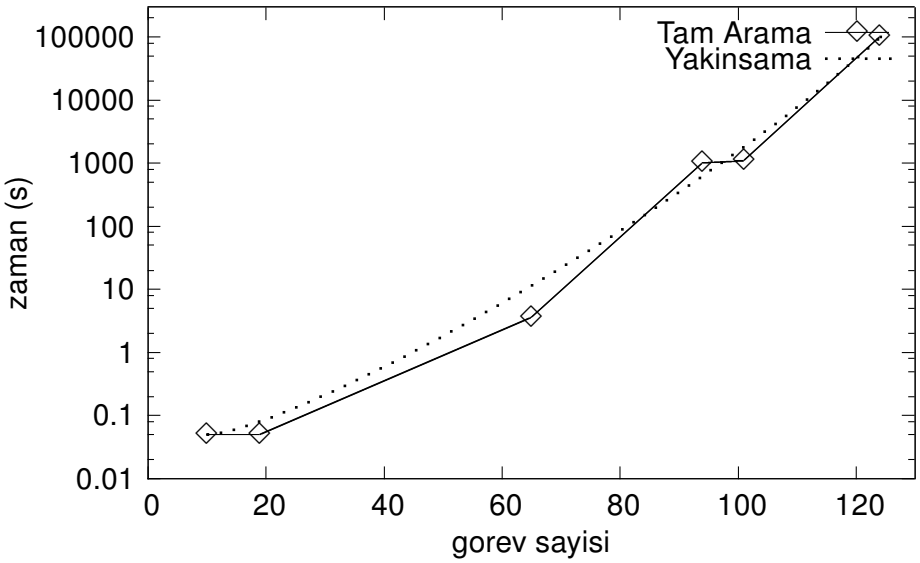
(Deney sonuçları, süre ve toplam getiri cinsinden verilmiştir).

Büyüklük	Kapasite Oranı	Tam Arama		Dallan ve Sınırla		Aç Gözlü Arama		Genetik Algoritma	
10	0.7	0.05s	112	0.08s	112	0.09s	112	0.21s	112
19	0.86	0.05s	280	0.01s	280	0.10s	280	0.21s	280
65	0.89	3.62s	463	0.04s	463	0.10s	419	0.40s	454
94	1.7	1003s	645	0.01s	645	0.10s	645	0.42s	645
101	1.2	1101s	581	0.01s	581	0.10s	581	0.29s	581
124	1.11	100000s*	664	0.01s	664	0.10s	664	0.31s	664

* (Algoritma işletimi, bu süreden sonra sonlandırılmıştır).



Şekil 3.1: Tam arama zaman kullanım grafiği (girdi kümesi 1)



Şekil 3.2: Tam arama zaman kullanım grafiği (logaritmik ölçekli, girdi kümesi 1)

girdiler için sıçrama yaptığı gözlemlenmiştir. Şekil 3.3 ve 3.4 içindeki grafikler, sıçrama yapılan yerlerin girdi büyüklüğü veya kapasite oranının bu yerleri tam açıklamadığını göstermektedir.

3.2 Etkililik Karşılaştırması

Bölüm 1.5.3 içinde belirtildiği gibi, algoritmaların etkililikleri, yani en iyi çözüme yakınlıkları, önemli bir tercih kriteridir. Bu sebeple, girdi 2, 3 ve 4 üzerinde çalışan çözücülerin, kapasite oranına göre doğruluk oranları Şekil 3.5, 3.6 ve 3.7 içindeki grafiklerde özetlenmiştir.

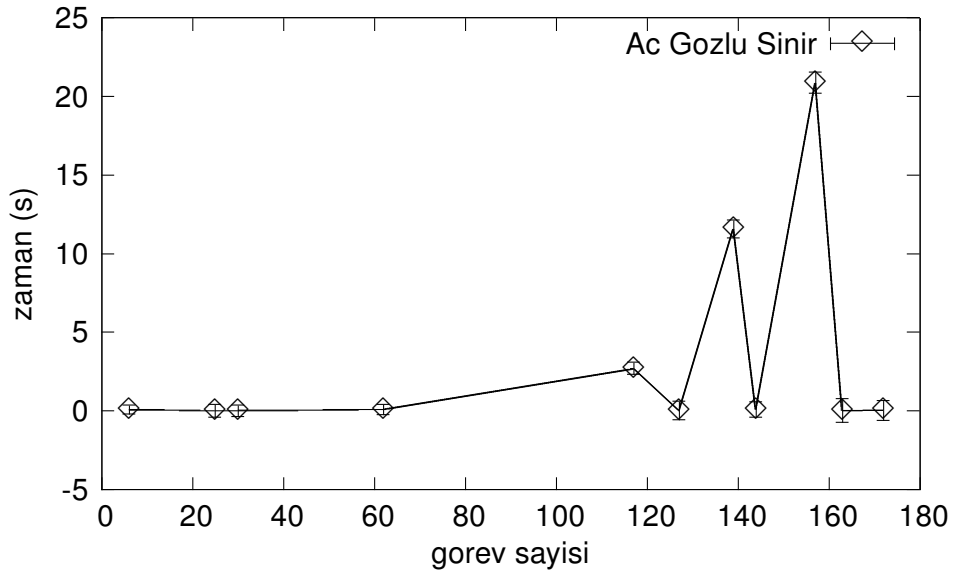
Burada önemli bir husus, birleşik grafiklerde (Şekil 3.6 ve 3.7) bilinen en iyi çözümün, her zaman olası en iyi olmadığıdır. Çünkü bazı girdiler için tam arama sonuçlanmamış ve tüm yöntemlerin sonuçları içindeki en yüksek değer, karşılaştırma değeri olarak kullanılmıştır.

Grafiklerden çıkarılabilecek iki sonuç bulunmaktadır. Birincisi, tüm çözüm yöntemleri için kapasite oranı arttıkça, etkiliğinde artmasıdır. Hatta, kapasite oranı 1 değerini geçtikten sonra, yöntemler tam çözüme ulaşmaktadır. İkincisi ise etkililiğin, tam arama, genetik algoritma, aç gözlü arama ve ağ akışı sırasında olduğudur.

Çizelge 3.2: Tam arama karşılaştırmalı deney sonuçları (girdi kümesi 2)

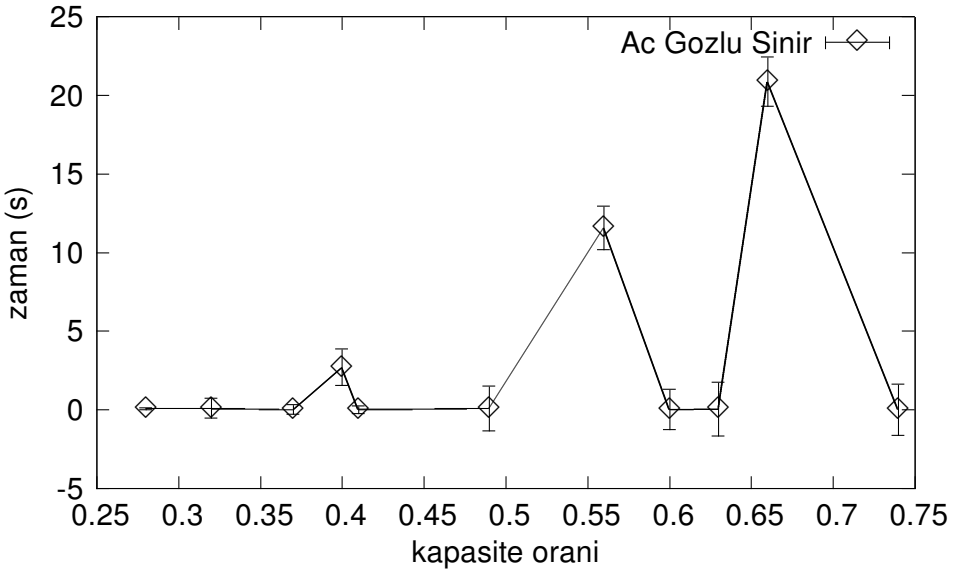
(Deney sonuçları, süre ve toplam getiri cinsinden verilmiştir).

Büyüklük	Kapasite Oranı	Dallan ve Sınırla		Ağ Akışı		Aç Gözlü Arama		Genetik Algoritma	
6	0.28	0.08s	43	0.04s	25	0.09s	43	0.22s	43
25	0.41	0.02s	71	0.01s	59	0.10s	71	0.20s	71
30	0.37	0.02s	101	0.01s	88	0.10s	101	0.20s	101
62	0.32	0.10s	154	0.05s	140	0.11s	141	0.40s	154
117	0.40	2.71s	186	0.01s	149	0.10s	186	0.32s	186
127	0.60	0.02s	218	0.02s	218	0.10s	218	0.31s	218
139	0.56	11.57s	217	0.16s	142	0.10s	203	0.37s	217
144	0.49	0.09s	227	0.02s	208	0.11s	225	0.31s	227
157	0.66	20.86s	250	0.01s	208	0.11s	236	0.33s	250
163	0.74	0.02s	264	0.02s	247	0.11s	264	0.42s	264
172	0.63	0.04s	228	0.15s	227	0.10s	226	0.42s	228



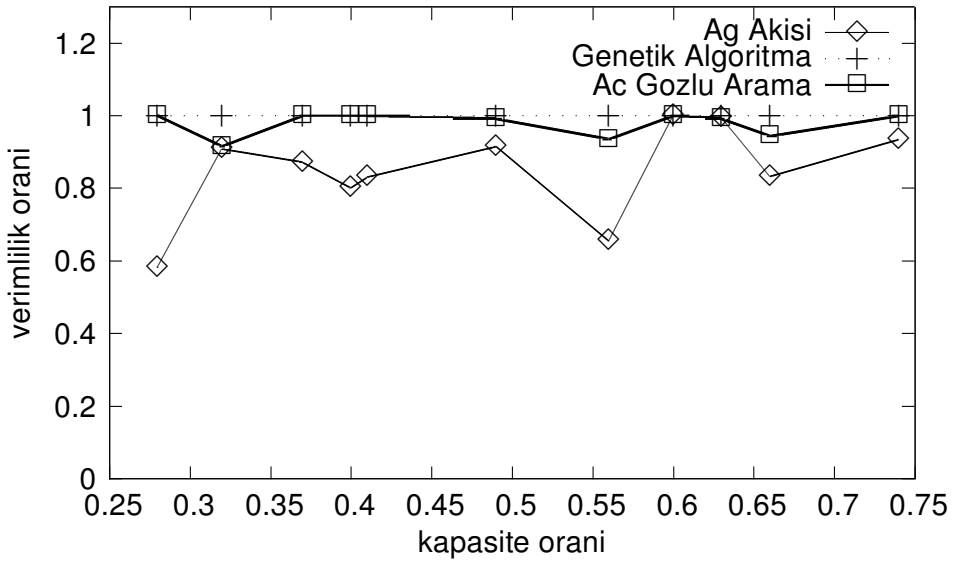
Veri noktalarındaki dikey çizgilerin uzunluğu, kapasite oranını göstermektedir.

Şekil 3.3: Girdi büyüklüğüne göre dallan ve sınırla yöntemi zaman kullanım grafiği (girdi kümesi 2)

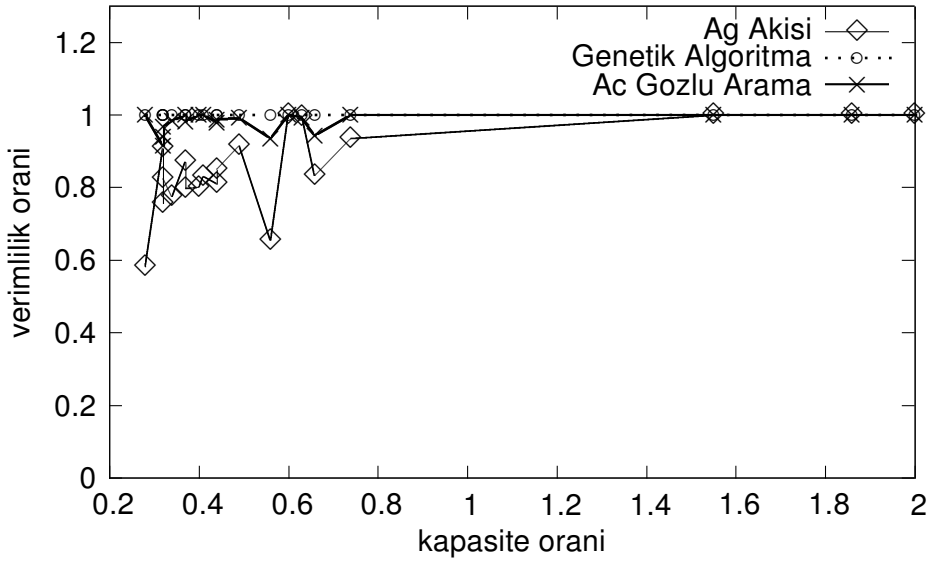


Şekil 3.4: Kapasite oranına göre dallan ve sınırla yöntemi zaman kullanım grafiği (girdi kümesi 2)

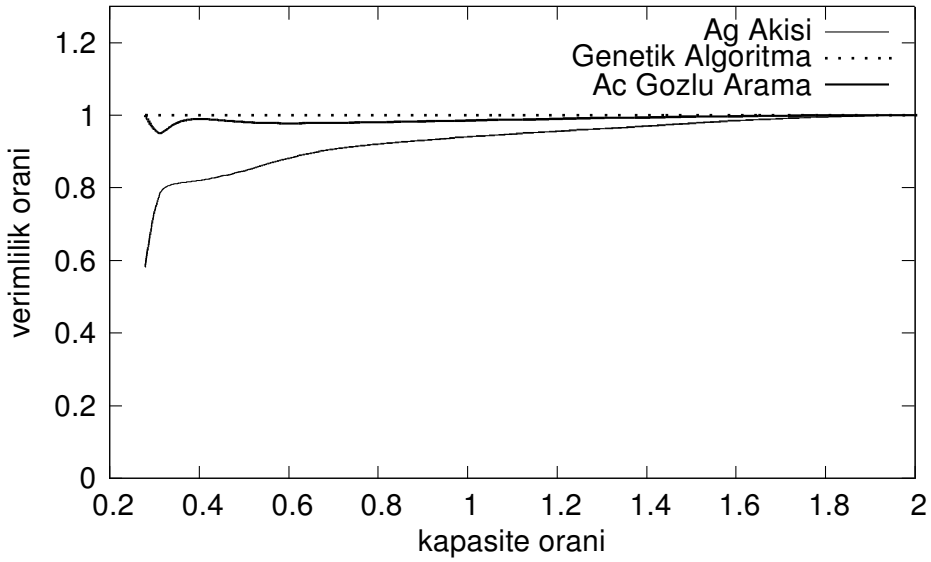
Veri noktalarındaki dikey çizgilerin uzunluğu, girdi büyüklüğünü göstermektedir.



Şekil 3.5: Kapasite oranına göre yöntemlerin verimlilik oranları (girdi kümesi 2)



Şekil 3.6: Kapasite oranına göre yöntemlerin verimlilik oranları (girdi kümesi 2, 3 ve 4)



Şekil 3.7: Kapasite oranına göre yöntemlerin etkinlik oranları (girdi kümesi 2, 3 ve 4)

Çizelge 3.3: Yüksek kapasite oranlı girdiler deney sonuçları (girdi kümesi 3)

(Bu küme içerisindeki girdilerin kapasite oranı 1.5'ten yüksek seçilmiştir. Deney sonuçları ise, süre ve toplam getiri cinsinden verilmiştir).

Büyüklük	Kapasite Oranı	Dallan ve Sınırla		Ağ Akışı		Aç Gözlü Arama		Genetik Algoritma	
2583	2.00	0.03s	782	0.34s	782	0.10s	782	1.20s	782
10749	1.55	0.07s	1597	1.70s	1597	0.11s	1597	5.52s	1597
21209	1.86	31.69s	2285	4.31s	2285	0.11s	2285	16.63s	2285

Çizelge 3.4: Düşük kapasite oranlı girdiler deney sonuçları (girdi kümesi 4)

(Bu küme içerisindeki girdilerin kapasite oranı 0.45'ten düşük seçilmiştir. Büyüklükler sınama makinasının bellek sığası ile sınırlanmıştır. Deney sonuçları ise, süre ve toplam getiri cinsinden verilmiştir).

Büyüklük	Kapasite Oranı	En İyi Çözüm	Ağ Akışı		Aç Gözlü Çözüm		Genetik Algoritma	
629	0.32	556	0.22s	421	0.12s	523	0.44s	556
2208	0.32	1102	0.20s	909	0.11s	1068	1.12s	1102
5499	0.44	1810	0.41s	1465	0.10s	1772	2.85s	1810
12255	0.44	2975	0.86s	2528	0.10s	2941	8.46s	2975
13175	0.37	2762	0.96s	2203	0.10s	2715	9.21s	2762
19244	0.34	3127	1.21s	2419	0.20s	3076	20.08s	3127

3.3 Yüksek Kapasite Oranlı Girdilerin İncelenmesi

Üçüncü bir girdi kümesi ise kapasite oranının yüksek olduğu, yani hedefleri gerçekleştirmek için gerekli olandan fazla miktarda kaynak bulunduğu durumları incelemek için hazırlanmıştır. İlgili deney sonuçları Çizelge 3.3 içerisinde sunulmuştur.

Sonuçlar incelendiğinde, tüm çözüm yöntemlerinin, kabul edilebilir bir süre içerisinde, tam sonuca ulaştığı görülecektir. Ayrıca dallan ve sınırla yönteminin de bu tür girdilerde kısa sürede sonlandığı gözlemlenmiştir.

Fakat gerçek hayatta ilgilenilen problem olgularında kaynaklar kısıtlıdır. Dolayısıyla elde edilen bu sonuçların değerledirmede ek bir katkısı bulunmamaktadır.

3.4 Genel Karşılaştırma

Son olarak, daha büyük girdiler için algoritmaların çalışma verimlerini ölçmek amacıyla dördüncü girdi kümesi hazırlanmıştır. Bu küme düşük kapasite oranlı

girdiler içermektedir. Bunların sonucunda da içindeki girdiler için tam arama algoritması ile çözülemeyen tek kümedir. Karşılaştırma değerleri ise çözücüler içindeki en iyisinin sonucu olarak kabul edilmiştir.

Bu girdi kümesinin deney sonuçları Çizelge 3.4 içerisinde görülebilir. Algoritmaların etkililikleri, daha önceden, Şekil 3.6 ve 3.7 içerisindeki grafiklerde gösterilmişti. Bu küme için çözücülerin çalışma zamanı grafikleri ise, Şekil 3.8 ve 3.9 içinde verilmiştir.

Aç gözlü yönteminin karmaşıklığı Bölüm 2.1 içerisinde $O(N \log N)$ olarak verilmişti. Şekil 3.8 içindeki grafikte ise, bunun etkisi son girdiler üzerinde görülmektedir. (Bu algoritma ilk olarak çalıştığından dolayı, çalışma ortamının ilklendirme maliyetleri olan sabit bir zaman $(0.10s)$, algoritmanın süresine eklemektedir).

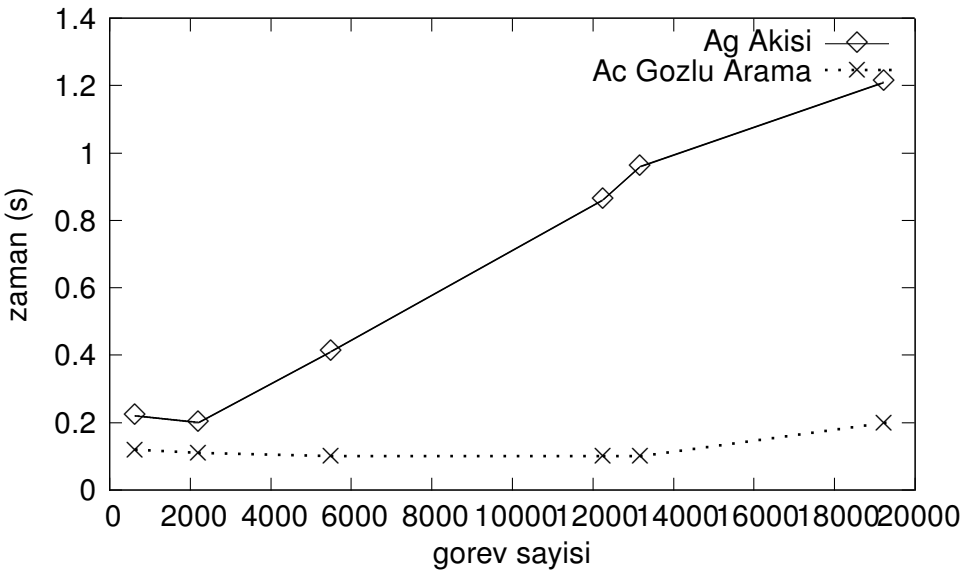
Ağ akışı çözücüsünün kullandığı, eksi maliyetli çevrim bulma en düşük maliyetli akım algoritması iyi bir teorik üst sınıra sahip değildir. n düğümlü, m kenarlı, en yüksek kenar maliyeti C ve en yüksek kenar sığası U olan bir çizge için $O(nm^2CU)$ zamanda çalışacaktır[AMO93]. Fakat buradaki problem olguları için, yaklaşık olarak doğrusal bir zaman artışı göstermiştir. Bunun sebepleri şu şekilde açıklanabilir:

- C ve U değerleri düşüktür ($C \approx$ eksi görev getiri değeri, $U \approx$ görev için gereken uçak sayısı)
- Çizge topolojisinden dolayı, kısa uzunluklu eksi maliyetli çevrimler bulmak olasıdır (Başlangıçta tüm kenarlar sıfır veya eksi maliyetlidir. Algoritmanın çalışması esnasında, artık ağlarda, artı maliyetli kenarlar oluşmaktadır).

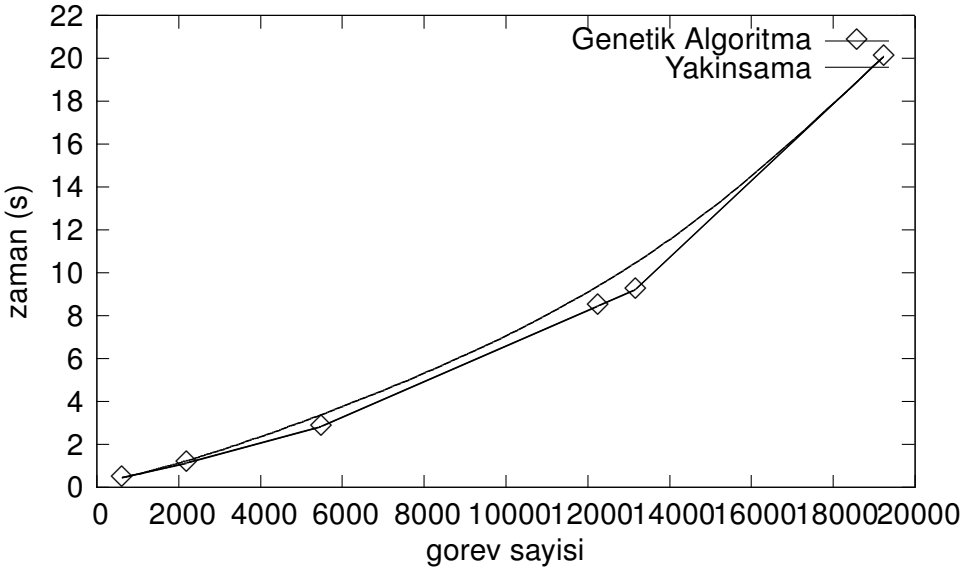
3.5 Genetik Algoritma Parametrelerinin İncelenmesi

Genetik algoritmaların çalışmasını etkileyen dört parametre bulunmaktadır. Bunlar; genetik algoritmanın türü (normal, *SteadyState* veya paralel), bir popülasyondaki birey sayısı, benzetimin adım sayısı ve mutasyon olasılığıdır. Ayrıca paralel algoritma için aynı anda bulunan popülasyon sayısı da önemlidir.

Bu parametrelerin etkililik ve verimlilik üzerine etkilerini ölçmek için, 4. girdi kümesi üzerinde ek deneyler yapılmıştır. Deney sonuçları Çizelge 3.5 (a) ve (b) içinde özetlenmiştir.



Şekil 3.8: Aç gözlü arama ve ağ akışı zaman kullanım grafiği (girdi kümesi 4)



Şekil 3.9: Genetik algoritma zaman kullanım grafiği (girdi kümesi 4)

Çizelge 3.5: Genetik algoritma parametreleri karşılaştırması (girdi kümesi 4)

(p = popülasyondaki birey sayısı, g = benzetimdeki adım sayısı değerlerini göstermektedir. Deney sonuçları, süre ve toplam getiri cinsinden verilmiştir).

Büyüklük	Kapasite Oranı	Temel Algoritma		SteadyState $p = 500, g = 50$		Paralel Popülasyonlar		Yüksek Mutasyon	
629	0.32	0.68s	551	1.20s	558	3.72s	558	1.13s	**559**
2208	0.32	2.05s	1093	3.03s	**1110**	10.95s	1107	3.00s	**1110**
5499	0.44	5.21s	1795	7.70s	1821	27.25s	1817	7.65s	**1826**
12255	0.44	15.40s	2964	23.25s	2967	80.86s	2980	21.56s	**2988**
13175	0.37	17.25s	2747	24.54s	**2779**	89.70s	2771	23.88s	2778
19244	0.34	33.18s	3115	45.32s	3141	168.00s	3139	46.79s	**3143**

(a) Genel Karşılaştırma

Büyüklük	Kapasite Oranı	SteadyState $p = 300, g = 30$		SteadyState $p = 300, g = 50$		SteadyState $p = 500, g = 50$		Yüksek Mutasyon	
629	0.32	0.44s	556	0.72s	558	1.20s	558	1.13s	**559**
2208	0.32	1.12s	1102	1.83s	**1111**	3.03s	1110	3.00s	1110
5499	0.44	2.85s	1810	4.45s	1823	7.70s	1821	7.65s	**1826**
12255	0.44	8.46s	2975	13.50s	2986	23.25s	2967	21.56s	**2988**
13175	0.37	9.21s	2763	15.14s	2770	24.54s	**2779**	23.88s	2778
19244	0.34	20.08s	3127	26.92s	3140	45.32s	3141	46.79s	**3143**

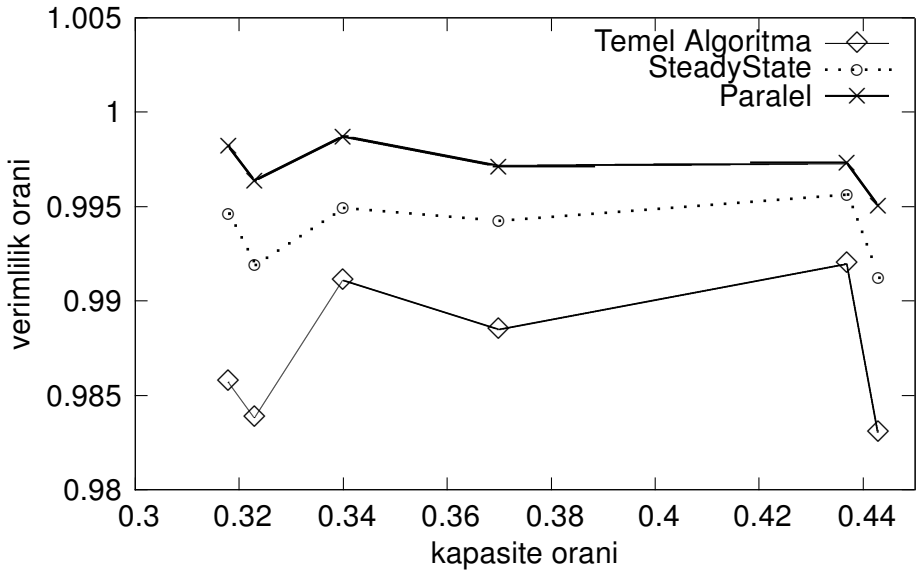
(b) *SteadyState* Parametreleri

Şekil 3.10 içindeki grafikte de görülebildiği gibi, algoritma türleri içinden paralel olanı en yüksek etkililiğe sahiptir. Bunun sebebi, paralel algoritmanın, aynı parametrelerle çalışıldığında, en yüksek ifade gücüne (*expressiveness*) sahip olması ve ayrıca birden çok paralel popülasyon kullandığı için, daha geniş bir arama yapabilmesidir.

Şekil 3.12 içindeki grafikte ise bu algoritmaların çalışma zamanı verimlilikleri incelenmiştir. İlgili grafikten de görülebileceği gibi, paralel genetik algoritma, elde ettiği etkililik avantajına karşılık yüksek çalışma zamanı maliyetine sahiptir. Ayrıca Çizelge 3.5 üzerinde de görülebileceği gibi, sınanan genetik algoritma seçenekleri içerisinde en yüksek etkililiğe de sahip değildir ($p = 500, g = 50$ için *SteadyState* daha kısa zamanda daha iyi sonuçlar elde etmektedir).

Bu grafikten ulaşılabilecek bir sonuç da, *SteadyState* yönteminin, temel algoritmadan daha hızlı çalıştığıdır. Bunun sebebi, *SteadyState* algoritmasının popülasyonun bir kesimini bir sonraki nesile değiştirmeden taşıması ve bu bireylerin uygunluk işlevlerini tekrar değerlendirmesinin gerekmemesidir.

Son olarak, Şekil 3.11 içindeki grafik incelendiğinde, mutasyon olasılığının arttırılmasının, popülasyon büyüklüğünün veya benzetim adım sayısının arttırılma-



Şekil 3.10: Farklı genetik algoritma türleri için etkililik grafiği (girdi kümesi 4)

sından daha büyük getirisinin bulunduğu görülmektedir.

3.6 Aç Gözlü Algoritmaların Karşılaştırılması

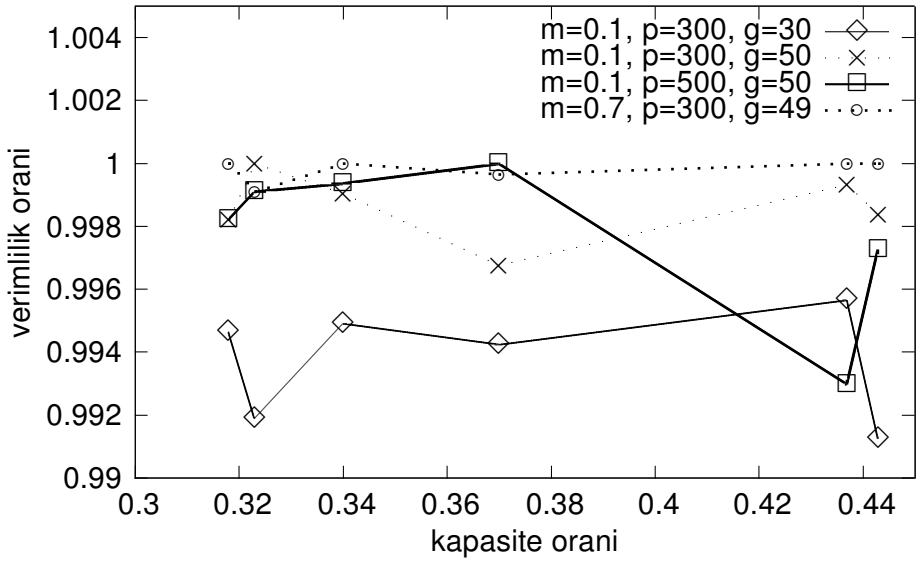
Şekil 3.13 içerisindeki grafikte farklı sıralama türlerinin, aç gözlü algoritmanın etkililiği üzerindeki etkisi incelenmiştir. Şekilden de görülebileceği gibi, çoğu girdi için, görev listesini en fazla getiriye göre sıralamak diğer yöntemlere göre daha üstündür.

Bunun yanında en fazla getiri oranına göre sıralamanın avantajlı olduğu girdiler de bulunmaktadır. Fakat elimizdeki parametreler (hedef sayısı, kapasite oranı) ve kullanılan girdiler ile bunun gerçekleştiği şartları açıklamak olası değildir.

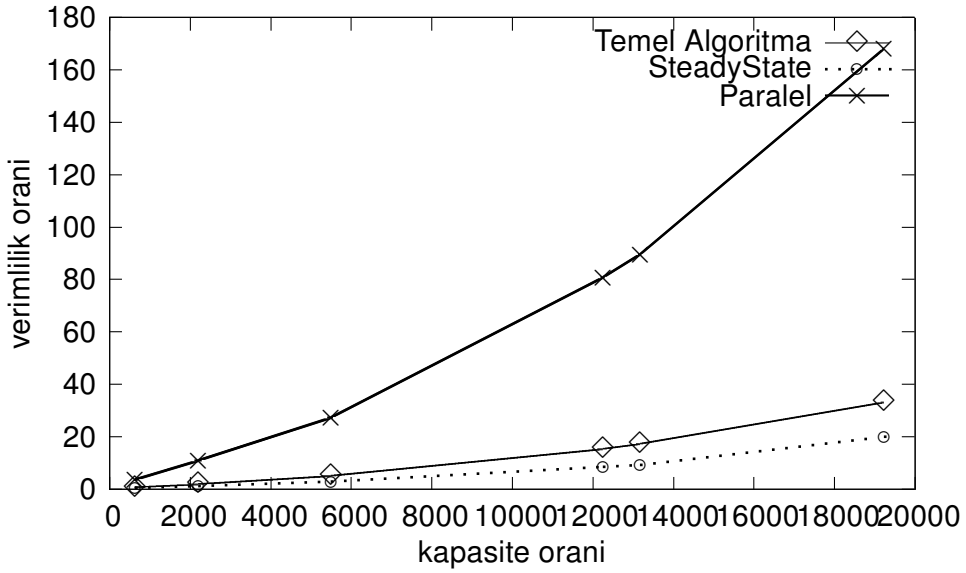
3.7 Değerlendirme

Yapılan deneyler sonucunda, pratik olarak genetik algoritmanın en uygun çözüm olduğu görülmüştür. Sınamalarda genetik algoritma, tam çözüme $\%1$'den daha yakın farkta sonuç bulmuştur.

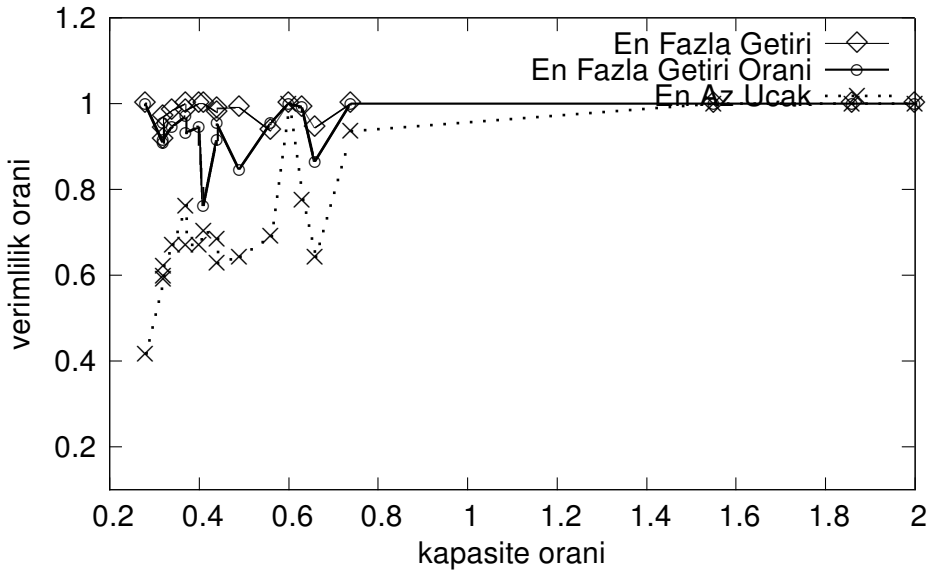
Genetik algoritma türleri içinde ise, zaman maliyeti ve etkililik getirisi dengesi açısından, en üstün durumda *SteadyState* algoritması bulunmaktadır. Çünkü temel algoritma daha düşük verimle çalışıp daha etkisiz sonuçlar elde etmektedir. Paralel



Şekil 3.11: *SteadyState* algoritmasının etkillik grafiği (girdi kümesi 4)



Şekil 3.12: Farklı genetik algoritmalar için zaman kullanım grafiği (girdi kümesi 4)



Şekil 3.13: Farklı sıralama türlerine göre aç gözlü algoritma etkililiği (girdi kümesi 2, 3 ve 4)

algoritmanın etkililik getirisi ise, benzetim adım sayısını ve popülasyon büyüklüğü arttırarak *SteadyState* ile, daha düşük zaman maliyetiyle elde edilebilmektedir.

Problemin ağ akışı biçiminde modellenmesinin başarısızlığının iki sebebi bulunmaktadır. Bunlardan birincisi girdi içerisinde rasyonel sayı değerlerin bulunması ve tamsayılaştırma algoritmasının çok az gelişmiş (aç gözlü) olmasıdır. İkinci sebep ise akımın korunması prensibi dolayısıyla, farklı filolardan farklı görevlere aynı birimde taşıma yapılması zorunluluğudur. Uçaklar birbirine dönüştürülemediği için, bu kısıt bilgi kaybına yol açmaktadır.

Dallan ve sınırla ile tam aramanın bazı durumlarda büyük girdiler için bile makul süre içinde çalışabildiği gözlemlenmiştir. Fakat hangi tür girdiler için hızlı çalıştığına dair kural bulunamamıştır. Bunun sonucunda, dallan ve sınırla yönteminin genetik algoritma ile paralel olarak başka bir makinada denenmesi önerilebilir.

4 Gerçek Zaman Kısıtları

Gerçek hayatta askeri kaynak planlama probleminin çözümünün gerçek zamanda yapılması beklenmektedir. Gerekli zaman kısıtı, harekat öncesi çalışmalar için 1-2 gün, harekat sırasındaki değişiklikler için saat mertebesindedir.

Ayrıca problem şartlarında değişiklik olması durumunda, daha önceden yapılan çözüm çalışmasının sonucunun veya geçmişinin, kısmen dahi olsa, yeni çözüm aramasında kullanılması beklenebilmektedir. Bunun sağlanması, yeni çözümün muhtemelen daha kısa sürede, veya aynı sürede içinde ise daha etkili bir şekilde, bulunmasına yardımcı olacaktır.

Bu çalışmada kullanılan yöntemlerden yalnızca genetik algoritma bu beklentileri sağlamaktadır. Çünkü genetik algoritmanın gerçekleştireceği adım miktarı ve her bir adımdaki iş büyüklüğü, parametre olarak verilmektedir. Böylelikle davranışı önceden kestirilebilir hale gelmektedir. Her bir adımın çalışma süresi de kısa ($O(N \log N)$) olduğu için, toplam adım sayısını sınırlayarak katı gerçek zaman (*hard real time*) kısıtlarına uyulabilir. Ayrıca en son üretilen popülasyonu, yeni probleme uyarlayıp, ilk popülasyon olarak kullanıp, eski çözümden faydalanmak da olasıdır.

Bunun dışında, dallan ve sınırla yöntemi, önce başka bir yöntemle alt sınır (*lower bound*) üretilip, özel bir zaman planlayıcı kullanılarak gerçek zamanlı bir sisteme uyarlanabilir. Bu tür zaman planlayıcıların tanıtımı Liu ve diğerleri tarafından yapılmıştır[LLS$^+$91].

5 Açık Konular

5.1 Gerçek hayata uygun girdiler

Çözücü algoritmaların sınanması sırasında, rastgele üretilmiş girdiler kullanılmıştır. Bunların yanında, gerçek hayattaki görevlerin de sınanması ve yöntemlerin zaman verimliliği ve etkililikleri açısından, hava kuvvetlerince kullanılabilir olup olmadığının ölçülmesi gerekmektedir.

Ayrıca, deney yazılımının yapısı dolayısıyla, tüm girdi, ara veri ve sonuçlar bellek içinde tutulmaktadır. Başlangıçta yazılım sağlamlığı açısından bu tercih yapılmıştır. Lakin, daha büyük girdilerin kullanılabilmesi için bir tampon bellek yöneticisi hazırlanması gerekmektedir.

5.2 Eksik yöntemlerin tamamlanması

Özellikle ağ akışı yönteminin geliştirilmesi gerekmektedir. Bölüm 3.7 içinde deney sonuçlarının değerlendirilmesinde anlatılan eksiklikler şu şekilde giderilebilir:

Rasyonel değerler Girdinin tamsayı olmaması için bir kısıt yoktur. Şu anki rasyonel değerli girdi tanımı, çözücülerin sınırlarını sınamak için tercih edilmiştir. Bu değiştirilerek ağ akışı için daha iyi sonuçlar elde edilebilir.

Akımın korunması En düşük maliyetli akım bulunurken, ağ yapısında, akımın korunması kuralını esnetecek bir tanım kullanılması mümkündür. Bu tür problem olgularına genelleştirilmiş akım adı verilmektedir[AMO93].

Bunun dışında tam bir karşılaştırma yapılabilmesi için tamsayı doğrusal programla yönteminin de çözücülere dahil edilmesi uygun olacaktır. Bu çalışmada eklenememesinin sebebi, varolan tamsayı doğrusal programlama çözücülerinin, yazılımın hazırlandığı *.Net* programlama ortamına teknik uyumsuzluğudur.

5.3 Yöntemlerin bir arada kullanılması

Birden çok algoritmanın birbirini destekleyecek şekilde kullanımı ve genel sezgisel yaklaşımlara problem alan bilgisinin eklenmesi verimliliği önemli ölçüde arttır-

maktadır. Bu daha önceden yapılmış çeşitli çalışmalarda da gösterilmiştir[LSL03], [LSL02] ve [AKJO03].

Gerçekleştirilen yazılım altyapısı içerisinde, yöntemleri bir arada kullanmak için destek bulunmaktadır. Fakat kullanıcı arayüzü ve ara kod eklentileri ile tamamlanması gerekmektedir. Bunun sonucunda aşağıdaki melez yöntemlerin sınanması faydalı olabilir:

- En düşük maliyetli akım çıktısının dallan ve sınırla ile tamsayılaştırılması
- En düşük maliyetli akım çıktısının genetik algoritma ile tamsayılaştırılması
- Genetik algoritma çıktısının, kısmen bozulup, üretilen alt problemin dallan ve sınırla ile tekrar çözülmesi
- Genetik algoritma çalışması sırasında popülasyonun soylulaştırmaya tabi tutulması
- Bir üst algoritma ile yöntemlerin sürekli kısmi olarak çalıştırılıp, bir sonraki alt problemin sezgisel olarak başka bir çözücüye devredilmesi

5.4 Tahmin karmaşıklığının incelenmesi

Problemin, karar verme problemi karşılığının karmaşıklık incelemesi yapılmış, fakat optimizasyon halinin tahmin zorluğu değerlendirilmemiştir. Karar verme problemlerindeki karmaşıklık kümelerinin (NP, $NP-complete$, vs) benzeri olarak, tahmin zorlukları için $APX-complete$ gibi kümeler bulunmaktadır. Bu tür bir çözümlemenin nasıl yapılacağı Cescenzi ve diğerleri tarafından anlatılmıştır[Pie00].

5.5 Çok amaçlı arama

Bu çalışmada, yalnızca tek bir amaç optimizasyonu yapılmıştır. Getirinin en fazlalaştırılması yanında, risklerin de en aza indirgenmesi istenebilir.

Çok amaçlı aramada temel olarak iki yöntem bulunmaktadır. Birincisi tüm değerlendirme işlevlerinin tek bir işlev içinde (ağırlıklı toplam, çarpım, vb yöntemlerle) birleştirimesi, diğeri ise çok boyutlu sıralama yapılmasıdır.

Çok boyutlu sıralamada, değer karşılaştırılmasında dört sonuç bulunmaktadır: birincisinin baskın olması, ikincisinin baskın olması, her ikisinin eşit olması ve hiçbirinin baskın olmaması. Baskın olma, bir değer vektörünün tüm elemanlarının diğerinin karşılık gelen elemanlarından büyük veya eşit olması durumudur.

İlk yöntemi halihazırdaki gerçekleştirim içerisine uyarlamak mümkündür. İkincisi için yeni algoritma gereçekleştirimi gerekmektedir. Bir seçenek olarak, genetik algoritmaların çok amaçlı aramalarda nasıl kullanılacağı Coello tarafından incelenmiştir[Coe99].

6 SONUÇLAR ve TARTIŞMA

Bu tez çalışması kapsamında, hava harekât görevlerinde filoların hedeflere atanması modellenmiş, ve bu problemin çözümü için, tam arama, aç gözlü arama, genetik algoritma ve ağ akışı tabanlı çözücüler gerçekleştirilmiştir.

Bahsi geçen yöntemlerin bir arada kullanılabildiği, genişleyebilir bir yazılım çatısı başarı ile gerçekleştirilmiş ve bu sistemin kullanımı sayesinde faklı yöntemlerin güçlü ve zayıf olduğu noktalar tespit edilmiştir.

Gerçekleştirilen yöntemlerin girdi büyüklüğü ve karmaşıklığına göre, zaman ve doğruluk açısından farklı başarım durumlarında olduğu gözlemlenmiştir. Artan büyüklük ve karmaşıklık için, sırasıyla aç gözlü yöntem, dallan ve sınırla destekli tam arama ve genetik algoritma çözücülerinin kullanılmasının uygun olduğu belirlenmiştir.

Sağlanan / istenen kapasite oranının düşük olduğu girdilerde genetik algoritma makul sürede çalışmaktadır. Kapasite oranının denk ve girdi büyüklüğünün nispeten daha küçük olması durumunda, dallan ve sınırla ile tam arama yapmak olanaklı hale gelmektedir. Kapasite oranının 1.8 veya daha yüksek olduğu girdiler için aç gözlü bir arama yapmak yeterlidir. Bu sonuçların çıkarılması ve ayrıntılı incelenmesi Bölüm 3.7 içerisinde yapılmıştır.

Daha ileriki çalışmalarda neler yapılabileceği ise Bölüm 5 içerisinde incelenmiştir. Özellikle, problemin ağ akışı uyarlamasının geliştirilmesi, makul zamanlı (polinomsal) bir çözüm yönteminin de araç kümesine eklenmesini sağlayacaktır.

Bunun yanında Bölüm 4 içerisinde anlatıldığı üzere, hazırlanan sistemin, uygun bir zaman planlayıcı ile, gerçek zaman kısıtlarının olduğu uygulamalarda kullanımı mümkün hale gelecektir.

Son olarak, deneylerde kullanılan sınama girdilerinin, bir tamsayı karışık programlama çözücüsü ile de çözülmesi, hazırlanan yöntemlerin getirisinin anlaşılırlığını arttıran bir çalışma olacaktır.

KAYNAKLAR

[AKJO03] Ravindra K. Ahuja, Arvind N. Kumar, Krishna Jha, and James B. Orlin. Exact and Heuristic Methods for the Weapon Target Assignment Problem. *SSRN eLibrary*, 2003.

[AMO93] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows : Theory, Algorithms, and Applications*. Prentice Hall, Englewood Cliffs, NJ, 1993.

[ATZ04] Douglas Aberdeen, Sylvie Thiébaux, and Lin Zhang. Decision-theoretic military operations planning. In Shlomo Zilberstein, Jana Koehler, and Sven Koenig, editors, *ICAPS*, pages 402–412. AAAI, 2004.

[Bud] Selçuk Budak. İngilizce türkçe psikoloji sözlüğü (Çevrimiçi). `http://www.termbank.net/psychology/`.

[Coe99] Carlos A. Coello Coello. A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information Systems*, 1(3):269–308, 1999.

[EK72] Edmonds and Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *JACM: Journal of the ACM*, 19, 1972.

[For96] Stephanie Forrest. Genetic algorithms. *ACM Computing Surveys*, 28(1):77–80, March 1996.

[GT88] A. Goldberg and R. E. Tarjan. A new approach to the maximum flow problem. *Journal of the ACM*, 35:921–940, 1988.

[HL05] F. S. Hiller and G. J. Lieberman. *Introduction to Operations Research*. Hoplden-Day, 8th edition, 2005.

[Lag98] Michail G. Lagoudakis. The 0-1 knapsack problem - an introductory survey, September 01 1998.

[LCB04] Vincent Chi-Wei Li, Guy L. Curry, and E. Andrew Boyd. Towards the real time solution of strike force asset allocation problems. *Computers & OR*, 31(2):273–291, 2004.

[LLS+91] Jane W. S. Liu, Kwei-Jay Lin, Wei-Kuan Shih, Albert Chuang shi Yu, Jen-Yao Chung, and Wei Zhao. Algorithms for scheduling imprecise computations. *Computer*, 24(5):58–68, May 1991.

[LSL02] Zne-Jung Lee, Shun-Feng Su, and Chou-Yuan Lee. Parallel ant colonies with heuristics applied to weapon-target assignment problems. In *Proceedings of the 7th Conf. On Artificial Intelligence and Applications, Taichung, Taiwan*, 2002.

[LSL03] Zne-Jung Lee, Shun-Feng Su, and Chou-Yuan Lee. Efficiently solving general weapon-target assignment problem by genetic algorithms with greedy eugenics. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 33(1):113–121, 2003.

[Met89] F.L. Metler, W.A.; Preston. "solutions to a probabilistic resource allocation problem". *Decision and Control, 1989., Proceedings of the 28th IEEE Conference on , vol., no.pp.1606-1611 vol.2, 13-15*, Dec 1989.

[Pie00] A compendium of NP optimization problems, 20 March 2000.

[Pis95] D. Pisinger. *Algorithms for Knapsack Problems*. PhD thesis, University of Copenhagen, Dept. of Computer Science, February 1995.

[Sch06] Alexander Schrijver. A course in combinatorial optimization, 2006.

[SIP96] M. SIPSER. *Introduction to the Theory of Computation*. PWS, Boston, MA, 1996.

[SJ97] GRIGGS B. J. PARNELL G. S. and LEHMKUHL L. J. An air mission planning algorithm using decision analysis and mixed integer programming. *Operations research vol. 45*, 1997.

[TBD] Türkiye Bilişim Derneği TBD. Bilişim terimleri sözlüğü (Çevrimiçi). `http://www.tbd.org.tr/genel/sozluk.php`.

[TBD96] Türkiye Bilişim Derneği TBD. *Bilişim Terimleri Sözlüğü*. Türkiye Bilişim Derneği, 1996.

[TDK] Türk Dil Kurumu TDK. Güncel türkçe sözlük (Çevrimiçi). http://www.tdk.gov.tr/.

EK A PROBLEM KARMAŞIKLIĞININ ÇÖZÜMLENMESİ

Çalışılan probleme çözüm yöntemi geliştirmeye başlamadan önce, problemin zorluğunu belirleyip, olası hedef algoritma kümesini daraltmak faydalı olacaktır. Örneğin tam aramadan daha iyi çözümü olamayacak bir problem için, daha iyi bir çözüm aramak makul değildir. Problemlerin zorluklarının belirlenmesi ise karmaşıklık teorisinin ilgi alanına girmektedir.

Karmaşıklık teorisinde, karar verme problemlerinin çalışma süreleri için P ve NP kümeleri tanımlanmıştır. P kümesi içindeki algoritmalar, gerekirci makinalar üzerinde polinomsal zaman içinde çalışabilmetedir. NP kümesi elemanları ise, gerekirci olmayan makinalar üzerinde polinomsal zaman içinde çalışabilmektedir. P ve NP arasındaki ilişki $P \subseteq NP$ şeklindedir. P'nin, NP'ye eşit mi, yoksa öz alt kümesi mi olduğu bilinmemektedir[SIP96].

Karmaşıklık teorisinde, NP içindeki en zor problemler kümesi $NP - complete$ olarak adlandırılmaktadır. NP içerisindeki tüm problemler P zamanda çalışan bir çevirme algoritması ile $NP - complete$ içerisindeki problemlere dönüşebilmektedir. Yani $NP - complete$ içindeki bir problem, en az tüm NP problemler kadar zordur. Dolayısıyla, $NP = P$ olduğu gösterilmediği sürece, $NP - complete$ içindeki herhangi bir problemin P zamanda çözümü olası değildir.

A.1 Problemin Karar Verme Problemi Olarak Tanımlanması

Karmaşıklık teorisi genel olarak karar verme problemleri ile ilgilendiği için, Bölüm 1.3 içindeki tanımın bir karar verme problemi türevinin oluşturulması gerekmektedir. Aşağıdaki tanımda bu tür bir problemin P gibi bir değere eşit veya daha iyi bir çözümünün olup olmadığı sorulmaktadır.

Tanım -

Problem Kaynak planlama karar verme problemi

Olgu U filolar ve $m(u) \in M$ her bir filodaki uçak miktarı; N hedefler; bir X görev kümesi için, $h(x) \in N$ ilgili hedef, $f(x) \in U$ ilgili filo, $k(x) \in Z^+$ kullanılan uçak miktarı, $g(x) \in R^+$ görev getirisi

Soru X kümesinin aşağıdaki gibi bir X' alt kümesi mevcut mudur?

$$\forall n \in N \left(\sum_{x \in X',\, h(x)=n} 1 \right) \leq 1, \;\; \forall u \in U \sum_{x \in X',\, f(x)=u} k(x) \leq m(u),$$

$$\sum_{x \in X'} g(x) \geq P$$

A.2 Polinomsal Zamanlı İndirgeme

Tanım Eğer bir A problemi, polinomsal zamanda çalışan bir B problemine dönüşebiliyorsa, A problemi B problemine polinomsal zamanda indirgenebilir denilmektedir. Bu durum $A \leq_p B$ biçiminde gösterilmektedir[SIP96].

Tanım Aşağıdaki şartları sağlayan bir B problemi $NP - complete$ kümesi içindedir-[SIP96]:

- B, NP içinde ise ve
- NP içindeki her A problemi, polinomsal zamanda B problemine indigenebiliyorsa.

Bu şartları sağlamak için teker teker tüm problemleri indirgemek gerekli değildir. Bunun yerine NP içinde olduğu bilinen bir problemin indirgenmesi, diğer tümünün de indirgenebildiğini göstermek için yeterlidir.

Teorem A.1 *0-1 Sırt çantası karar verme problemi problemi, kaynak planlama karar verme problemine indirgenebilir.*

0-1 Sırt çantası karar verme problemi, 0-1 sırt çantası probleminin, karar verme problemi haline dönüşmüş şeklidir. 0-1 sırt çantası karar verme problemi problemi $NP - complete$ içindedir. Tanımı Lagoudakis tarafından şu şekilde yapılmaktadır-[Lag98]:

Tanım Problem 0-1 sırt çantası karar verme problemi

Olgu Sonlu V nesne kümesi için, her $v \in V$ elemanı için, $w(v) \in Z^+$ ağırlık; $p(v) \in Z^+$ getiri. Ayrıca W (sığa), P (amaçlanan getiri) tamsayı değerleri

Soru V kümesinin aşağıdaki gibi bir V' alt kümesi mevcut mudur?

$$\sum_{v \in V'} w(v) \leq W, \sum_{v \in V'} p(v) \geq P$$

0-1 sırt çantası karar verme probleminin, kaynak planlama karar verme problemine indirgenmesi şu şekilde yapılmaktadır:

- Tek bir filo: $U = u$ ($M(u) =$ sırt çantasının W değeri kadar uçak)
- Herbir nesne için bir hedef: $N = V$
- Yine herbir nesne için bir görev: $X = V$
- Her görevin hedefi ayrık: $h(x) = x$
- Her görev varolan tek filo için tanımlı: $f(x) = u$
- Her görevin kullandığı uçak miktarı, nesne ağırlığı: $k(x) = w(x)$
- Her görevin getirisi, nesne getirisi: $g(x) = p(x)$
- Amaçlanan getiriler eşit (P)

İspat Yukarıda tanımlı indigeme sonucu oluşan kaynak planlama problem olgusunun çözümü aynı zamanda ilgili 0-1 sırt çantası probleminin de çözümüdür. Çünkü, her hedef en fazla bir kez gerçekleştirileceği için, bir nesnenin birden çok kez seçilmesi mümkün değildir. Ayrıca filodan sağladığından fazla uçak talep edilemeyeceği için de sığa sınırının aşılması olası değildir. Son olarak, P'den büyük veya eşit getirili sırt çantası doldurulup, P'den büyük veya eşit getirili kaynak planlanaması mümkün olamaz. Çünkü sırt çantasına eklenebilen her nesneye karşılık gelen görevin, kaynak planına eklenmesi de mümkündür.

Ayrıca, kaynak planlama karar verme problemi, açık olarak NP içerisindedir. Çünkü verilen bir çözüm olgusunun toplam değerini bulmak ve şartlara uygunluğunu denetlemek işleri P zaman içerisinde gerçekleştirilebilmektedir.

Bunların sonucunda, her iki şartı da sağladığından dolayı; kaynak planlama karar verme problemi $NP - complete$ kümesi içindedir.

EK B DENEY YAZILIMI KULLANIM BİLGİLERİ

B.1 Projeler

Gerçekleştirilen deney yazılımı, proje tabanlı olarak çalışmaktadır. Bir girdi kümesi ile ilgili; girdi üretme parametreleri, üretilen girdiler ve ara veriler, algoritma seçenekleri ve çalışma sonuçları bu projeler içerisinde saklanmaktadır.

Şekil B.1 içerisinde projelerin genel hali görülebilir. Sol taraftaki ağaç yapısı, projenin alt öğelerinin düzenlenmesini ve yönetilmesini sağlamaktadır.

B.2 Rastgele Girdi Üretme Parametreleri

Girdi parametreleri Şekil B.3 içinde görüldüğü gibi, boş olarak veya hazır bir başkasını kopyalayarak oluşturulabilir[1].

Girdi üretimini etkileyen parametreler şu şekilde özetlenebilir:

- *Bases* (Üsler)

  Base count toplam üs sayısı

  SquadCountAverage bir üs üzerindeki ortalama filo sayısı

  SquadCountVariance bir üs üzerindeki filo sayısının varyansı

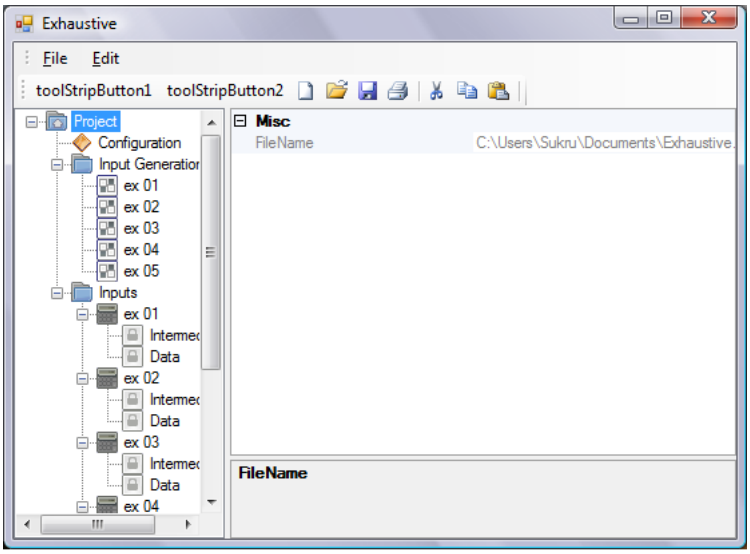
  (Toplam filo miktarı $\cong$ toplam üs sayısı $\times$ bir üs üzerindeki ortamalama filo sayısı. Büyük üs sayıları için varyansın etkisi kalkmaktadır).

- *Missions* (Görevler)

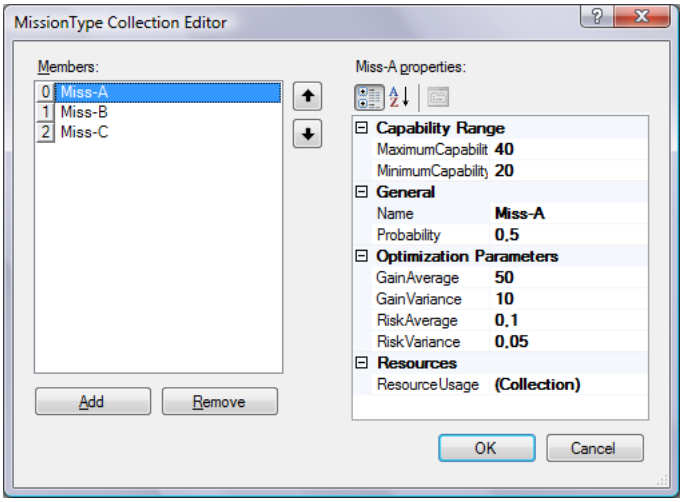
  MissionTypes Görev türleri. Her bir görev türünün; varolma olasılığı, gerektirdiği kabiliyet aralığı, getiri ortalaması ve varyansı ve risk ortalaması ve varyansı bulunmaktadır. Örnek bir görev türü tanımı Şekil B.2 içerisinde görülmektedir.

  TargetCount Hedef sayısı (her hedef için, tüm görevlerden ayrı ayrı bulunması olasılığı vardır).

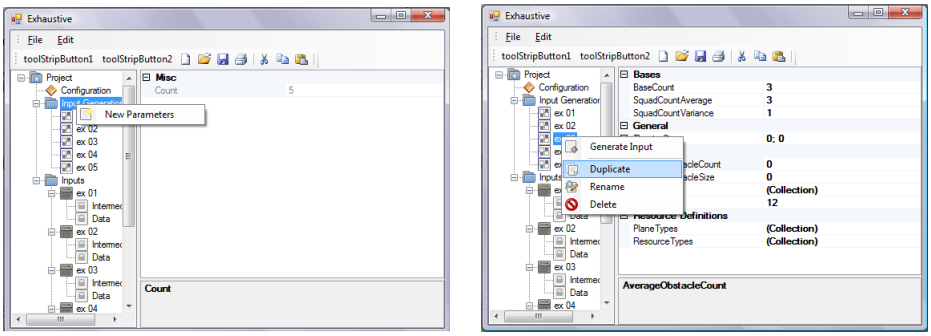
[1] Proje içerisindeki çoğu nesnenin "*duplicate*" komutu ile çoğaltılması mümkündür



Şekil B.1: Proje görünümü

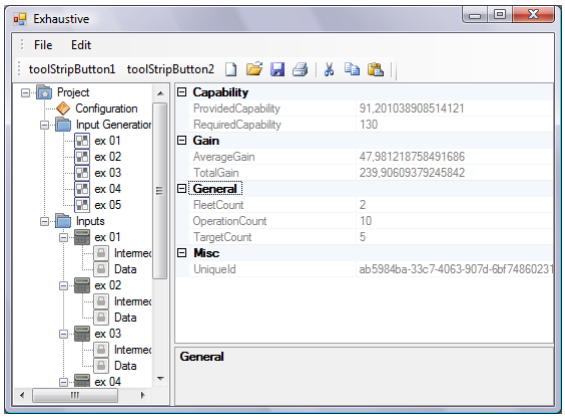


Şekil B.2: Görev türü tanımı



(a) Yeni parametreler (b) Çoğaltma

Şekil B.3: Girdi üretme parametresi oluşturma



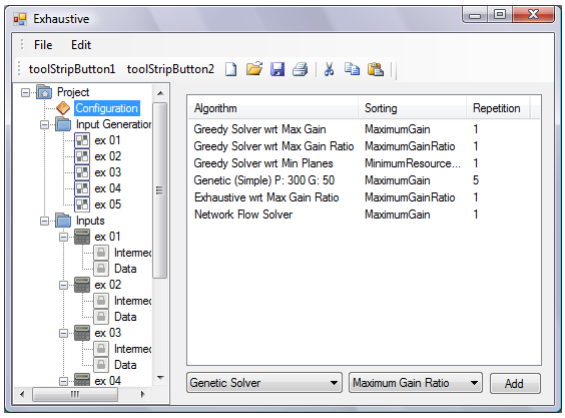
Şekil B.4: Girdi nesneleri

- *Resource Definitions* (Kaynak Tanımları)

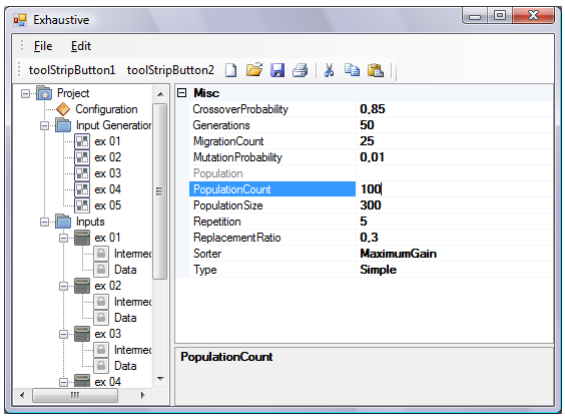
  PlaneTypes Uçak türleri. Herbir uçak türünün, görevlere benzer şekilde; her bir filodaki sayı aralığı ve bir uçağın sağladığı ortalama kabiliyet değeri bulunmaktadır.

B.3 Girdiler

Girdiler, girdi üretme parametrelerinin işlenmesi sonucu elde edilmektedir. Proje içindeki girdi nesneleri, Şekil B.4 içinde görüldüğü üzere, özet bilgi sağlarlar. Asıl girdi bilgisi altlarındaki "*Data*" nesnesi içinde tutulmaktadır. Ayırca, girdi üretirmindeki ara veriyi belirten "*IntermediaData*" isimli bir alt nesneler de bulunmaktadır.



Şekil B.5: Algoritma ayarları



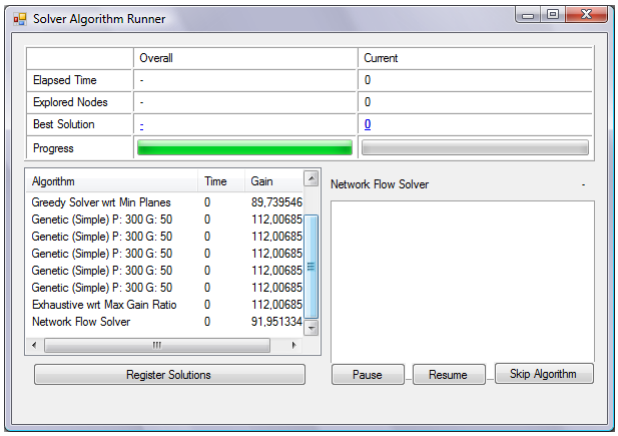
Şekil B.6: Tek bir algoritmanın ayar ayrıntıları

B.4 Algoritma Ayarları

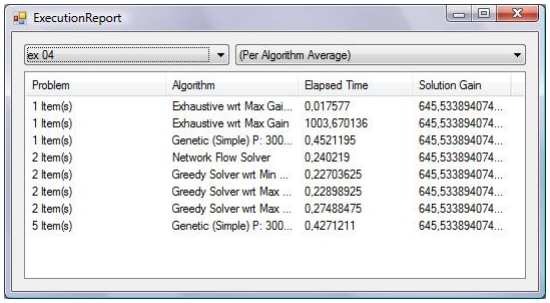
Sınamalarda hangi algoritmalaron, ve hangi parametreler ile çalıştırılacağı, projenin "*Configuration*" nesnesi ile tanımlanmaktadır. Şekil B.5 içinde görülen arayüz ile bu algoritma listesi yönetilebilir. Ayrıca bir öğenin üzerine çift tıklanarak ayrıntılı şekilde ayarlanması da mümkündür (Şekil B.6).

B.5 Algoritma İşletme

Bir girdinin sınanması için, açılır menüsü üzerinden "*Run All Solver Algorithms*" öğesi seçilir. Bunun sonucunda Şekil B.7 içindeki form görünür hale gelip, çözücüler çalışmaya başlar. Tüm çözücülerin tamamlanmasından sonra "*Register Solutions*" komutu ile sonuçların proje içerisine eklenmesi gerçekleştirilir.



Şekil B.7: Tüm algoritmaların çalıştırılması ve sonuçların elde edilmesi



Şekil B.8: Çözücü sonuçlarının öğrenilmesi

B.6 Sonuçların Çıkarılması

Çözücü sonuçları, projeye bağlı ikinci bir form içerisinde sunulmaktadır. Bu formun filtreleme kabiliyetleri ile, girdi ve algoritma seçeneklerine göre sonuçları özetlemek mümkündür. Ayrıca tüm algoritmaların ortalama çalışma zamanları da sorulabilir. İlgili form Şekil B.8 içerisinde gösterilmiştir.

EK C TERİMLER SÖZLÜĞÜ

Bu tez çalışmasında kullanılan terimlerin İngilizce karşılıkları bu ek içerisinde verilmiştir. Terimlerin karşılıklarının bulunmasında aşağıdaki kaynaklardan faydalanılmıştır:

- Türkiye Bilişim Derneği Bilişim Terimleri Sözlüğü [TBD96, TBD]
- Türk Dil Kurumu Güncel Türkçe Sözlük [TDK]
- İngilizce Türkçe Psikoloji Sözlüğü [Bud]
- Çevrimiçi İngilizce — Türkçe sözlükler

Bir kelimenin birden çok karşılığı bulunması durumunda, günlük Türkçe kullanımda yaygın olan veya bağlama uygun gelen seçenekler tercih edilmiştir.

C.1 İngilizce — Türkçe

AI planning yapay zekâ planlama

Ammunition mühimmat

Ant colony optimization karınca kolonisi eniyileştirmesi

Branch and bound dallan ve sınırla

Combinatorial optimization kombinasyonel optimizasyon

Complexity theory karmaşıklık teorisi

Constraint satisfaction kısıt tatmini

Convergance yakınsama

Decision problem karar verme problemi

Deterministic gerekirci

Domain alan

Dynamic devingen

Effectiveness etkililik

Efficient verimli

Eugenics soylulaştırma

Exhaustive search tam arama

Exponential üssel

Expressiveness ifade gücü

Flow akım

Generalized flow genelleştirilmiş akım

Graph çizge

Hard real-time katı gerçek zaman

Heuristic sezgisel

Hill climbing tepe tırmanma

Initialization ilklendirme

Instance olgu

Interdisciplinary disiplinler arası

Knapsack sırt çantası

Linear programming doğrusal programlama

Loop döngü

Maximum flow en fazla akım

Mixed integer programming tamsayı karışık programlama

Multi objective search çok amaçlı arama

Network flow ağ akışı

Non-deterministric gerekirci olmayan

Operations research yöneylem araştırması

Optimization en iyileme

Performance başarım

Reduction indirgeme

Robustness sağlamlık

Scheduling zamanlama

Scheduler zaman planlayıcı

Simulation benzetim

Transportation taşıma

Weapon target assignment silah hedef ataması

C.2 Türkçe — İngilizce

Ağ akışı network flow

Akım flow

Alan domain

Başarım performance

Benzetim simulation

Çizge graph

Çok amaçlı arama multi objective search

Dallan ve sınırla branch and bound

Devingen dynamic

Disiplinler arası interdisciplinary

Doğrusal programlama linear programming

Döngü loop

En fazla akım maximum flow

En iyileme optimization

Etkililik effectiveness

Genelleştirilmiş akım generalized flow

Gerekirci deterministic

Gerekirci olmayan non-deterministic

İfade gücü expressiveness

İlklendirme initialization

İndirgeme reduction

Karar verme problemi decision problem

Karınca kolonisi eniyileştirmesi ant colony optimization

Karmaşıklık teorisi complexity theory

Katı gerçek zaman hard real-time

Kısıt tatmini constraint satisfaction

Kombinasyonel optimizasyon combinatorial optimization

Mühimmat ammunition

Olgu instance

Sağlamlık robustness

Sezgisel heuristic

Sırt çantası knapsack

Silah hedef ataması weapon target assignment

Soylulaştırma eugenics

Tam arama exhaustive search

Tamsayı karışık programlama mixed integer programming

Taşıma transportation

Tepe tırmanma hill climbing

Üssel exponential

Verimli efficient

Yakınsama convergance

Yapay zekâ planlama AI planning

Yöneylem araştırması operations research

Zaman planlayıcı scheduler

Zamanlama scheduling